Q-lite MLN Protocol v1.8

This document explains how the MLN Protocol can be used to control MLN displays over a serial connection (RS232 or RS485/422) or Ethernet connection.

Contents

Legend

`Byte1 ^ Byte2`	means XOR.
`Word`^{LO}`, Word`^{HI}	means Low and High Byte of a 2-Byte (Word) value.
`00h`	means a Hexadecimal value.
`00000000b`	means a Binary value.
`[Parameter]`	means a parameter consisting of several Bytes.

General Understanding

A Function Code (FC) is used to send, request, or receive data to/from a display. This can include sending Command Codes.

A Command Code (CC) represents content such as text, counters, and page effects.

Methods to show content

FC allow writing CC to the Insert Segment, Initial Segment, or one of 256 Segments. Segments can be activated by a schedule or continue-table, as illustrated below. All methods sorted by their respective priority:

Method	**Prio.**	**Use**	**Related Function Codes**
Insert	1	Insert content that must go live immediately, highest priority.	FC 05h (Send to Insert Segment)
Schedule	2	Schedule content shown (in a loop) during predefined time slots.	FC 04h (Send schedule-table) FC 02h (Send to Segment)
Continue	3	Continual content shown (in a loop) at any time.	FC 03h (Send continue-table) FC 02h (Send to Segment)
Initial	4	Initial content shown, lowest priority.	FC 00h (Send to Initial Segment)

So when the Insert method has no data, it tries Schedule, then Continue, then Initial. If no method has data the display will show “Waiting…” on the first line.

The sender (your software) can use any method or a combination thereof.

Data communication

For your convenience, the default communication configuration is:

Serial baud rate	9600 bps
Serial framing	8N1 (8 data bits, no parity bits, 1 stop bit)
Receiver ID	0
TCP port	9100

Depending on your display model and configuration this configuration may differ (except for serial framing).

As always, a serial bus (even if transported over a TCP/IP network) does not allow crosstalk. If you request a reply from multiple displays make sure to do so sequentially and wait for each respective response before sending so data does not collide.

Message Format

A message (packet) consists of Message Header, Message Body, and Ending:

Message Header								
Protocol ID		Sender ID		Receiver ID		Message Length		Header Checksum
AAh	BBh	SID^{HI}	SID^{LO}	RID^{HI}	RID^{LO}	$Length^{HI}$	$Length^{LO}$	LRC

	Message Body				Ending
	Function Code	Message Bytes			Checksum
	FC	[Data1	Data2, …	Data n]	CS

Specification

1. Protocol ID always is AAh BBh.
2. SID is the Sender ID number.
3. RID is the Receiver ID number. Valid values / ranges are as follows:

RID^{HI}	RID^{LO}	Transmission type
FFh	FFh	Broadcast to all displays.
00..FEh	FFh	Broadcast to group of matching RID^{HI}.
00..FEh	00..FEh	Send to display with matching 16-bit ID.

   Other values / ranges are reserved.
4. Length is the total number of Bytes in Message Body (from FC to Data n), not including CS. Note that a Message Body can not exceed 64KByte.
5. LRC is the checksum of the Message Header, which is

   $LRC = SID^{HI} \wedge SID^{LO} \wedge RID^{HI} \wedge RID^{LO} \wedge Length^{HI} \wedge Length^{LO}$
6. When sending big packets, allow the time to write the data into the flash memory and prevent the receiving queue (4KByte) from overflow. Some FC use RAM only and are significantly faster. Append FC 21h (Request for display transmission result) to determine when the queue is processed.
7. Function Codes (FC) are the commands sent to the display(s). Message Bytes are FC parameters. See the chapter Function Codes.
8. CS is the checksum of the Message Body, which is the XOR of the Message Body:

   $CR = FC \wedge MessageByte\_1 \wedge MessageByte\_2 \wedge .. \wedge MessageByte\_n$

   Note that for any FC that has no parameters (thus is 1 Byte long) the CS value = FC value.

Function Codes

If you are unfamiliar with FC usage, see General Understanding first.

Covered in this document are:

FC 00h (Send to Initial Segment)

The Initial Segment is shown when no Insert is shown, and no segments are active (either not set in the continue-table, or scheduled, or empty).

The message body is as follows:

```
00h, [DisplayData_1], [DisplayData_2], .. [DisplayData_n]
```

DisplayData	Command Codes and their associated parameters or data if any.

FC 01h (Set clock)

The message body is as follows

```
01h, Year, Month, Day, DayOfWeek, Hour, Minute, Second
```

Year	The years since 2000 (where value 01h is the year 2001).
Month	The month.
Day	The day.
DayOfWeek	Valid values are as follows: 00h Sunday 01h Monday 02h Tuesday 03h Wednesday 04h Thursday 05h Friday 06h Saturday
Hour	The hour.
Minute	The minute.
Second	The second.

FC 02h (Send to Segment)

Segments are used by the Schedule-table and Continue-table to hold content.

The message body is as follows:

```
02h, Segment, [CommandCode_1], [CommandCode_2], .. [CommandCode_n]
```

Segment	The program segment to receive the program. Valid values are in range 0 .. 255.
CommandCode	See Command Codes.

FC 03h (Send continue-table)

The continue table loops through its segments shows these when no Insert is shown, and no segments are active (either scheduled, or empty).

The message body is as follows:

```
03h, Segment_1, Segment_2, Segment_3, .. Segment_256
```

`Segment`	The display segment to receive the program. Valid values are in range 0 .. 255.

Any parameter count up and including 256 is valid. A parameter count of 0 clears the table.

FC 04h (Send schedule-table)

The schedule table shows the time slotted segments when no Insert is shown.

The message body is as follows:

```
04h, [Timeslot_1], [Timeslot_2], .. [Timeslot_128]
```

`Timeslot`	The scheduled time slots, as outlined below.

Any parameter count up to 128 is valid. A parameter count of 0 clears the table.
The Timeslot format is as follows:

```
ST_Y, ST_M, ST_D, ST_TLO, ST_THI,
EN_Y, EN_M, EN_D, EN_TLO, EN_THI,
WeekdaySwitch, SegmentCount, Segments_1, .. Segments_12
```

`ST_Y`	Schedule starting year (where 00h = the year 1900)
`ST_M`	Schedule starting month
`ST_D`	Schedule starting date
`ST_TLO`	Program starting time, low byte
`ST_THI`	Program starting time, high byte
`EN_Y`	Schedule ending year (where 00h = the year 1900)
`EN_M`	Schedule ending month
`EN_D`	Schedule ending date
`EN_TLO`	Program ending time, low byte
`EN_THO`	Program ending time, high byte
`WeekdaySwitch`	Weekday scheduling flags, is a bitmask: `0xxxxxx1b` Sunday (bit 0) `0xxxxx1xb` Monday (bit 1) `0xxxx1xxb` Tuesday (bit 2) `0xxx1xxxb` Wednesday (bit 3) `0xx1xxxxb` Thursday (bit 4) `0x1xxxxxb` Friday (bit 5) `01xxxxxxb` Saturday (bit 6)
`SegmentCount`	Number of segments to be included in the list that follow.
`Segments`	Twelve segment numbers. Valid values are in range 0..255.

Fill unused slots with 0.

The Time (ST_T and EN_T) values are as follows:

```
Time = Hours x 60 + Minutes
```

The scheduled message is shown if:

```
( ST_T <= CurrentTime <= EN_T ) AND
(
  ( ST_Y / ST_M / ST_D <= CurrentDate <= EN_Y / EN_M / EN_D ) AND
  (
    ( DayOfWeek = Sunday    AND WeekdaySwitch = 0xxxxxx1 ) OR
    ( DayOfWeek = Monday    AND WeekdaySwitch = 0xxxxx1x ) OR
    ( DayOfWeek = Tuesday   AND WeekdaySwitch = 0xxxx1xx ) OR
    ( DayOfWeek = Wednesday AND WeekdaySwitch = 0xxx1xxx ) OR
    ( DayOfWeek = Thursday  AND WeekdaySwitch = 0xx1xxxx ) OR
    ( DayOfWeek = Friday    AND WeekdaySwitch = 0x1xxxxx ) OR
    ( DayOfWeek = Saturday  AND WeekdaySwitch = 01xxxxxx )
  )
)
```

FC 05h (Send to Insert Segment)

The Insert Segment interrupts the regularly scheduled display sequences and is only shown for the duration of Delaytime. This Insert Segment is not saved in flash memory and will be lost after a powerdown.

The message body is as follows:

```
05h, Delaytime, [DisplayData_1], [DisplayData_2], ..[DisplayData_n]
```

`Delaytime`	The amount of minutes that insert will be shown.
`DisplayData`	As in FC 00h (Send to Initial Segment).

FC 06h (Set worktime)

The message body is as follows:

```
06h, Worktime_On^LO, Worktime_On^HI, Worktime_Off^LO, Worktime_Off^HO
```

The Time (Worktime_On and Worktime_Off) values are as follows:

```
Time = Hours x 60 + Minutes
```

Example: operation from 6:00 to 23:30 is 06h 0168h 0582h E8h

FC 07h (Clear all)

Display will clear all segments, initial segment, insert segment, the continue table, and the schedule table. It now shows “Waiting…” in the first line of the display, also see Methods to show content.

No data field is required. (Note that Checksum = 07h, as pointed out in General Understanding)

FC 09h (Send dimmer-table)

Configure the dimmer table and threshold. This can be used to switch external devices.
The message body is as follows:

```
09h, Table_1, Table_2, .. Table_256, RelayThreshold
```

`Table`	Dimmer table data.
`RelayThreshold`	Relay is engaged when (the highest) light measurement is lower.

FC 0Ah (Send heater threshold)

The message body is as follows:

```
1Fh, Humidity, HighTemperature, LowTemperature
```

`Humidity`	Byte value resembling humidity value.
`HighTemperature`	Temperature in degrees Celsius.
`LowTemperature`	Temperature in degrees Celsius.

The heater will be enabled if:

- InternalTemperature is below Low, or is below zero, or
- InternalTemperature is between Low and High, and InternalHumidity is above Humidity. (note for stability there is an extra 5% threshold on Humidity before enabling/disabling)

FC 0Bh (Handle leading zeroes)

This applies to how Counters are formatted when shown.

The message body is as follows:

```
0Bh, LeadingChar
```

`LeadingChar` Valid values are as follows:

`00h`	Padded with “0”	(digit zero)
`01h`	Padded with “ “	(space)

FC 18h (Set parallel-out)

Output a value via the parallel output. This can be used to toggle external devices.

The message body is as follows:

```
18h, Value
```

`Value`	The data (8 bit binary) to be set on the parallel output port.

FC 1Dh (Set display XY unit)

Tell the display how to interpret X Y coordinates, either in Pixel or Character units.

The message body is as follows:

```
1Dh, Switch
```

Switch	Valid values are as follows:		
	00h	Characters as unit	(in current active font)
	NOT 00h	Pixels as unit	

FC 1Eh (Send bitmap)

The message body is as follows:

```
1Eh, Index, Width, Height,
[BitmapR_0, BitmapR_1, .. BitmapR_n],
[BitmapG_0, BitmapG_1, .. BitmapG_n],
[BitmapB_0, BitmapB_1, .. BitmapB_n]
```

Index	Index referring to that bitmap. Valid values are in range 0..255.
Width	Width of the bitmap in pixels.
Height	Height of the bitmap in pixels.
BitmapR	Every bit is one Red pixel.
BitmapG	Every bit is one Green pixel. *
BitmapB	Every bit is one Blue pixel. *

*) Both BitmapG and BitmapB are optional.
See related CC 1Eh (Show bitmap).

FC 1Fh (Start pixeltest)

Starts a pixel test, after which data can be requested using FC 2Fh (Request for display pixeltest data). The display shows content again when the pixeltest is complete.

The message body is as follows:

```
1Fh, ColorChannel
```

ColorChannel	A bitmask, valid are:		
	00000xx1b	Red	(bit 0)
	00000x1xb	Green	(bit 1)
	000001xxb	Blue	(bit 2)

FC 20h (Send ping)

When the display receives this Ping message it will send this Ping message back to the transmitter.

No data field is required.

FC 21h (Request for display transmission result)

The sign will return the transmission result of the previous message.
The response from the display is FC 31h (Response from display with transmission result).

No data field is required.

FC 22h (Request for display status)

The sign will return its Status. This can be used to retrieve actual clock and temperature values. The response from the display is FC 32h (Response from display with status).

No data field is required.

FC 23h (Request for display Initial Segment data)

The response from the display is FC 33h (Response from display with Initial Segment data).

No data field is required.

FC 24h (Request for display Segment data)

The response from the display is FC 34h (Response from display with Segment data).

The message body is as follows:

```
24h, Segment
```

`Segment` Valid values are in range 0..255.

FC 25h (Request for display continue-table data)

The response from the display is FC 35h (Response from display with continue-table data).

No data field is required.

FC 26h (Request for display schedule-table data)

The response from the display is FC 36h (Response from display with schedule-table data).

No data field is required.

FC 2Bh (Request for display scanbuffer data)

Retrieves a screenshot of the display. The response from the display is FC 3Bh (Response from display with scanbuffer).

The message body is as follows:

```
2Bh, Buffer
```

`Buffer` Valid values are:

`00h`	Red
`01h`	Green
`02h`	Blue

FC 2Ch (Request for counter/variable data)

Retrieves current display counter and variable values. The response from the display is FC 3Ch (Response from display with counter/variable data). See related FC 40h (Set variable), FC 41h (Set counter).

No data field is required.

FC 2Fh (Request for display pixeltest data)

Retreives the latest pixel test data. See related FC 1Fh (Start pixeltest), which should be send prior.

The response from the display is FC 3Fh (Response from display with pixeltest data).
The message body is as follows:

```
2Fh, ColorChannel
```

`ColorChannel` Valid values are:

`00000001b`	Red	(bit 0)
`00000010b`	Green	(bit 1)
`00000100b`	Blue	(bit 2)

FC 30h (Response from display for ping without encryption)

This FC is sent in response to FC 20h (Send ping) when display requires encryption.

FC 31h (Response from display with transmission result)

The message format is described in General Understanding. In this case the sender is the display and the receiver is the controller such as a PC. This FC is send in response to FC 21h (Request for display transmission result).

The return message body is as follows:

```
31h, Comm_status
```

`Comm_status` Valid values are as follows:

`60h`	Normal
`61h`	Timeout error
`62h`	Checksum error
`63h`	Overflow error
`64h`	Functioncode error
`65h`	Parameter error
`66h`	LRC error
`67h`	Decrypt error

FC 32h (Response from display with status)

The message format is described in the General Understanding. In this case the sender is the display and the receiver is the controller such as a PC. This FC is sent in response to FC 22h (Request for display status).

The return message body is in the following format:

```
ColumnsLO, ColumnsHI,
Scanheight, Rows,
FPGA_Scanheight, FPGA_Rows,
Gap_width,
RefreshLO, RefreshHI,
Flashrate,
[Userdata_1, Userdata_2, .. Userdata_128],
FirmwareTypeLO, FirmwareTypeHI,
FirmwareVersionLO, FirmwareVersionHI,
[Description_1, Description_2, .. Description_16],
HardwareVersionLO, HardwareVersionHI,
Worktime_OnLO, Worktime_OnHI,
Worktime_OffLO, Worktime_OffHI,
InternalTemp_Value, InternalTemp_Available,
ExternalTemp_Value, ExternalTemp_Available,
ADC0_Value, ADC1_Value, ADC2_Value, ADC3_Value,
DimmerValue,
Hours, Minutes, Seconds,
Day, Month, Year,
DayOfWeek,
ColorChannel0, ColorChannel1, ColorChannel2,
OptionByte,
ExternalParallelInput, InternalParallelInput
```

All relevant value formats are equal to their respective FC, others are explained below.

`FirmwareType` — Valid values are:

0000h	MLN
0001h	QLG
0002h	QLM |
0003h	QLG SMD

`FirmwareVersion` — The HI and LO bytes respectively denote the major and minor version.

`HardwareVersion` — Highest four bits of value are a bitmask for hardware type:

0000xxxxb	Multiline
0001xxxxb	Graphic
0010xxxxb	Zone 30
0011xxxxb	Graphics SMD
0100xxxxb	MLN Pixeltest Num
1111xxxxb	STORM keypad

The remaining HI bits and LO byte respectively denote the major and minor version.

`Userdata..` — 128 ASCII characters.

`Description..` — 16 ASCII characters, often the display serial number.

`Worktime..` — See FC 06h (Set worktime).

`..Temp_Available` — Valid values are:.

	00h Available
	NOT 00h Not available
..Temp_Value	Signed Byte value in degrees Celsius.
Hours .. Year	See FC 01h (Set clock).
DayOfWeek	Valid values are: (differs from FC 01h (Set clock))
	01h Sunday
	02h Monday
	03h Tuesday
	04h Wednesday
	05h Thursday
	06h Friday
	07h Saturday
ColorChannel	Valid values are:
	00h None
	01h Red
	02h Green
	03h Blue
	04h Yellow
	05h White
OptionByte	Value is a bitmask:
	xxxxxxx1b Display has a physical fontgap (bit 0)
..ParallelInput	Read the internal/external parallel input ports, 8 bits each. This can be used to read external devices/states.

FC 33h (Response from display with Initial Segment data)

The message format is described in the General Understanding. The sender is the display and the receiver is the controller such as a PC. This FC is send in response to FC 23h (Request for display Initial Segment data).

The return message body is the same as in FC 00h (Send to Initial Segment).

FC 34h (Response from display with Segment data)

The message format is described in the General Understanding. The sender is the display and the receiver is the controller such as a PC. This FC is send in response to FC 24h (Request for display Segment data).

The return message body is the same as in FC 02h (Send to Segment).

FC 35h (Response from display with continue-table data)

This FC is send in response to FC 25h (Request for display continue-table data).

The message body is as follows:

```
35h, Segment_1, Segment_2, Segment_3, ..  Segment_n
```

Segment	The display segment in the continue table.

Valid values are in range 0 .. 255.

FC 36h (Response from display with schedule-table data)

This FC is send in response to FC 26h (Request for display schedule-table data).

The message body is as follows:

```
36h, [Timeslot_1], [Timeslot_2], .. [Timeslot_128]
```

`Timeslot` As in FC 04h (Send schedule-table).

FC 3Bh (Response from display with scanbuffer)

This FC is send in response to FC 2Bh (Request for display scanbuffer data).

The message body is as follows:

```
3Bh, PixelByte_1, PixelByte_2, .. PixelByte_8193
```

`PixelByte` Every bit is one pixel.

This response always includes 65544 pixels.

FC 3Ch (Response from display with counter/variable data)

This FC is send in response to FC 2Ch (Request for counter/variable data).

The message body is as follows:

```
3Ch,  [Count1_1, Count1_2, Count1_3, Count1_4],
      [Count2_1, Count2_2, Count2_3, Count2_4],
      ..
      [Count32_1, Count32_2, Count32_3, Count32_4],
      [Reserved1, Reserved2, .. Reserved128],
      [Var1_1, Var1_2, .. Var1_31, Var1_32],
      [Var2_1, Var2_2, .. Var2_31, Var2_32],
      ..
      [Var16_1, Var16_2, .. Var16_31, Var16_32]
```

`Count..` 16 down- and 16 up-counters respectively, 4 ASCII bytes each.
`Reserved..` 128 reserved bytes / garbage data.
`Var..` 16 variables, 32 ASCII bytes each.

FC 3Fh (Response from display with pixeltest data)

The message body is as follows:

```
3Fh, ColumnsHI, ColumnsLO, Rows, Scanheight, Gap_Width, Font_Width,
       [ PixColumnHI, PixColumnLO, PixRow ],
                          ..
       [ PixColumnHI, PixColumnLO, PixRow ]
```

`Columns` Width in pixels.

`Rows`	Height in pixels.
`Gap_Width`	Width of the gap in pixels.
`Font_Width`	Width of the Font (at index 0).
`PixColumn`	Column of a defect pixel.
`PixRow`	Row of a defect pixel.

FC 40h (Set variable)

Set a variable in the display. See related CC 0Ch (Show variable).

The format is as follows:

```
40h, Index, Chars_1, Chars_2, Chars_3, .. Chars_n
```

`Index`	The variable to store. Valid values are in range 00h .. 0Fh. (0..15)
`Chars`	String of 1 up to 32 ASCII characters.

FC 41h (Set counter)

Set a variable in the display. See related CC 0Dh (Show counter).
Counters are updated at 0:00 hours (midnight). Index 0..15 is incremented, 16..31 is decremented.

The format is as follows:

```
41h, Index, Digits_1, Digits_2, Digits_3, Digits_4
```

`Index`	The counter to store. Valid values are in range 00h .. 1Fh. (0..31)
`Digits`	Four ASCII values. Valid values are in range 30h .. 39h.

FC F0h (Reboot display)

No data field is required.

Command Codes

If you are unfamiliar with CC usage, see Function Codes first.

Most Command Codes operate on a buffer, which is displayed by CC 07h. The buffer is not cleared automatically, so one needs to issue CC 01h first.

Covered in this document are:

Popular Parameters

Some parameters are not unique to a specific CC, and only require a detailed explanation here.

Values may depended on your display model and configuration, as outlined here.

`Font`
Font to use, default is 00h. Value 01h usually is a bold font.
Note: values can vary per display model.

`Fcolor`
Foreground color. Valid values are:

`00h`	Black
`01h`	Red
`02h`	Green
`03h`	Yellow

Note: values can vary per display model. Also see related FC 32h (Response from display with status).

`Bcolor`
Background color. See Fcolor above.

`X, Y`
The position of the character, where Y is the line (vertical).
Note: Units can be Characters or Pixels. Characters are calculated using the current Font:
X range = 0 to n-1 (where n is display width in characters)
Y range = 0 to n-1 (where n is display height in characters)
Value 0,0 is top left.
Also see related FC 1Dh (Set display XY unit).

CC 01h (Clear buffer)

Clears the display contents.

The format is as follows:

```
01h
```

No parameters.

CC 03h (Show text string)

The display data are as follows.

```
03h, Font, Fcolor, Bcolor, X, Y, Text_String, 00h
```

`Text_String` — The ASCII text string.
Accepts the following inline special modifier codes

- `~Cn;` Change the foreground color, where n is:
  - "0" Black
  - "1" Red
  - "2" Green
  - "3" Yellow
- `~Bn;` Change the background color, n see above.
- `~R;` Reverse foreground and background colors.
- `~F;` Flash the following characters.
- `~N;` Do not flash the following characters.
- `~I;` Revert back to initial Fcolor and Bcolor value.
- `~Pn;` Change the parallel output value, where n is a 1 to 3-digit ASCII value*.
  See FC 18h (Set parallel-out).
- `~An;` Change font, where n is a 1-digit ASCII value.
- `~Dn;` Outputs 8 bits over 8 columns (used for dots / arrows), where n is a 1 to 3-digit ASCII value*.

*) Valid values are in range "0" .. "255".
This text should not contain a Byte equal to 00h.

`00h` — String terminator.

Parameters missing in this list are detailed in Popular Parameters.
Known issue: certain modifier codes wont work when they appear at the end of a text.

CC 04h (Show current time)

The display data format is as follows:

```
04h, Type, Font, Fcolor, Bcolor, X, Y
```

`Type` — Format of the shown time. Valid values are:

00h	"HH:MM:SS AM"	12-hour, with AM/PM
01h	"HH:MM:SS"	24-hour
02h	"HH:MM"	24-hour, short

Re-purposing the here affected pixels must be preceded by issuing CC 01h (Clear buffer).
Parameters missing in this list are detailed in Popular Parameters.

CC 05h (Show current date)

The display data format is as follows:

```
05h, Type, Font, Fcolor, Bcolor, X, Y
```

`Type` Format of the shown date. Valid values are:

00h	"MM/DD/YYYY"	(09/30/1998)
01h	"MMM. DD, YYYY"	(SEP. 30, 1998)
02h	"DD MMM. YYYY"	(30 SEP. 1998)
03h	"DD/MM/YYYY"	
04h	"DD/MM/YY"	
05h	"DD/MM"	
06h	"DD"	
07h	"MM"	
08h	"YY"	

Note support of 01h and 02h and its locale/language may vary per display model.

Re-purposing the here affected pixels must be preceded by issuing CC 01h (Clear buffer).
Parameters missing in this list are detailed in Popular Parameters.

CC 06h (Delay)

Delay during which time and counters are not updated.

The message body format is as follows:

```
06h Delaytime^LO, Delaytime^HI
```

`Delaytime` How much time in milliseconds to wait.

CC 07h (Display the buffer)

The message body format is as follows:

```
07h, Delaytime^LO, Delaytime^HI, Mode, Speed
```

`Delaytime` How much time in milliseconds to wait.

`Mode` The effect to show the display content. Valid values are:

00h	Instant
01h	Scan Right
02h	Scan Left
03h	Scan Down
04h	Scan Up
05h	Shift Right

	06h	Shift Left
	07h	Push Down
	08h	Push Up
	09h	Flow
	0Ah	Cover Right
	0Bh	Cover Left
	0Ch	Cover Down
	0Dh	Cover Up
	0Eh	Scan Open
	0Fh	Scan Close
	10h	Push Open
	11h	Push Close
	12h	Cover Open
	13h	Cover Close
	14h	Sparkle
Speed	Speed of effect. Valid value range is 00h (slow) to 09h (fast).	

CC 08h (Show text immediately)

Show a string as a scroll-text, while all other lines remain static.

The display data format is as follows:

```
08h, Mode, Speed, Font, Fcolor, Bcolor, X, Y, Text_String, 00h
```

Mode	Valid values are:	
	00h	Instant
	05h	Shift Right
	06h	Shift Left
Speed	Speed of scroll. Valid value range is 00h (slow) to 09h (fast).	

Parameters missing in this list are identical to CC 03h (Show text string).

Usage:

For a scrolling text use Mode=06h.
If you do not want to see the preexisting line contents scroll out of view, you can first issue a Mode=00h (with enough space characters as Text_String) to clear the line instantly.
Do not issue CC 07h (Display the buffer) after CC 08h (Show text immediately) without having issued CC 01h (Clear buffer) first. Or more precisely: having drawn to all pixels which can also be accomplished using others like CC 03h (Show text string).

CC 09h (Show countdown to date)

Shows a countdown to a specified date. Days are shown 4 characters width, prefixed with spaces. Behaves like CC 0Ah (Show countdown to date and time) where Hour=0, Minute=0.

The display data format is as follows:

```
09h, Type, Font, Fcolor, Bcolor, X, Y, YearLO,YearHI, Month, Day
```

Type	Format of the shown countdown. Valid values are:

	00h	“DDDD”	(show days only)
	01h	“DDDD HH:MM:SS”	(includes time)
Year	Count down to year (where value 07D0h is the year 2000).		
Month	Count down to month.		
Day	Count down to day.		

Re-purposing the here affected pixels must be preceded by issuing CC 01h (Clear buffer).
Parameters missing in this list are detailed in Popular Parameters.

CC 0Ah (Show countdown to date and time)

Shows a countdown to a specified date and time. Days are shown 4 characters width, prefixed with spaces.

The display data format is as follows:

```
0Ah, Type, Font, Fcolor, Bcolor, X, Y, YearLO, YearHI, Month, Day,
Hour, Minute
```

Type	Format of the shown countdown. Valid values are:		
	00h	“DDDD”	(show days only)
	01h	“DDDD HH:MM:SS”	(includes time)
Year	Count down to year (where value 07D0h is the year 2000).		
Month	Count down to month.		
Day	Count down to day.		
Hour	Count down to hour.		
Minute	Count down to minute.		

Re-purposing the here affected pixels must be preceded by issuing CC 01h (Clear buffer).
Parameters missing in this list are detailed in Popular Parameters.

CC 0Bh (Show temperature)

The display temperature format is as follows:

```
0Bh, Type, Font, Fcolor, Bcolor, X, Y
```

Type	Format of the shown temperature.		
	00h	“+XX°C”	External, in Celcius.
	03h	“+XX°C”	Internal, in Celcius.

Parameters missing in this list are detailed in Popular Parameters.

CC 0Ch (Show variable)

Show the variable, which is a string of upto 32 characters. See related FC 40h (Set variable).

The format is as follows:

```
0Ch, Index, Font, Fcolor, Bcolor, X, Y
```

`Index` The variable to show. Valid values are in range 00h .. 0Fh. (16)

Parameters missing in this list are detailed in Popular Parameters.

CC 0Dh (Show counter)

Counters are shown 4 characters width, prefixed with zeroes. See related FC 41h (Set counter).

The format is as follows:

```
0Dh, Index, Font, Fcolor, Bcolor, X, Y
```

`Index` The counter to show. Valid values are in range 00h .. 1Fh. (32)

Parameters missing in this list are detailed in Popular Parameters.

CC 0Eh (Loop a CC block / start)

Repeat the command codes between CC 0Eh and CC 0Fh.
The code has to be used together with CC 0Fh (Loop a CC block / end) below.

The format is as follows:

```
0Eh, TimesLO, TimesHI
```

`Times` The number of times to repeat this block of Command Codes.

CC 0Fh (Loop a CC block / end)

Loop the command codes between CC 0Eh and CC 0Fh.
The code has to be used together with CC 0Eh (Loop a CC block / start) above.

The format is as follows:

```
0Fh
```

No parameters.

CC 1Eh (Show bitmap)

The format is as follows:

```
1Eh, XLO, XHI,Y, Index
```

`X` The X position of the bitmap in pixels.
`Y` The Y position of the bitmap in pixels.
`Index` The index of the bitmap to be shown.

See related FC 1Eh (Send bitmap).

CC 1Dh (Freeze display)

Shows the buffer instantly and freezes CC execution. Shown time / counters will be updated. The current display contents will be shown indefinitely until a new message (with new Command Codes) is received.

See related CC 07h (Display the buffer), which allows a maximum of FFFFh msec delay.

The format is as follows:

```
1Dh
```

No parameters.

CC 1Fh (End of segment data)

Signifies the end of segment data.

The format is as follows:

```
1Fh
```

No parameters.

Appendix A: Examples

Example 1: Send a text message

Here is a full example (including header and checksums) to send a text message to a display.

```
AAh BBh                 Protocol identifier   (General Message Format)
00h 00h                 Sender ID
00h 00h                 Receiver ID (the display)
00h 13h                 Length of the Message Body (13h = 19)
13h                     LRC of header (00h^00h^00h^00h^00h^13h)

00h                     Send to Initial Segment                (FC 00h)

 01h                    Clear display                          (CC 01h)

 03h                    Draw a text string                     (CC 03h)
  00h                   Use font 0
  03h                   Foreground color (3 = Yellow)
  00h                   Background color (0 = Black)
  00h                   X = 0
  00h                   Y = 0
  54h 45h 58h 54h       String "TEXT" represented in ASCII
  00h                   String terminator

 07h                    Show the data on the display           (CC 07h)
  E8h                   DelayLO
  03h                   DelayHI (03E8h = 1000 msec)
  00h                   Mode 00h = Instant
  09h                   Speed 09h = fastest

 1Fh                    End of display data                    (CC 1Fh)

E6h                     Ending checksum       (General Message Format)
```

To check if the display has processed the frame already you can send a FC 21h (Request for display transmission result) to the display. When the display sends back its response FC 31h (Response from display with transmission result) it is finished processing this frame.

If a transmission error has occurred (the status Byte will indicate this) then you could act on it by resending the frame, for example.

Example 2: Send a text message with two lines

Here is example 1 with addition of displaying a 2nd line, illustrating that more Bytes are affected by (even simple) changes in a message.

```
AAh BBh                 Protocol identifier   (General Message Format)
00h 00h                 Sender ID
00h 00h                 Receiver ID (the display)
00h 1Dh                 Length of the Message Body (1Dh = 29)
1Dh                     LRC of header (00h^00h^00h^00h^00h^1Dh)

00h                     Send to Initial Segment               (FC 00h)

 01h                    Clear display                         (CC 01h)

 03h                    Draw a text string                    (CC 03h)
  00h                   Use font 0
  03h                   Foreground color (3 = Yellow)
  00h                   Background color (0 = Black)
  00h                   X = 0
  00h                   Y = 0
  54h 45h 58h 54h       String "TEXT" represented in ASCII
  00h                   String terminator

 03h                    Draw a text string, on the 2nd line   (CC 03h)
  00h                   Use font 0
  01h                   Foreground color (1 = Red)
  00h                   Background color
  00h                   X = 0
  01h                   Y = 1
  31h 32h 33h           String "123" represented in ASCII
  00h                   String terminator

 07h                    Show the data on the display          (CC 07h)
  E8h                   DelayLO
  03h                   DelayHI (03E8h = 1000 msec)
  00h                   Mode 00h = Instant
  09h                   Speed 09h = fastest

 1Fh                    End of display data                   (CC 1Fh)

D5h                     Ending checksum       (General Message Format)
```

Notice how the addition of 10 Bytes (Command Code 03h showing "123") affects more bytes:

Length of the Message Body	due to addition of CC 03h.
Message Header Checksum	due to above mutation in Message Header.
Ending Checksum	due to addition of CC 03h.

Example 3: Send a scrolling text

Below you will find some small example snippets on how to add scrolling text functionality to a program.

These snippets can be merged together with existing programs (so first show some static text and then let some text line scroll), or you can simply us these as is.

Normally we first clear the line where the scrolling text will be displayed:

```
08h                  Show text string immediately          (CC 08h)
00h                  Mode 00h = Instant
09h                  Speed 09h = fastest
00h                  Use font 0
03h                  Foreground color (3 = Yellow)
00h                  Background color (0 = Black)
00h                  X = 0
00h                  Y = 0
20h 20h .. 20h       String represented in ASCII
                     “                    “
                     (enough spaces to clear the line)
00h                  String terminator
```

Now we show the text which must be scrolling from right to left:

```
08h                  Show text string immediately          (CC 08h)
06h                  Mode 06h = Push Left
09h                  Speed 09h = fastest
00h                  Use font 0
03h                  Foreground color (3 = Yellow)
00h                  Background color (0 = Black)
00h                  X = 0
00h                  Y = 0
54h 45h 58h 54h 20h
54h 45h 58h 54h 20h
54h 45h 58h 54h 20h
54h 45h 58h 54h      String represented in ASCII
                     “TEXT TEXT TEXT TEXT”
00h                  String terminator
```

When the scrolling text is done we normally want to wait a little:

```
06h                  Wait                                  (CC 06h)
E8h                  DelayLO
03h                  DelayHI (03E8h = 1000 msec)
```

Now we can continue with other texts as shown in the previous example.

Example 4: Using bitmaps

Here is a full example (including header and checksums) to send a bitmap (referred to with index 0) to the display.

```
AAh BBh                 Protocol identifier
00h 00h                 Sender ID
00h 00h                 Receiver ID (the display)
00h 44h                 Length of the message body (16x2x2+4=68=44h)
44h                     LRC of header (00^00^00^00^00^44)

1Eh                     Send Bitmap to display                  (FC 1Eh)
00h                     Index
10h                     Width (10h = 16)
10h                     Height (10h = 16)

87h E1h 48h 12h ..      Bitmap data for 1st color (red)
..  04h 12h 07h E1h

87h E1h 48h 12h ..      Bitmap data for 2nd color (green)
..  04h 12h 07h E1h

66h                     Ending checksum
```

Here is a full example (including header and checksums) to show a bitmap referred to with index 0 on the display.

```
AAh BBh                 Protocol identifier
00h 00h                 Sender ID
00h 00h                 Receiver ID (the display)
00h 0Eh                 Length of the message body (0Eh = 14)
0Eh                     LRC of header (00^00^00^00^00^0E)

02h                     Send segment to display
 00h                    Segment 0

 01h                    Clear display

 1Eh                    Draw a bitmap (at 0,0)                  (CC 1Eh)
  00h                   Bitmap X LOW byte
  00h                   Bitmap X HIGH byte
  00h                   Bitmap Y
  00h                   Bitmap Index

 07h                    Show the data on the display            (CC 07h)
  E8h                   Delay LOW byte
  03h                   Delay HIGH byte (03E8h = 1000 msec)
  00h                   Mode 00h = Instant
  09h                   Speed 09h = fastest

 1Fh                    End of display data                     (CC 1Fh)

E7h                     Ending checksum
```

Example 5: Send a text message with two pages

Here is an example that illustrates how to apply page effects in a segment.

```
AAh BBh                 Protocol identifier   (General Message Format)
FDh E8h                 Sender ID
00h 00h                 Receiver ID (the display)
00h 23h                 Length of the Message Body
36h                     LRC of header

02h 00h                 Send to Segment index 0                (FC 02h)

 01h                    Clear display                          (CC 01h)

 03h                    Draw a text string                     (CC 03h)
  00h                   Font
  01h                   Foreground color (red)
  00h                   Background color (black)
  00h                   X
  00h                   Y
  46h 6Fh 6Fh           String “Foo” represented in ASCII
  00h                   String terminator

 07h                    Page effect                            (CC 07h)
  D0h 07h               Delay (07D0h = 2 seconds)
  05h                   Mode (Shift Right)
  09h                   Speed (fastest)

 01h                    Clear display                          (CC 01h)

 03h                    Draw a text string                     (CC 03h)
  00h                   Font
  02h                   Foreground color (green)
  00h                   Background color
  00h                   X
  00h                   Y
  42h 61h 71h           String “Bar” represented in ASCII
  00h                   String terminator

 07h                    Show the data on the display           (CC 07h)
  D0h 07h               Delay
  06h                   Mode (Shift Left)
  09h                   Speed

 1Fh                    End of display data                    (CC 1Fh)

09h                     Ending checksum       (General Message Format)
```

Notice how this example used FC 02h (Send to Segment) although this is not required, and there is a repeating sequence of CC per page that starts with 01h and ends with 07h.

Appendix B: Default character set

The character set for font 0 (Western 5x7 Ansi) is shown here.