LED Display Programmer's Guide

Version 5.0.1

1 About this manual

This manual is intended to be used by professional programmers in order to develop an application that supplies data to LED displays.

The manual assumes that the programmer knows how to program a serial (or TCP/IP) link to the display.

1.1 Numbers

- Binary: 01000100_2
- Decimal: 68 or 68_{10}
- Hexadecimal: 0x44 or 44_{16}

1.1.1 Restriction on hexadecimal digits in data fields

Display firmware requires that hexadecimal digits from 'A' to 'F' must always be capital letters. Lower case variants: 'a' to 'f' - are handled as errors.

1.2 Characters and strings

All character representations referred to in the manual are using ASCII encoding.

- Single (capital A) character: 'A'
- Multiple characters: “ABCD”
- Not displayable characters are often represented in the manual as hexadecimal values: 0x05.

1.3 Version numbers in this manual

The manual's version applies to the firmware version within the LED Display. Higher LED display firmware versions may require an updated manual due to added features or resolved bugs.

Backward compatibility of firmwares is maintained whenever it is possible. Most of the details in this manual will apply to future firmwares.

1.3.1 Firmware version dependent features

Marked as: **[Version >= 1.1.2]**

This means that a feature is being carried from (including) firmware version 1.1.2.

Always refer to the section 7 Firmware version history to check the list of different changes that have been made between firmware versions.

1.3.2 How to identify the firmware version number of an LED display

Use “Get System Information” in the Multimedia LED software to read information details of the LED display. The result will contain the firmware version number.

Remember to address the display exactly (by unit number) in order to be sure which unit responded to the command. (If the intended Communication ID is visible in the response text, this will also clarify.)

If an LED display can be programmed to display texts but the “Get System Information” command fails this means that the firmware version is below 1.0.0. (Only limited number of units were produced with such pre-release firmware.)

2 Theory of display operation

2.1 Terms

2.1.1 Line

Text limited by either a line break character (ASCII 127 or ASCII 13) or end of text.

Vertical size of line is given by the highest element of the line. (Text, graphics, etc.)

If a line cannot fit horizontally on the screen it will be scrolled from right to left. (Except when horizontal positioning is used in the line by using TAB (ASCII 9) character.)

2.1.2 Screen

One line or a set of lines that can vertically fit onto the display at once. This depends mostly by the number of vertical pixels (height) of the LED display, and the fonts used in the line text.

2.1.3 Program

A program consists of one or multiple lines.

A program consists of one or multiple screens.

Each program has properties that are used to select their behaviour:

- **Start date** (program runs from the selected date inclusively)
- **Start time** (program runs from the selected hour and minute inclusively)
- **End date** (program runs to the selected date inclusively)
- **End time** (program runs to the selected hour and minute inclusively)
- **Days of week mask** (program runs on the selected days only)
- **Repeats:** Number of program repeats before switching to the next valid program.
- **Effect:** Method of program screens appearing on the display. (**[Version < 5.0.0] limitation:** only “HOLD” effect is supported.) Screens with a scrolling line will not have any effect applied.
- **Speed** of the selected effect, and rolling speed for rolling lines.
- **Pause** in seconds before switching to the next screen. Screen scrolling lines will be shown at least until this period and at least until all scrolling lines have been shown, at least once completely. The screen will switch to the next message when it has scrolled out completely

- Default **Align** of lines: left, right or centre.

Maximum allowed program size is about[1] 4000 characters including control sequences. If that limit is exceeded, the program will be dropped, however the frame will be acknowledged.

2.1.4 Boxed mode

From **[version>=5.0.0]** a new display mode has been introduced. This mode allows the screen to be divided up into “boxes” where the following line data will be displayed.

Boxes having horizontal boundaries, a line will not necessarily take the whole width of the display.

In boxed mode every program has only one screen where every line of that program will be put into.

If any part of the line content is outside the box vertically, that part will not be displayed.

If a line won't fit to the box horizontally, the content will be scrolled from right to left inside the box. The only exception is lines with TAB positioning.

2.1.5 Base and expanded mode

In base mode the start and end date, time properties and days of the week mask are not checked, all downloaded programs are considered valid. In expanded mode these properties will be checked against current date and time.

2.1.6 Splash Screen

The splash screen is displayed immediately after the display gets powered. It is held for 5 seconds. It contains:

- Configuration ID (ii) as decimal number
- Firmware version (v.v.v)
- Configuration Word (wwww) in hexadecimal format
- Communication ID (cc) as decimal number
- (Forced) channel assignations for RS485 and Network (4N)
- RTC type ID and Flash memory status (rf)

Splash screen information display format: ii.v.v.v:rf:wwww:cc:4N

(Splash screen information is broken into two lines when the display is less than 80 pixel wide but its height is at least 15 pixels.)

Splash screen is always static. Larger screens may have space to contain additional custom text(s) below the information line. Such additional text option should be requested in an order.

The display of the Splash screen can be disabled by setting the corresponding bit in Configuration Word. Check section 5.4.2 'W' + 0x06 – Change Configuration Word for details.

1 Cannot determine exact number of characters. Every new line requires additional 8 bytes (only 4 bytes from V4.0.0) of storage, however this structure also stores the line starting colour and font selections therefore having those at the start of every line will not need additional storage space.

Splash screen functionality exists only in firmware **[Version >= V3.1.0]**.

2.1.7 Communication timeout screen

Communication timeout screen feature is used to display a screen that informs the viewer of the display that the screen had gone offline. This condition happens when:

- No proper frame received until the pre-set period elapsed since the last proper frame.
- Instantly after erroneous frame condition detected. (This condition can be enabled or disabled separately.)

Display returns to normal operation after a first proper frame of any of the followings:

- Write Program (text) ('A')
- Write Graphics ('E')
- Write Variable ('C') command.

This feature was available on request from **[Version >= V3.2.0].** From **[Version >= V4.0.0]** this is a standard feature that is switched off by default. As a standard feature it writes “Offline” on the top left side of the screen with 3x7 pixel font. Other text(s) or empty screen are available on request.

Check section 5.4.3 'W' + 0x07 – Change Additional Configuration for further details how to set up this function.

2.1.8 RAM vs. Flash mode

RAM mode is when a downloaded program (text), graphics and variables are only stored in volatile memory. All of these changes will be lost if the display restarts. (Loss of power/Soft Reset).

In Flash mode changes are written instantly to non volatile memory (flash) and will be retained until the next change regardless of power loss.

This is a very simple approach that commonly used in most commercial displays. The approach that is used in our displaya are more complex than this. See the corresponding details in section 2.2.3 How RAM and Flash mode works.

2.1.9 Initial stream

A method that allows to load default data (or settings):

- Every start up if flash memory is not populated on the mainboard,
- Once, when the flash memory gets initialised

Most important is the first one which allows for a RAM only configuration to have default data loaded that is available for immediate display after the power returns.

Initial stream is a custom option available on request.

What is also available on request is called “Conditional Initial Stream” or “Alternative Initial Stream”. With this option it is possible to have a secondary Initial Stream that will be applied instead of the primary stream if any of the following conditions met:

- Pins 1-2 of DIGTEMP connector has been shorted (jumpered). (This condition is optional

and only available on request.)

- UserConfig0 bit has been set in Configuration Word. (Available if UserConfig0 bit is not used to enable some other function.)

2.2 Operation

LED display will show programs that previously has been downloaded onto it. After finishing a program completely (all repeats) the display will switch to the next valid program. After the last program displayed the first valid will be shown.

Normal display mode:

Before firmware version 5.0.0 only the following normal displaying modes of lines were available.

Programs are shown as one or several screens depending on the number of lines and their heights. If the remaining pixel rows are enough for the height of the next line, that line will be put on that screen too. It goes on until the following line will not fit onto the remaining part of the screen. That line will be the first line of the next screen.

- **[Version >= V1.1.2]** additional feature: When the height of the last line of the screen is (almost) half of the height of the display, the line will be displayed one pixel row lower – it will just touch the bottom of the display. **[version >= 5.0.0]** this function can be switched off by setting the corresponding bit in Legacy Flags. (See in section 5.4.3.5.)

A line that horizontally doesn't fit onto the screen will be scrolled from right to left.

Boxed display mode:

Boxed mode is an additional feature available with **[Version >= V5.0.0].**

In this mode all display lines of the program will be put onto one screen. Line data will be shown in the defined “boxes” on the display. Box coordinates have to be embed in program text at the start of every line.

Scrolling will be governed by box width instead of screen with.

Default configurations will handle up to 32 boxes for a screen. If there are more lines/boxes than the limit in the firmware, the remaining lines/boxes will not be displayed. Use the Read Display Dimensions ('R' + 0x05 + 'D') command to determine the maximum number of supported boxes.

Data refreshing on a drawn screen:

- **[Version < 5.0.0]:** Displayed data is not refreshed unscrolling lines. When variable or graphics changes, the current screen will be invalidated.
- **[Version >= 5.0.0]:** Displayed data refreshed on non scrolling lines when necessary. Can switch back and forth between scrolling and non scrolling. In boxed mode no change could trigger an early finish of the current screen. In normal mode the only reason to do that is a change in screen height.

2.2.1 TAB positioning

A line that contains TAB character (ASCII 9) is special in two ways:

- It will never scroll. Even if it doesn't fit on the screen it will be displayed as steady line.
- Always will be left aligned

Three character digits after the tab character gives the starting horizontal pixel position for the rest of the line text.

Whenever a character of the line cannot be printed completely because it (partially) falls off at the right side of the display, the rest of the line will not be printed. Any TAB positioning after that character will not be evaluated regardless if the new position will be on the screen or not.

From **[Version >= 5.0.0]** TAB character can lead other control parameters too. Using these will not have the above mentioned effects! However it will remain true for all TAB formats that the next three characters after a TAB character is part of a control and will not be displayed.

2.2.2 Base vs. expanded mode

In **base mode** start,end date,time properties and days of week mask are not checked, all downloaded programs are always considered valid. An LED display controller without an RTC unit will run in normal mode by default.

In **expanded mode** the above mentioned properties checked against the actual date and time kept in the unit. Validity of the program is given by three constraint groups of properties. These three groups are independent. Any of these groups can make the program currently invalid, and the program will not play. These constraint groups are:

1. Start date and end date. Evaluation is valid when the current date is between start date and end date inclusively.
2. Start time and end time. Evaluation is valid when current time is between start time and end time inclusively. When the start time is bigger than end time it selects the period that laps over midnight.
3. Days of week mask. If current day-of-week is marked on the days of week mask, the evaluation of this property is valid.

[Version >= 1.1.6] additional features:

In expanded mode, when selecting no days on the days of week mask will combine date and time groups to one group and days of week mask evaluation will always bring “valid” result. (0x00 will be converted to 0xFF to switch on date+time combined constraint mode.)

When bit 7 of days of week mask is set, days of week mask will also be evaluated. However bit 7 cannot be reached from the Multimedia LED software – requires custom application.

Switching LED display between base and expanded mode is supported.

[Version 1.0.0] limitation:

This version always runs in base mode and cannot run in expanded mode.

[Version >= 2.2.0] additional feature:

From firmware version V2.2.0 switching operation mode between base and expanded mode is stored to the flash configuration area. In this operation mode it is not forgotton by the display when powered off and so default operation mode is not restored when the display power on.

2.2.3 How RAM and Flash mode works

Mainboards without Flash chip (IC5) installed will always operate in Classic RAM mode.

Flash mode supported only from **[Version >= 4.0.0]**.

To switch between RAM and Flash mode use Change Configuration Word command. See section 5.4.2 'W' + 0x06 – Change Configuration Word for more details.

Default mode is:

- RAM for mainboards without a flash chip
- Flash for mainboards with a flash chip

2.2.3.1 Classic RAM mode

Without the flash chip being populated on the mainboard, Flash mode is not available at all. The display will always be in RAM mode.

Since there is no flash storage available, the display will have no content that can be displayed, so it will be empty.

The only way to provide default data displayed after powering up the unit is to use the Initial Stream option that makes possible to load static data at every startup. However this static data cannot be changed, it has been burnt into the firmware.

RAM capacity limits the accepted amount of data. (Use “Ra” – Get Program Text Information, “Rc” – Read Variable or “Re” – Get Graphics Information commands to check if a change has been accepted. Alternatively in this mode the use 'R' + 0x05 + 'R' – Read free RAM amount command can also provide predictable result.)

2.2.3.2 Advanced RAM mode

In the firmware design, a more advanced RAM mode is available when flash chip has been populated on the mainboard. In this case all data that has been stored in the display before switching to RAM mode will be available in RAM mode after every startup. In this way a default state of data can be stored. The use of these can be wide, just a few examples:

- Have default text program(s) to display
- Have some default variable values that are available for use
- Have a few graphics stored that can be used any time without the need to send them to the display every time – saving bandwidth.

So to put it simply reading of flash memory is still available in advanced RAM mode, and only write (including delete) will be blocked. Of course any data defaults from flash can be overwritten in RAM.

Of course RAM capacity still limits the accepted amount of changes. Use “Ra” – Get Program Text Information, “Rc” – Read Variable or “Re” – Get Graphics Information commands to check if a change has been accepted.

It can be a possible scenario with lots of data, that although all changes have been accepted, some defaults stored in flash cannot be loaded because not enough RAM space remained. Environments

where total loaded content (including flash stored content) is close to (more than half of) the available RAM amount should be assessed for operability. When the assessed result is not convincing, flash stored defaults should not be used to avoid having missing details; All stored content from flash memory should be deleted before switching to RAM mode.

It is also necessary to delete all unused default variable values before switching to RAM mode. The reason is that all variables are always loaded into RAM space, even if they are not being used at all.

Switching back to Flash mode:

By switching back to Flash mode from Advanced RAM mode, flash write operations will be enabled instantly, and all changes made in RAM mode will be synchronized to the Flash memory. This way the defaults stored for RAM mode will be overwritten with all the new data. To retain flash memory content, execute “WB” – Soft Reset command before switching to Flash mode.

2.2.3.3 Flash mode

In flash mode all data changes are almost immediately synchronized to the Flash memory. Content will be available at its last stored state after restart of the display.

Warning! Never use flash mode when any content requires frequent changes. Depending on the change frequency this will eventually wear the flash chip and that can cause content write errors, rejected content loads and unwanted display pauses (non-responsiveness); Reaching that stage the Mainboard has to be changed on the cost of the user.

In flash mode more content can be stored than would fit into the RAM at once. For best results the order to perform such high content amount is:

- Delete all active programs. (Time constrained programs will not use RAM outside the selected active period.) This will allow that only variables taking away space from RAM during the update.
- Change Graphics as necessary. Check if a change have been accepted by using “Re” – Get Graphics Information command.
- Change Variables as necessary. “Rc” – Read Variable command can be used to check if the proper value has been stored.
- Update/reload active programs. Check if a change has been accepted by using “Ra” – Get Program Text Information command. The order of loading active programs is a strategic decision. This depends on the content amount that each program will load into RAM. A method to avoid problems during this load step could be doing the following sub-steps:
    - Send first a dummy active program (no constraints) with only a space as content and with long hold time (and with many repeats) as necessary to cover the whole update time surely. Use the program location of the active program with the most content. The display will play this dummy program for a while because there are no active programs before. That playback should cover the time of all transmission (including checks and retries).
    - Send all active programs having a different program location.
    - Update the program location used in the first sub-step with the relevant program data.

3 Brief hardware introduction of the main control board of the LED display

3.1 Communication interfaces

The LED display controller is capable of communicating over RS232, RS485, RS422 or Network (TCP/IP). RS232 and RS422 having two independent ports. This brings the possibility of six physical ports. (Availability of these physical interfaces depend on order.)

The CPU of the controller has only two communication channels. (Identified as Ch1 and Ch2 on the PCB of the controller.) The two channels are identical, either of them can be a connection towards the host or slave display(s).

From the above six physical ports, only two can be used for communication. Since the firmware has a gateway function between the channels, it allows it to use the point-to-point interfaces (RS232 and RS422) to be used as a bus by linking them up by using the other channel.

Of course it is possible to use this two channel on different physical interface type. One of the many practical uses of this could be to use one channel for WAN access via TCP/IP and use the other channel to link up the other LED displays nearby via some relatively cheaper interface like RS422 or RS232.

Assigning a Channel to a physical interface is made with DIP switch block “COMM_CONFIG”. (For fixed/hard wired physical configuration orders this switch is not soldered onto the PCB.)

Network connection can also be TTL connection, GSM/GPRS, WIFI, ZigBee or Bluetooth module not just TCP/IP module.

Some controller variants like the display PCB integrated variants may have all physical connections enabled. In this case the installer should avoid installing more than one physical connection of the same channel at once[2].

3.1.1 RS232

There are two physical connectors supporting RS232. These are marked as RS232A and RS232B.

- RS232A can only be connected to logical channel Ch1.
- RS232B can only be connected to logical channel Ch2.

Connectors are shared with RS485.

Warning! Special cabling is required between two LED displays that are connected via RS232

3.1.2 RS485

RS485 (bus) can be connected on the same two connectors the RS232 is getting provided. The RS485 bus signals of the two connectors are internally connected in parallel on the main board PCB. Therefore no Y splitter necessary to build up an RS485 bus.

2 If collision and data flow violation avoided by other means the display hardware can handle time-shared communication on the same channel.

RS485 bus terminator is provided on the controller main board. Two jumpers must be installed in position to install the terminator on the bus. Both ends of the RS485 bus must be terminated in order to limit signal interference. Do not install terminator jumpers of displays that are on the middle of the bus!

Straight RJ12 ended cables are required to build an RS485 bus.

RS485 is not recommended especially when communication checksum of the protocol is not used. The reasons are:

- Half duplex (multi-source) use of the cable requires that the bus should float during idle periods and between command and response for short periods. That makes it more susceptible to electromagnetic disturbance.
- Gateway function of the controller allows bus-like (active chain) configuration without floating signal lines. Using chained RS422 interfaces even allows higher bus lengths than it is possible with RS485. Better signal quality maintained because of no intermediate drops and no floating phase.

3.1.3 RS422

There are two physical connectors supporting RS422. These are marked as RS422A and RS422B.

- RS422A can only be connected to logical channel Ch2.
- RS422B can only be connected to logical channel Ch1.

Crossed RJ12 ended cables are required to connect two LED displays with RS422.

3.1.4 TCP/IP

TCP/IP connection is supported via Lantronix XPort module. Refer to XPort module manual for TCP/IP settings.

3.1.5 TTL

TTL connection is also available on order. TTL input is 5V tolerant, output high level is 3.3V only.

3.2 LED display address selection

LED display address can be selected by using DIP switch block “COMM_ID”. (May not be populated for orders with fixed address units.)

[Version < 2.2.0] Limitation: Software-based selection of display address is not supported.

[Version >= 2.2.0]: Software-based display address becomes effective when hardware based address on “COMM_ID” DIP switch block is set to zero (all switches off.)

3.3 Watchdog

CPU/Firmware having a watchdog set in order to limit the effect of:

- power fluctuations

- interface shocks
- oscillator problems
- electromagnetic radiations
- coarse firmware errors

Watchdog activates approximately 65[3] seconds after the firmware becomes non-responsive. After that the mainboard restarts and all content stored in RAM only will be lost.

3.4 Other features *(These features are also order dependent.)*

- Analogue temperature sensor,
- Ambient light sensor for automatic brightness control,
- Real-time clock (RTC) for keeping date and time,
- Flash memory to store programs (text), graphics and variables.

3 Previous versions (<V2.2.3) of this documentation contained incorrectly 8-15 seconds of watchdog reset time.

4 Low level communication protocol details

[Version <= 1.3.0]: Serial communication baud rate is 9600 bps, with 8 data bits, no parity and 1 or 2 stop bits. The LED display controller can receive commands with only one stop bit, and sending responses with two stop bits.

[1.1.8 <= Version <= 1.2.0] warning: Communication setup was 9600,N,8,2 **– two stop bits!**

[Version <= 1.1.7] warning: Communication setup was 9600,N,8,1 **– only one stop bit!**

Communication method is half duplex. That's true even when using a full duplex physical interface such as RS232 or RS422. Response must be fully waited before any new frame can be sent.

4.1 Frames

4.1.1 Long frame

Long frame is getting transmitted when:

1. Computer (Master device) sends command to a display
2. Display responds with data – was requested in the preceding command

Overall frame format:

Offset:	0-4	5	6-9	10	11-	n	n+1 - n+4	n+5
Length:	5	1	4	1	variable	1	4	1
Function:	preamble		addresses	mark	command	mark	checksum	postamble
ASCII:	NUL	SOH		STX		ETX		EOT
decimal:	0	1		2		3		4

As is clearly visible, the frame format has some fixed parts that are delimited with special characters. These special characters (NUL, SOH, STX, ETX and EOT) are not allowed to appear in the contents of the variable fields (address, command and checksum.) Using these characters in the variable fields is a clear violation of low level transmission, and therefore the frame will be dropped. Such frame will be never respond - neither by short nor long frame.

4.1.1.1 Addresses field

Addresses field has two parts:

- Originator address
- Target address

Both addresses are have two hexadecimal digits. However to address one group of displays, it is allowed to use '?' character as second digit but only in this particular case.
Either the originator or the target must be the master (computer) – with address of “FF”.

Description	ASCII	Hexadecimal	Decimal
Master (computer)	'F','F'	46, 46	70, 70

Description	ASCII	Hexadecimal	Decimal
Broadcast (to all displays)	'0','0'	30, 30	48, 48
Group (broadcast to a display group)	<group>+'?'	<group>, 3F	<group>, 63
Single display	Any other two hexadecimal digits (with two characters from '0'-'9' or 'A'-'F' used)		

Addresses of groups and all display broadcast values are only allowed in the target field, when the master (computer) is the originator.

Since “00” and “FF” addresses having fixed function, only the remaining 254 addresses can be used as display addresses. (“01” to “FE”)

Expected responder:

Target type	Target address	Responder address
Broadcast	“00”	“01” (broadcast responder)
Group '0' broadcast	“0?”	“01” (group '0' responder)
All other <group> broadcast	<group>+”?”	<group>+'0' (group responder)
Single display	Example: “5A”	“5A” (natural responder)

As seen in the above table, some display addresses are special, because displays with these addresses are entitled responding to group and all display broadcasts.

If an application plans to use group messages or broadcasts, it is required to install:

- Broadcast responder (“01”) for broadcast to all displays,
- Group responder of the targeted group,

otherwise no response will arrive.

4.1.1.2 Command field

Command field contains the actual command and data getting transmitted from master (computer) to one or many displays.

In case of long response from a display, the command field contains the data requested by the master (computer) in the preceding (query) command.

In both cases this field is considered as the high level part of the communication.

For details refer to chapter 5 Commands.

4.1.1.3 Checksum field

Checksum field is always 4 characters of hexadecimal digits – '0'-'9' or 'A'-'F'.

[Version >= 2.2.0] feature: The look for the checksum field can be switched off by setting the configuration word accordingly.

[Version < 2.2.0] limitation: Upon special request we can provide firmware version that will not look at the checksum field.

When the checksum field is inactive it means it accepts all hexadecimal values. The application still

has to maintain that the four characters of the field must be hexadecimal digits, however this is not recommended due to loosing some communication robustness. Especially never recommended on RS485 or when longer cable lengths are getting used, so when the hardware level reliability has already been reduced.

To maintain compatibility with the Multimedia LED software the checksum computation method depends to the command. Although this is rather unfortunate, only three calculation methods exist. Check for command details which method is used for that particular command.

1. **Additive checksum (shortly ADD):** Used by many commands and long responses.

   Initial checksum seed is zero. Method of calculation is to add all values of the frame from STX to ETX inclusively. This method doesn't include the address field (and SOH, before.)

2. **XOR checksum (shortly XOR):** Only “write program (text)” command using this method.

   Initial checksum seed is zero. Method of calculation is to apply XOR function between the current checksum value and the frame bytes. Frame bytes from SOH to ETX (inclusively) should be used in this checksum calculation. Since bytes XOR-ed together, the first two characters of the checksum field will always be '0'. Simply speaking all bytes of the frame before the checksum field should be XOR-ed to get the checksum value. (NULs are not affecting the checksum value, so it's enough to start from SOH.)

3. **No calculation (shortly FFFF):** Most query type commands are using this. Checksum field should be filled with four 'F' characters.

Here must be highlighted again the important difference between ADD and XOR checksum is that only the XOR checksum calculation should run on the bytes of the address field and SOH!

[Version >= 3.0.0] feature: All commands can accept checksums calculated by any of the above three checksum calculation methods.

4.1.2 Short frame

Short frame is transmitted as a simple acknowledge to a command. However when the command requests display response in long frame, only the first character of this (acknowledge) frame will be sent.

Short frame is always two bytes:

Offset:	0	1
ASCII:	EOT	SOH
decimal:	4	1

Meaning of these two bytes:

- EOT: Frame receive acknowledge
- SOH: Frame processed successfully

It is allowed to have a time gap between these two bytes.

For compatibility reasons successfully receiving a short frame as response to any command doesn't mean that the command is accepted and executed. It only means that the target unit is reachable on the bus and the bus is free now for the next command to be sent.

5 Commands

This chapter describes the commands - the high level protocol details.

The first character of the command field is the command identifier. The following character is often an index or a subcommand code.

5.1 'A' - Write program (text)

Command 'A' uses XOR checksum calculation method.

This command is used to put anything (text, graphics, etc.) on the display.

5.1.1 Command field structure:

command	index	program properties	program text
'A'	'0'-'9', 'A'-'Z'	in either V2006, V2007 or short format	displayed content & controls

Program properties part has three forms. Referred as V2006 and V2007 in relation to the Multimedia LED software year produced. Short format requires custom program (has no commercial program support).

V2007 is just an extension to V2006 format with two additional (start and end show date) fields.

Blackspaces in the marker field is crucial to identify the used program property version.

Short format is marked by an 'N' character at the first character of days of week mask (at offset 3) indicating no program constraints, and therefore that part (and the 0xFF markers) are skipped completely.

[version < 1.4.0] limitation: Short format is not available.

5.1.1.1 Program properties - V2006 format:

Property	Offset	Size	Remarks
effect	0	1	**[Version < 5.0.0]:** 'A'-'Z' allowed, but always “Hold” effect is getting used. **[Version >= 5.0.0]:** Any character allowed, check 5.1.3.10 section Effects for details.
effect/roll speed	1	1	'0'-'9' allowed. '0'=fast, '2'=normal, '4'=slow ('9'=very slow)
hold time	2	1	'0'-'9' allowed. Screen hold time in seconds. **[version >= 2.0.1]** allows to use characters 0x80-0xFF.
days of week mask	3	2	Two hexadecimal digits. Days of week (mask) constraint.
start time	5	4	'0'-'9' allowed in HHMM pattern. Start time constraint.
end time	9	4	'0'-'9' allowed in HHMM pattern. End time constraint.

Property	Offset	Size	Remarks
marker	13	2	Two blackspace characters. (Hex:FF, FF; Dec:255, 255)
repeats	15	1	'1'-'9' allowed: program repeats. Hex: FF translated to '1'. **[version >= 1.4.0]** allows to use 'A'-'Z' as 10-35 repeats.
align	16	1	'1':left, '2':right and '3':center default alignment of lines. **[Version >= 5.0.0]** this character also used to switch to boxed mode. Boxed mode variants of the default line alignments: '<':left, '>':right, '=':center.

5.1.1.2 Program properties - V2007 format:

Property	Offset	Size	Remarks
effect	0	1	**[Version < 5.0.0]:** 'A'-'Z' allowed, but always “Hold” effect is getting used. **[Version >= 5.0.0]:** Any character allowed, check 5.1.3.10 section Effects for details.
effect/roll speed	1	1	'0'-'9' allowed. '0'=fast, '2'=normal, '4'=slow ('9'=very slow)
hold time	2	1	'0'-'9' allowed. Screen hold time in seconds. **[version >= 2.0.1]** allows to use characters 0x80-0xFF.
days of week mask	3	2	Two hexadecimal digits. Days of week (mask) constraint.
start date	5	8	'0'-'9' allowed as YYYYMMDD. Start date constraint.
end date	13	8	'0'-'9' allowed as YYYYMMDD. End date constraint.
start time	21	4	'0'-'9' allowed in HHMM pattern. Start time constraint.
end time	25	4	'0'-'9' allowed in HHMM pattern. End time constraint.
marker	29	2	Two blackspace characters. (Hex:FF, FF; Dec:255, 255)
repeats	31	1	'1'-'9' allowed: program repeats. Hex: FF translated to '1'. **[version >= 1.4.0]** allows to use 'A'-'Z' as 10-35 repeats.
align	32	1	'1':left, '2':right and '3':center default alignment of lines. **[Version >= 5.0.0]** this character also used to switch to boxed mode. Boxed mode variants of the default line alignments: '<':left, '>':right, '=':center.

5.1.1.3 Program properties - short format [version >= 1.4.0]:

Property	Offset	Size	Remarks
effect	0	1	**[Version < 5.0.0]:** 'A'-'Z' allowed, but always “Hold” effect is getting used. **[Version >= 5.0.0]:** Any character allowed, check 5.1.3.10 section Effects for details.

Property	Offset	Size	Remarks
effect/roll speed	1	1	'0'-'9' allowed. '0'=fast, '2'=normal, '4'=slow ('9'=very slow)
hold time	2	1	'0'-'9' allowed. Screen hold time in seconds. **[version >= 2.0.1]** allows to use characters 0x80-0xFF.
marker	3	1	'N' indicating short format with no program constraints
repeats	4	1	'1'-'9' and 'A'-'Z' allowed: program repeats. ('A'-'Z'=10-35) Hex: FF translated to '1'.
align	5	1	'1':left, '2':right and '3':center default alignment of lines. **[Version >= 5.0.0]** this character also used to switch to boxed mode. Boxed mode variants of the default line alignments: '<':left, '>':right, '=':center.

[version >= 2.0.1] addition: allows to use characters 0x80-0xFF as hold time character. Hold time can be calculated as:

- 0x80 – 0xBF: Hold time=50ms*(hold_character-0x80).
- 0xC0 – 0xFF: Hold time=500ms*(hold_character-0xC0).

(0ms hold time is not 0ms, but the shortest possible time, that is given by the screen refresh speed. It also depends on (cannot be shorter than) the time by the firmware delivers the next screen.)

5.1.2 Design considerations/limits:

- In total 36 programs can be set up at once on an LED display. There are indexed as '0'-'9' then 'A'-'Z' on the second character of the command field.
- No program can be longer than 4000 bytes. This is the command field size limit, not the number of displayed characters. It includes the control character combinations and program properties too.
- In flashless (RAM) operation mode the amount of available RAM will allow about 8-12[4] kbytes to store all programs, graphics and variables. (From **[Version >= V4.0.1]** 'R' + 0x05 + 'R' – Read free RAM amount command can be used.) Whenever the RAM has no room to store another program it will be dropped, however the command will be acknowledged. Especially in case of RAM operating mode, the application developer should ensure that this would never occur.

5.1.3 Program text field

This field contains the actual text that is getting displayed with formatting.

May contain special texts or other objects like time, temperature, variables, graphics etc.

Formatting, special text and objects are accessed via reserved characters.

[version <= 1.0.0] limitation: Only reserved characters of (decimal) 249-254 can be used.

4 The amount of available RAM depends on the display configuration. Rule of thumb is that displays with the most pixels, colours and colour depth having less available RAM to store anything else, like the programs.

[version >= 1.1.0] recommendation: Both set of reserved characters can be used, but it's recommended to use the lower set of 0x06-0x10.

[version >= 5.0.0] additional feature: High set of control characters (249-254) and new line character 0x7F can be disabled to use them to accessing regular characters from a font. See section 5.4.3 for 'W' + 0x07 – Change Additional Configuration command details.

[version >= 5.0.0] additions: Text field has to contain box coordinates that must be located at the beginning of every line. This means that atext field is further divided to sections in boxed mode like in the following example of 3 boxes:

Coordinates	Line text	New Line character	Coordinates	Line text	New Line character	Coordinates	Line text

5.1.3.1 Formats of Box Coordinates

Box coordinates can be given in several different formats. They can provide identical results, it's up to the user to decide which one fits best for the current box.

- Top pixel row is having vertical index of zero. If a display having Y*X dimensions, the bottom row is has vertical index of Y-1.
- The leftmost pixel column is having index of zero. If a display having Y*X of dimensions, the bottom row is has vertical index of X-1.

The box defining row and column numbers are always part of the box. Any coordinates offsetting out of the display will be limited to be on the display area – at the display edge in which direction the over-offsetting happens.

Boxes are allowed to overlap each other. However care must be taken to check whether the desired result has been achieved, because the firmware doesn't take any special measures to deal with this. Boxes are drawn in the order they and are in the program text.

The first character in Box Coordinates selects whichever main format is being used for the whole field:

- '!' - Limited entry mode - **[version >= 5.0.1]** required to use it
- '#' - Division based limited entry mode
- >='@' - Binary mode (this first character is also used as coordinate data)
- others – Comma separated mode

Box coordinate parts are come in the following order: x of top left corner, y of top left corner, x of bottom right corner and y of bottom right corner.

- In both of the limited entry modes all coordinate parts has only one byte. (That makes in total 5 bytes including the mode selection character.)
- In binary mode all coordinate parts having two bytes. (In total: 8 bytes.)
- In comma separated mode the field have variable lengths. It requires commas (',') (or any unused characters) to be put after every coordinate part.

5.1.3.1.1 Limited entry mode – **[version >= 5.0.1]** required to use it.

In limited entry mode every character fully defines a coordinate part. It might not be suitable for large displays because some parts of the display may be out of reach with the following offsets. Character value range is 5 to 255. This is divided into two sections:

- 5-125: relative 0..120 pixels from left or top edge of the display
- 126-255: relative 119..0 pixels back from right or bottom edge of the display

Example: '!' + 0x05 + 0x05 + 0x0B + 0xEF; Box starts at the top-left corner of the display. Bottom right end is 6 rows below top row of the display and 16 columns back from the rightmost column of the display.

5 .1.3.1.2 Division based limited entry mode
In division based limited entry mode every character is fully define a coordinate part.

Character value range is 5 to 255. This will be transformed to 0-100% of the display height or width - depending on which coordinate part is getting defined by the value.

Example: '#' + 0x05 + 0x05 + 0x7D + 0xFF; Box takes the left half of the display – may include the column right in the middle of the display.

5.1.3.1.3 Binary mode
In binary mode two characters define a coordinate part.

Both characters' value range must be between 64 and 191 inclusively. Value form this range will be transformed to 0-127 by subtracting 64. The resulting 7 bit numbers will be used by concatenating them to a 14 bit binary number.

- Having bit #13 set selects division based relative mode. Value on bit #12..#0 will be transformed to 0-100% of the display height or width - depending on which coordinate part is getting defined by the value.
- When bit #13 is clear, that selects offsetting mode.
  - When bit #12 is clear then the offset is calculated from the left or the top of the display (positive offsetting). Bit #11..#0 is the offset.
  - When bit #12 is set then the offset is calculated back from the right or the bottom of the display (negative offsetting). Bit #11..#0 is the offset.

Example: 0x40 + 0x40 + 0x40 + 0x40 + 0xA0 + 0x40 + 0x60 + 0x48;

- Box left edge: Positive offset 0 = Leftmost column of the display.
- Box top edge: Positive offset 0 = Top row of the display.
- Box right edge: Division based selection of 50% of the display width = about the middle column of the display.
- Box bottom edge: Negative offset of 8 = 8 row above the bottom row of the display. (Display must have more than 8 rows not to cause over-offsetting condition.)

5.1.3.1.4 Comma separated mode
This mode is more complex. Coordinate parts are divided with characters that have no other meaning, like the comma ',' character.

First character can be special mode selection character for this mode:

- '%' selects division based mode
- '-' selects negative/reversed direction offsetting mode

For the rest of the characters the field should contain characters of decimal digits that carries the value.

Like for the other modes negative offsetting means that the origin of the offset is not the top or left edge, but the right or bottom edge of the display, and the offset is counted backwards.

Division based mode here means defining in:

- 1/9 parts if 1 decimal digit follows the '%' sign,
- 1/99 parts if 2 decimal digit follows the '%' sign,
- 1/999 parts if 3 decimal digit follows the '%' sign.

If no data is given for a coordinate, the corresponding edge of the display will be used. (',' + ',' + ',' + ',' is a valid box definition example that will use the whole screen area.

Example: “,%0,-16,%49,”

- Left edge is default: = left edge of the display,
- Top edge is at 0/9 of the height: = top edge of the display. (“%0,” here is practically useless, the ',' comma alone would have provided same result. The same example in a more space efficient form would be: “,,-16,%49,”)
- Right edge is 16 pixel columns back from display's rightmost column.
- Bottom edge is at 49/99 of the height (from the top).

5.1.3.2 Font select and font modifier

[version >=1.5.6] additional feature: Font modifiers. By its second character either of font select or font modifier can be encoded the same way. In case of font modifier the font selection is kept however the behavior of its printing properties can be modified. See this described later in depth.

Always having the length of 2 bytes. First character is the reserved character, then the font ID character.

Reserved character code:	**Decimal**	**Hexadecimal**
Lower (new) set	15	0F
Higher (old) set	254	FE

Default fonts assigned to font ID characters:

ID	**Name**	**Height**	**Usual width + (spacing)**	**Description**
'A'	SS5	5	4 (0-1)	Smallest (5 pixels) height font.
'B'	ST5	5	5 (0-1)	SS5 with bold character effect.
'C'	WD5	5	8 (0-1)	SS5 with double-strike effect.
'D'	WS5	5	9 (0-1)	SS5 with both bold and double-strike effect.

ID	Name	Height	Usual width + (spacing)	Description
'E'	SS7	7	5 (0-1)	Normal face 7 pixels height font.
'F'	ST7	7	6 (0-1)	SS7 with bold character effect.
'G'	WD7	7	10 (0-1)	SS7 with double-strike effect.
'H'	WS7	7	11 (0-1)	SS7 with both bold and double-strike effect.
'I'	SDS	7	6 (0-1)	SS7 with color shadow effect. (Colored bold effect.)
'J'	SSF	7	8 (0-1)	Fancy face 7 pixels height font.
'K'	STF	7	9 (0-1)	SSF with bold character effect.
'L'	WDF	7	16 (0-1)	SSF with double-strike effect.
'M'	WSF	7	17 (0-1)	SSF with both bold and double-strike effect.
'N'	SDF	7	9 (0-1)	SSF with color shadow effect. (Colored bold effect.)
'O'	SS10	10	7 (0-1)	Normal face 10 pixels height font.
'P'	ST10	10	8 (0-1)	SS10 with bold character effect.
'Q'	WD10	10	14 (0-1)	SS10 with double-strike effect.
'R'	WS10	10	15 (0-1)	SS10 with both bold and double-strike effect.
'S'	SS15	15	7 (0-1)	Normal face 15 pixels height font.
'T'	ST15	15	8 (0-1)	SS15 with bold character effect.
'U'	WD15	15	14 (0-1)	SS15 with double-strike effect.
'V'	WS15	15	15 (0-1)	SS15 with both bold and double-strike effect.
'W'	SS23	23	8 (0-1)	Normal face 23 pixels height font.
'X'	SS31	31	12 (1-2)	Normal face 31 pixels height font.
'@'	Small	7	3 (0-1)	Small face 7 pixels height font.

If a font from the above table cannot fit onto the display (height is bigger than display height) a smaller normal face font will be used automatically.

Usual width is just for information. All fonts are proportional. Spacing is getting decided individually by which characters are adjacent to each other.

- If non-colliding characters are adjacent to each other (“L7”, “Ta”, etc.): in this case the minimum spacing will be used – the first number in parenthesis of spacing.
- If colliding characters are adjacent to each other (“88”,”T7”, etc.): in this case the maximum spacing will be used – the second number in parenthesis of spacing.
- When text and graphics are adjacent to each other: it is always treated like colliding characters and maximum spacing will be used between them – the second number in parenthesis of spacing.

These are the default fonts. On request other fonts or custom fonts are available. These may or may not replace the default font set. Fixed formatting of the fonts is also available when needed.

If a font ID is not assigned to any font, SS7 font (ID: 'E') will be used instead.

[Version >= 1.1.2] font height improvements: If a font would be one pixel less in height than the display height, a higher font will be available to use all display pixel rows:

Display height	Fonts	Original height	Improved height
8	SS7, SSF, Small	7	8
16	SS15	15	16
24	SS23	23	24
32	SS31	31	32

[version >=1.5.6] additional feature: Font modifiers:

Currently there are six group of font modifiers:

1. Font spacing modifiers **[only available from version >= 1.5.7]**
2. Font print method modifiers **[available from version >= 1.5.6]**
3. Print clearing mode **[available from version >= 3.3.0]**
4. Increase vertical line drop **[available from version >= 3.3.0]**
5. Increase line spacing below **[available from version >= 4.0.5]**
6. Set horizontal alignment for current line **[available from version >= 4.0.5]**

Font modifiers are only valid for all the characters behind in same line. If the line contains another font selection later, all previous font modifiers are not effective to the text previous. To put it in another way modifiers are restored to their default values after every font select or line change.

Font modifiers:

ID	ID (dec)	ID (hex)	Modifier function
0x1C	28	0x1C	Set current line alignment to left
0x1D	29	0x1D	Set current line alignment to right
0x1E	30	0x1E	Set current line alignment to center
0x1F	31	0x1F	Increase row spacing value by one
' '-'"	32-39	0x20-0x27	Set minimum spacing from 0 to 7 respectively.
'('-'/'	40-47	0x28-0x2F	Set maximum spacing from 0 to 7 respectively.
'0'	48	0x30	Bold printing off
'1'	49	0x31	Bold printing on (not colour shaded)
'2'	50	0x32	Colour shading on (special bold print method)
'3'	51	0x33	Toggle bold printing (no colour shading)
'4'	52	0x34	Double-strike (wide) printing off
'5'	53	0x35	Double-strike (wide) printing on
'6'	54	0x36	Toggle double-strike printing

ID	ID (dec)	ID (hex)	Modifier function
'7'	55	0x37	Fixed width printing off
'8'	56	0x38	Fixed width printing on
'9'	57	0x39	Toggle fixed width printing
':'	58	0x3A	Set print clear mode to: no clear (painted pixel data OR-ed with previous content)
';'	59	0x3B	Set print clear mode to: transparent (clear for painted pixels only)
'<'	60	0x3C	Set print clear mode to: clear everywhere
'='	61	0x3D	Increase vertical drop by 1 pixel
'>'	62	0x3E	Increase vertical drop by 2 pixel
'?'	63	0x3F	Increase vertical drop by 3 pixel

5.1.3.3 Color select

Always having the length of 2 bytes. First character is the reserved character, then the colour ID character.

Reserved character code:	Decimal	Hexadecimal
Lower (new) set	16	10
Higher (old) set	253	FD

Colors assigned to color ID characters:

In the following table the colors are described as if the color layers would be Red, Green and Blue.

ID	Color	4-4-4 bit R,G,B value (Hex)	Description
'A'	Auto	F,F,F	Implemented as white. Warning! “Auto” color hasn't implemented as random changing color!
'B'	Light red	F,0,0	Full brightness on the primary color layer (can be any other color than red)
'C'	Light green	0,F,0	Full brightness on the secondary color layer – if exists (can be any other color than green)
'D'	Dark red	7,0,0	Same as 'B' if color depth is 1 bit.
'E'	Dark green	0,7,0	Same as 'C' if color depth is 1 bit.
'F'	Yellow	F,F,0	'B'+'C'
'G'	Brown	F,7,0	'B'+'E'
'H'	Amber	7,F,0	'D'+'C'
'I'	Orange	7,7,0	'D'+'E'
'J'	RGGYYY	N/A	Horizontal rainbow color effect, repeated by 6 rows.

ID	Color	4-4-4 bit R,G,B value (Hex)	Description
'K'	RGY	N/A	Horizontal rainbow color effect, repeated by 3 rows.
'L'	RG	N/A	Vertical rainbow color effect, repeated by 2 rows.
'M'	Light blue	0,0,F	Full brightness on the tertiary color layer – if exists (can be any other color than blue) On a 1 or 2 color displays it's black.
'a'	Dark blue	0,0,7	Same as 'M' if color depth is 1 bit.
'b'	Magenta	F,0,F	'B'+'M'
'c'	Purple	F,0,7	SS10 with bold character effect.
'd'	Plum	7,0,F	SS10 with double-strike effect.
'e'	Dark magenta	7,0,7	SS10 with both bold and double-strike effect.
'f'	Cyan	0,F,F	'C'+'M'
'g'	Turquoise	0,F,7	Greenish cyan
'h'		0,7,F	Blueish cyan
'i'	Dark cyan	0,7,7	
'j'	White	F,F,F	'B'+'C'+'M' – All three color layers at full brightness
'k'		F,F,7	Pale yellow
'l'		F,7,F	Pale magenta
'm'	Rose	F,7,7	Pale red
'n'		7,F,F	Pale cyan
'o'		7,F,7	Pale green
'p'		7,7,F	Pale blue
'q'	Grey	7,7,7	
r	Hor RGB	N/A	Horizontal rainbow color effect, repeated by 3 rows.
s	Vert RGB	N/A	Vertical rainbow color effect, repeated by 3 rows.
t	Black	0,0,0	

5.1.3.4 Graphics selection

Graphics selection is always two bytes long. First character is reserved character. Second character is graphics index: '0' to '9' or 'A' to 'Z'.

Reserved character code:	Decimal	Hexadecimal
Lower (new) set	7	07
Higher (old) set	252	FC

[version < 1.2.0] limitation: Graphics will not be displayed at all.

5.1.3.5 Temperature

Temperature display selection is always two bytes long. First character is reserved character. Second character is: 'A' for Fahrenheit, 'B' for Celsius display.

Reserved character code:	Decimal	Hexadecimal
Lower (new) set	11	0B
Higher (old) set	249	F9

[version < 1.1.0] limitation: Temperature display not supported.

[version >= 3.1.1] feature: Second character of 'C' or 'D' is also supported. Using this characters only the temperature value will be displayed without appending it with degree sign and F/C character. 'C' displays Fahrenheit value, 'D' displays Celsius value.

5.1.3.6 New line character

New line character is the delimiter used to separate lines in a text program.

Reserved character code:	Decimal	Hexadecimal
Lower (new) set	13	0D
Higher (old) set	127	7F

5.1.3.7 Time and countdown

Time and countdown display is two or more bytes long. First character is a reserved character. Second character selects what sort of detail to display. There may be additional characters dependant on the parameters of the detail to be displayed.

Reserved character code:	Decimal	Hexadecimal
Lower (new) set	6	06
Higher (old) set	250	FA

Detail selection characters:

Selection character	Parameter length	Description
'A'	0	hh:mm:ss
'B'	0	hh:mm:ss A/PM
'C'	0	hh:mm
'D'	0	hh:mm A/PM
'E'	0	mm/dd/yyyy
'F'	0	yyyy-mm-dd
'G'	0	dd.MM yyyy
'H'	0	mm'dd'yyyy

Selection character	Parameter length	Description
'I'	0	Day-of-week short format
'J'	0	Day-of-week full word format
'K'	12	Time countdown. **Supported by [version >= 4.0.0]**
8_{10} (0x08)	Variable	Custom date/time display. Parameter field is closed by 9_{10} (0x09) character. **Supported by [version >= 1.7.0]**

Custom date/time parameter field codes:

- yy: Year – last two digits
- yyyy: Year
- M: Month as 1-12
- MM: Month as 01-12
- MMM: Month by short name
- MMMM: Month by full name
- d: day as 1-31
- dd: day as 01-31
- ddd: day-of-week short format
- dddd: day-of-week full format
- h: Hour as 1-12
- hh: Hour as 01-12
- H: Hour as 0-23
- HH: Hour as 00-23
- m: Minute as 0-59
- mm: Minute as 00-59
- s: Second as 0-59
- ss: Second as 00-59
- t: A or P (by AM/PM)
- tt: AM or PM

Other characters are displayed unaltered.

Time countdown ('K') details (only Supported by [version >= 4.0.0]):

- Parameters on additional 12 characters:
  - 6 decimal digits of countdown start time as HHMMSS
  - 6 decimal digits of countdown time (length) as HHMMSS.
- Displays countdown as HH:MM:SS. If higher digits of the countdown time is replaced with non-numerical digits, display format can be shortened. (Look at the example below.)

During the day, before countdown start time, countdown time will be shown. After countdown ends, 00:00:00 will be displayed for the rest of the time. Special case when countdown overlaps (or ends) at midnight. Then the non-counting period is split evenly, first half will show 00:00:00, second half will show countdown time.

Parameter example #1: “201215000500” - Countdown will start at 20:12:15, and counting down from 00:05:00 (5 minutes), displayed as HH:MM:SS.

Parameter example #2: “201215---500” - Same countdown as in example #1, but display format will be: M:SS.

[version >=2.3.0] additional feature: When RTC chip clock is stopped (or never has been set) all date and time related data is displayed having '?' character instead of any data character until the date and time gets set with Set Date and Time (“WA”) command. From **[version >=4.0.0]**, '?' can be changed to any other character. Look at section 5.4.3 describing 'W' + 0x07 – Change Additional Configuration command.

When the mainboard having no RTC chip installed, the date and time is only simulated and will always start from a fixed (past) date/time after powering up the display. (In recent firmware versions from 2012.01.01 00:00:00.) It also has to be taken into account that the simulated clock is not accurate: the seconds are derived from CPU clock that having tolerance of 0.5 to 5%. Therefore it is not suitable for applications where accurate timing is required.

5.1.3.8 TAB positioning and TAB leaded parameters

[Version < 5.0.0] limitation: only the following TAB positioning is available:

First character is reserved character 9_{10} (0x09). Next three characters are decimal digits. The number formed by the three digits will define the horizontal pixel position in the current line for the text that follows.

TAB positioning will prevent the line from scrolling even if it does not fit onto the screen.

[Version >= 5.0.0] additional features:

When the first character is:

- '0' – '9': Same as before - TAB positioning.
- 'A' – 'Z': Also TAB positioning, used to address pixels 1000-3599.
- Others: can be TAB leaded parameters.

TAB positioning can be used in boxed mode too. The pixel position is relative to the left side of the box.

TAB leaded parameters:

After the parameter selection character, the remaining two characters must be hexadecimal digits that carries the parameter value.

These parameters can be used:

- **'r' – roll restart setting:** Set the number of roll steps (empty column of pixels) rolled in after a rolling line print completes, before printing restarts. In other words the size of the gap between the head and tail of a rolling line. By default the roll will restart after the tail of the line has rolled out the screen (or box). For a line that is not rolling, this parameter have no effect at all.
- **'<' – forced roll restart setting:** Same as 'r', but the line will be forced to roll even if it would fit into the screen or box.
- **'x' – horizontal offsetting:** By the given (signed!) value the current print horizontal position will be changed relatively. This can be used to put exact pixel sized gaps (spaces) between characters by using positive values, or with negative values to print over something. (See print clear method font modifiers.)
- **'y' – vertical offsetting:** By the given (signed!) value the current print vertical position will be changed relatively. This can be used to sub- or superscript in text. Use positive value to elevate text, negative to lower it
- **'v' – set vertical alignment in boxed mode:** The given signed value sets vertical align:
  - Zero and positive values: Set above the bottom of the box by the given value.
  - -1: Center vertical alignment
  - -2 and below: Set alignment below the top of the box by 0.. (N=-value-2) pixels.

5.1.3.9 Variable selection

Variable selection is always two bytes long. First character is reserved character. Second character is variable index: '0' to '9' or 'A' to 'V'.

Reserved character code:	Decimal	Hexadecimal
Lower (new) set	12	0C
Higher (old) set	251	FB

[version < 1.5.0] limitation: Variables not supported.

5.1.3.10Effects

From **[Version >= 5.0.0]** effects are implemented, but disabled by default. The reason is to maintain compatibility with previous versions. See section 5.4.3 about 'W' + 0x07 – Change Additional Configuration command how to enable effects.

List of effects by reference characters:

Effects can have two phases, one before and one after holding screen content for the desired hold time.

- Underscored effect having no post-hold effect phase.
- *Italic* effect having no effect phase before holding screen content.

Ref:	Effect:	Ref:	Effect:	Ref:	Effect:	Ref:	Effect:
'B'	Flash	**'C'**	*Hold*	**'D'**	Interlock	**'E'**	Roll down
'F'	Roll up	**'G'**	Roll in	**'H'**	Roll out	**'I'**	Roll left
'J'	Roll right	**'K'**	Rotate	**'L'**	Slide	**'M'**	Snow
'N'	Sparkle	**'O'**	Spray	**'P'**	Star burst	**'Q'**	Switch
'R'	Twinkle	**'S'**	Wipe down	**'T'**	Wipe up	**'U'**	Wipe in
'V'	Wipe out	**'W'**	Wipe left	**'X'**	Wipe right	**'Y'**	Cycle color
'Z'	Clock	**'['**	*Hold+Roll*	**'\'**	Roll+Hold		

Auto effect: By using **'A'** character for effect selection the display will use the character given for enabling effects. (See section 5.4.3.) If that character was **'A'** the display will randomly choose an effect from the above list of effects.

Small letters: By using small letters for effect selection the post-hold effect phase will not be made, only the effect phase before holding the content of the screen. Small **'a'** character will do the same as **'A'** would but without post-hold effect.

Special function of the '!' character:

This is a very special effect that keeps the content of the previous screen. (No other effect will be used.) This is most likely useful for screens with a rolling line(s).

Usage examples:

- Push out the previous screen content out by a new (full screen) rolling content. (Roll will

not start from a blank screen.)

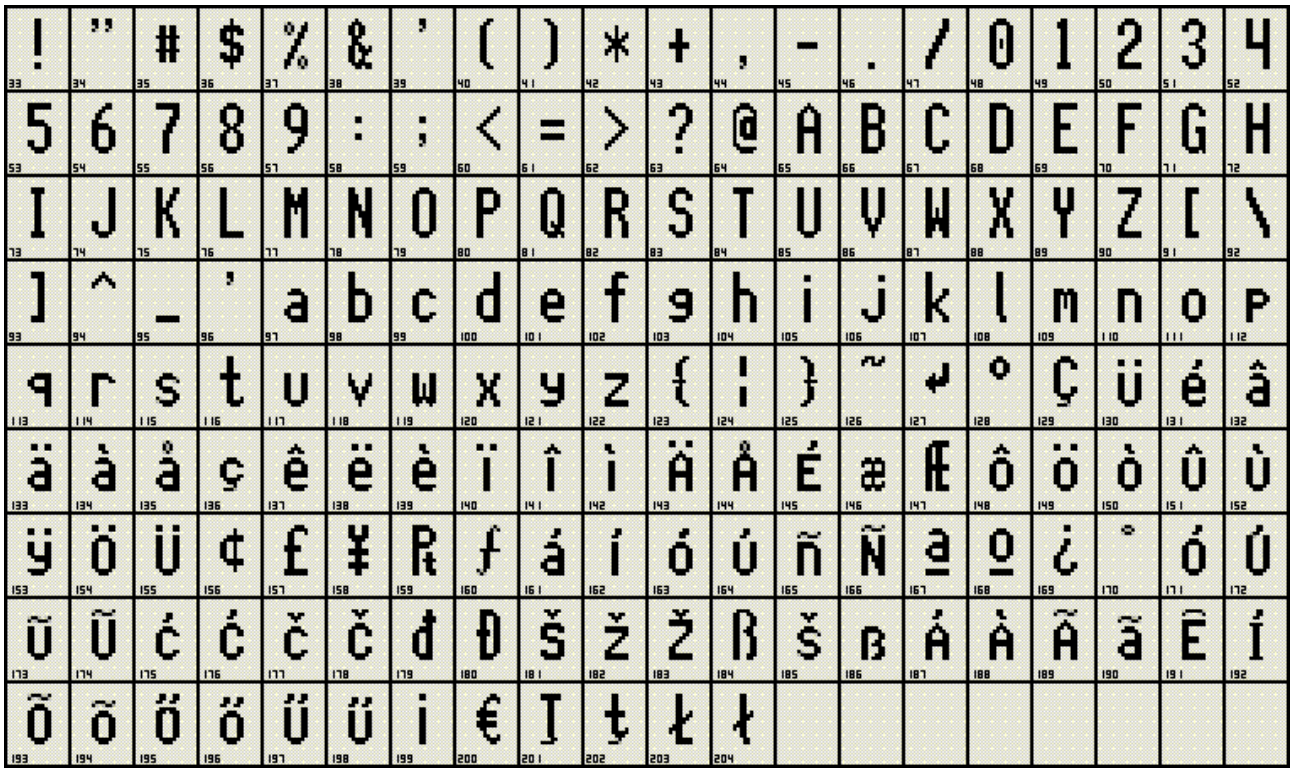
- Keep some content and replace partially the content of the screen.

5.1.3.11 Default character map

The following table shows the current character map (without the space character at ASCII 32). All generic fonts are implementing these characters regardless their size. This example contains the font faces of SS15 (7x15 pixels) font.

On request the same fonts without having the zero character crossed ('Ø') are also available.

[version >=5.0.0] default font set changed. The default font set will have not crossed zero character, and the crossed zero variant have to be requested in an order.



5.2 'E' - Write graphics

Command 'E' uses FFFF checksum calculation method.

[version < 1.2.0] limitation: Write graphics command not supported.

This command is used to load graphics on the display. (Preferably prior using it in a text program.)

5.2.1 Command field structure:

command	index	graphics properties	graphics data
'E'	'0'-'9', 'A'-'Z'	Height, width and load method	Pixel colour information

Graphical data is always loaded in the order of top row first, from left to right. Some loading methods having more than one pixel per byte, for those methods bits from high to low representing pixels from left to right.

Each graphics data byte value must be between 64 (0x40) and 191 (0xBF) inclusively.

5.2.1.1 Graphics properties

The following fixed size fields are defined in graphics properties

Field	Length (bytes)	Description
Height	2	Height of graphics in pixels as hexadecimal digits
Load method	1	Load method (translation) of graphics data
Width	2	Width of graphics in pixels as hexadecimal digits

- If height is larger than display height, the lower exceeding pixel rows are neither stored or displayed, however the graphics could be loaded and partially displayed successfully. Zero height will just command to delete a previously loaded graphics.
- Load method can be compatible (legacy) or by other methods listed and described fully in the table below.
- Zero width will just command to delete a previously loaded graphics. First hexadecimal character is allowed to be above 'F' to allow the representation of graphics widths up to 575 pixels - represented by “ZF”.

Load method list:

Method selection character	Data representation	Description
','	colour table	1 byte per pixel. Compatible (legacy) load mode.
'A'	R from table	Compressed storage as 1 bit per pixel. Only first colour layer is stored.
'a'	RRRRRRR	Compressed load and storage as 1 bit per pixel. Each graphics data byte contains data of seven pixels. Will be displayed on all colour layers. (Compressed representation of black and white graphics.)
'B'	RG from table	Compressed storage as 2 bits per pixel. First and second colour layer with colour depth of 1 bit.
'b'	0RGRGRG	Compressed load and storage as 2 bit per pixel – one bit for first and second colour layer. Each graphics data byte contains data of three pixels.
'C'	Rr from table	Compressed storage as 2 bits per pixel. Loaded as first colour layer only with colour depth of 2 bits. Will be displayed on all colour layers. (Compressed representation of greyscale graphics.)

Method selection character	Data representation	Description
'c'	0RrRrRr	Compressed load and storage as 2 bit per pixel. Loaded as first colour layer only with colour depth of 2 bits. Each graphics data byte contains data of three pixels. Will be displayed on all colour layers. (Compressed representation of greyscale graphics.)
'D'	FR from table	Compressed storage as 2 bits per pixel. Flash and first colour layer only with colour depth of 1 bit. Will be displayed on all colour layers.
'd'	0FRFRFR	Compressed load and storage as 2 bit per pixel. Loading and first colour layer only with colour depth of 1 bits. Each graphics data byte contains the data of three pixels. It will be displayed on all colour layers.
'E'	FRGB from table	Compressed storage as 4 bits per pixel. Flash and all three colour layers with colour depth of 1 bit.
'e'	0RGBRGB	Compressed load and storage as 3 bit per pixel – one bit for all three colour layers. Each graphics data byte contains data of two pixels.
'F'	RrGg from table	Compressed storage as 4 bits per pixel. First and second colour layers with colour depth of 2 bits.
'f'	000RrGg	Compressed storage as 4 bits per pixel. First and second colour layers with colour depth of 2 bits. Using direct data representation instead of table lookup.
'g'	0FRGFRG	Compressed load and storage as 3 bit per pixel – flash plus first and second colour layer only with one bit colour depth. Each graphics data byte contains data of two pixels.
'H'	FRrGgBb from table	Similar to compatible/legacy mode (',') - but all details are always stored.
'h'	FRrGgBb	Using direct data representation instead of table lookup.
'X'	FRrrrGg + 0ggBbbb	2 bytes per pixel load format for more than 2 bits colour depths.

Black and white graphics:

Data loaded having the first colour layer only (modes: 'A', 'a', 'C', 'c', 'D' and 'd') will be displayed on all colour layers.

Colour table:

Colours are represented the same way as it has been described for font colours.

Flashing colour is colour character ASCII code + 64. Example of flashing red: 'B' + 64_{10} = 130_{10} (0x82).

Characters not listed in the colour table are implemented as black.

Bulk data bytes:

For load modes where the pixel colours are not derived from the colour table, data bytes are having direct representation of pixel data bits. If not all seven bits are used, the highest bits will always be filled with zeroes. Finally, 64 should be added to get the data byte in the graphics data field.

Example:

- Mode 'e': 0RGBRGB
- Pixels: two yellow pixels.
- Bulk data: $0110110_2 = 0x36 = 54_{10}$
- Data byte in the graphics data field: $0x76 = 118_{10}$

5.2.2 Graphics storage considerations

Graphics data size in the write graphics command cannot exceed 4000 bytes in length. However with compressed storage loads this could allow storing up to 32000 pixels in 1 bit per pixel data modes.

In compatible (legacy) load mode (',') the controller will select the most efficient data packing method to reduce graphics size in its memory. However this is only possible if one or more from the following list is relevant for a particular display:

- Colour depth is only one bit per colour layer.
- Number of base colours (colour layers) is less than three
- Doesn't have a flash support layer (default if not requested)

Storage bits required per pixel: colour_depth * base_colours + flash_layer. (Flash layer requires 1 additional bit.)

Pixels cannot span across byte boundaries.

- in case of 3 bits per pixel, 4 bits per pixel of storage will be used. (2 pixel per byte.)
- in case of 6 or 7 bits per pixel, 8 bits (1 byte) per pixel of storage will be used.

5.2.3 Checking graphics load

Graphics frames can be relatively large. Since even frames with problems are acknowledged, it is required to use another method to check a load.

Use Get Graphics Information (“Re”) command to check whether a graphics has been successfully loaded (or deleted.)

5.3 'C' - Write Variable

Command 'C' uses FFFF checksum calculation method.

[version < 1.5.0] limitation: Write variable command is not supported.

This command is used to load, update or delete a variable.

5.3.1 Command field structure:

command	index	variable properties	variable data
'C'	'0'-'9', 'A'-'V'	length and colour	text

Delete variable action:

Command works as delete variable when:

- variable data (text) length is 0 characters.
- variable properties are not supplied or only partially included.

Limitations:

- **[Version<5.0.0]:** Control characters in variable data handled as regular characters, not control characters.
- Variable data (text) cannot exceed 250 characters.
- The total length of all variables cannot exceed 4000 characters.
- If there's not enough RAM to store the updated set of variables, the updating frame will be dropped and the variable update will not happen. This could only happen in case the new variable text length is longer than the previous.

[Version>=5.0.0] additional feature:

Some of the lower (new) set control characters are allowed to be used in variables and will be handled as their function described:

- Allowed to change fonts, use font modifiers but only with lower (new) selector of 0x0F.
- Allowed to change color but only with lower (new) selector of 0x10.
- Allowed to reference graphics but only with lower (new) selector of 0x07.
- Allowed to use TAB positioning or TAB leaded parameters (selector 0x09)
- Not allowed to reference date or time data (selector 0x06)
- Not allowed to reference temperature data (selector 0x0B)
- Not allowed to reference variable (selector 0x0C)

Higher (old) set control characters will be displayed as regular characters.

5.3.1.1 Variable properties

The following fixed size fields are defined in variable properties

Field	Length (bytes)	Description
Length	2	Length of variable in characters as two digit hexadecimal number. 00 for auto-size. This value is not used at the moment.
Colour	1	Colour ID character – display colour of the variable. Same colour ID character table is valid as it has been defined for

Field	Length (bytes)	Description
		Write Program command. See section 5.1.3.3 Colour select. **[version >= 1.5.1] feature:** ' ' – space character means no colour selection – uses the currently selected program text colour.

5.3.2 Checking variable update / Reading out

Some applications may require reading out the content of variables. Since even frames with problems are acknowledged, it is required to use another method to check whether an update went through.

Use Read Variable (“Rc”) command to check whether a variable has been successfully updated (or deleted.)

[version <4.0.0] limitation: “Rc” command is not supported.

5.4 'W' – Write configuration

Write configuration commands are used to change various settings of the LED display.

Most write configuration commands are using ADD[5] checksum calculation method. Exceptions are: 'WA', 'WE' and 'WP' that are using FFFF checksum calculation method.

5.4.1 'W' + 0x05 – Enter key

During key exchange the previously queried key should be returned by using this command in order to finish the unlock sequence.

[version <2.0.0] limitation: Command not supported.

Field:	command	subcommand	key
Offset:	0	1	2
Length:	1	1	4
ASCII data:	'W'	0x05	Hexadecimal digits

5.4.2 'W' + 0x06 – Change Configuration Word

[version <2.2.0] limitation: Command not supported.

Key exchange may be required before executing this command. (Depends on the value of NotSecureConfig bit in configuration word.)

5 Previous versions [version<=V2.2.1] of this document incorrectly stated that most write configuration commands are using FFFF checksum calculation method.

Field:	command	subcommand	Change method	data
Offset:	0	1	2	3
Length:	1	1	1	4
ASCII data:	'W'	0x06		Hexadecimal digits

5.4.2.1 Method characters

Character	function	description
'0'	Zeroise / turn off	Bits that are one in data will be zeroised in configuration word.
'1' or '|'	Inclusive OR or turn on	Bits that are one in data field will be set to one in configuration word.
'='	Set all bits	Data field value will be the value of configuration word
'^'	Exclusive OR	Bits that are one in data field will be inverted.
'&'	And	Bits that are zero in data will be zeroised in configuration word.

5.4.2.2 Using '-' character in a data field

This feature is available when **[version >=4.0.0]**.
This character is allowed to mark that on that hexadecimal digit no change necessary. If there's no previous value on that digit, the default value will be used.

5.4.2.3 Configuration Word bits

Bit	name	description
0-3	AddedSpacing	Number of additional empty pixel rows inserted between displayed characters.
4	AlwaysMaxSpacing	When set, displaying font characters will always use the regular spacing gap, never the minimal gap.
5	UseProgConstraints	When this bit set, program constraints will be checked. (Turns on play-by-time mode)
6	DisableSummerTime	When this bit is set, date/time will not be adjusted to daylight savings time and back. (Accordingly to EU daylight savings standard.)
7	RamMode	When set, flash write functionality is turned off. Only used by firmware **[version >=4.0.0]**.
8	StoreBrightness	When this bit is set, brightness setting will be stored. Do not turn on this bit if brightness is changed frequently!
9	StorePowerOnOff	When this bit is set, power on/off times will be stored. Do not turn on this bit if power on/off times are changed frequently!

Bit	name	description
10	SecuredBrightness	When this bit is set, change brightness command requires key exchange sequence. (When NotSecureConfig is not set.)
11	NotSecureConfig	When this bit is set, write configuration commands doesn't require key exchange sequence.
12	SecuredDateTime	When this bit is set, set date/time command requires key exchange sequence. (When NotSecureConfig is not set.)
13	CheckSumOff	When this bit set, communication checksum will not be checked. (Any 4 digit hexadecimal number will be accepted.)
14	NoSplashScreen	**[version >=3.1.0]:** Disable displaying of Splash Screen. **[version <3.1.0]:** Not used by firmware.
15	UserConfig0	Bit for user function. Function is firmware dependent.

Default values are usually zero for all fields. Only exception is RamMode: this bit is usually 1 – when Flash functionality is not included in the firmware.
(Special configurations may have different default values.)

5.4.3 'W' + 0x07 – Change Additional Configuration

[version <4.0.0] limitation: Command not supported.

Key exchange may be required before executing this command. (Depends on the value of NotSecureConfig bit in configuration word.)

Field:	command	subcommand	change method	ID	primary data	more data
Offset:	0	1	2	3	5 or 7	9
Length:	1	1	1	2 or 4	2 or 4	n*6
ASCII data:	'W'	0x07		Hexadecimal digits	Hexadecimal digits or '-'	Hexadecimal digits or '-'

5.4.3.1 Additional configuration data IDs

Additional configuration data is identified by ID.

ID can be either 8 or 16 bits long (2 or 4 digits). ID and primary data length is always 24 bits (6 digits).

- When the first two (hexadecimal) digits are 00 or 80 then ID is 16 bits (4 digit) long and primary data is a byte (2 hexadecimal digits).
- If the first two (hexadecimal) digits are neither 00 nor 80 then ID is 8 bits (2 digit) long and primary data is a word (4 hexadecimal digits).

An ID can be user or system. System IDs are when the first ID digit is 8 or above.

- User IDs are available to be set freely by the user. However the total user data length is more limited than it is for system IDs (in order to let enough free space for system use).
- System IDs are used to store several system settings. See the list below about the currently

assigned IDs. Even unused (reserved) system IDs can be written, please never write those IDs unless it has been instructed! User data written to this IDs can cause severe problems (like a firmware deadlock) after a firmware upgrade!

5.4.3.2 About more data

More data is a possibility to store information that is more than 8 or 16 bits long. Important considerations when more data is used:

- More data is stored in 22bit long packets. The command expecting 6 hexadecimal digits so the top 2 bits will not be used/stored. Less than 6 digits per more data packet will be considered as an error, and the command will not be executed.
- When more_data is used, the lowest 2 digits of final primary data should match the number of more_data packets. (This may not be possible with other methods than '='. Be careful using them.)

5.4.3.3 Using '-' character in a data field

This character is allowed to mark that on that hexadecimal digit no change necessary. If there's no previous value on that digit, the default value will be used (usually '0').

5.4.3.4 Method characters

The same method characters used that are available for Change Configuration Word command:

Character	function	description
'0'	Zeroise / turn off	Bits that are one in primary data and more data will be zeroised in configuration word.
'1' or '|'	Inclusive OR or turn on	Bits that are one in primary data and more data fields will be set to one in configuration word.
'='	Set all bits	Primary data and more data field value will be the value of the additional configuration
'^'	Exclusive OR	Bits that are one in primary data and more data field will be inverted.
'&'	And	Bits that are zero in primary data and more data field will be zeroised in configuration word.

5.4.3.5 Assigned system IDs

Access types:

- RO: Read only – Cannot be changed by 'W' + 0x07 command, 'R' + 0x07 command can read its data.
- RW: Read and write – Can be changed by 'W' + 0x07, and read by 'R' + 0x07 command.

ID	access	description

8001	RO	(Soft) Communication ID. Used only when communication ID hardware is having zero value.
8002	RO	Power on/off times. Should have more_data records.
8003	RW	RTC calibration data
8004	RW	Invalid time replacement character
8005	RW	'A' Effect character. If this parameter value is zero, effects are disabled. Other values enabling effects and determining what effect shall be used when 'A' effect is selected. If this parameter value is 'A' that means auto (random) effect selection.
8006	RW	Legacy flags: • bit 0: Disable higher reserved character set (0xF9-0xFE) • bit 1: Disable high line break character (0x7F) • bit 2: Enable auto (random) color selection • bit 3: Disable smart half pull down of line
81	RO	Configuration Word
82	RO	Brightness setting. May have more_data records.
83	RO	Configuration ID
84	RO	Version (M.m.s) word: bit [15:11]:Major, bit [10:6]:minor, bit [5:0]:sub
85	RO	Flash endurance counter (number of configuration flash page used)
86	RO	Heat shock value (highest temperature recorded)
87	RO	Boot flash request (never visible)
88	RW	Communication timeout setting. Bit #15: instant timeout on error; Bit #14: timeout time unit: 0=0.1 second, 1=1 minute; Bit [13:0] timeout time unit multiplier.
89	RW	Display screen width setting. Only useful in firmware configurations prepared to have variable screen width.
8A	RW	Default roll restart position (signed 16 bit value) • Positive values (and zero) set the roll restart trigger position from the left side. • Negative values set the roll restart trigger position from the right side.
8B	RW	Temperature sensor calibration: Signed multiply constant (part of it will be sign added to 0x40000 - which means 1.0)
8C	RW	Temperature sensor calibration: Signed offset constant. will be sign added to the 13 bit result value (after multiply part made)
>=C0	RO	Reserved for storing more_data records

5.4.3.6 Examples

- Set communication timeout to 2 seconds, no instant error timeout: 'W' + 0x07 + “=880014”
- Enable instant error timeout #1: 'W' + 0x07 + “|888000”
- Enable instant error timeout #2: 'W' + 0x07 + “|888---”
- Set display size to 64 pixels: 'W' + 0x07 + “=890040”
- Set display size to 384 pixels: 'W' + 0x07 + “=890180”
- Set user primary data=1876 on ID=0x02: 'W' + 0x07 + “=021876”
- Set user primary data=18, more_data=037654 on ID=0x02: 'W' + 0x07 + “=021801037654”
- Set user primary data=F8, more_data=255404376 on ID=0x42: 'W' + 0x07 + “=42F802255404376---”

5.4.4 “WA” – Set date and time

This command sets date and time of the controller.

This command uses FFFF checksum calculation method.

[version 1.0.0] limitation: Command not supported.

[version >=2.2.0] feature: Depending on Configuration word settings, key exchange sequence might be required before sending this command!

After command characters “WA” there should be 15 more characters: “YYYYMMDDHHMMSSw” where:

- YYYY: year
- MM: month
- DD: day (of month)
- HH: hour
- MM: minute
- SS: second
- w: day-of-week

The 'w' value is not used and only there for compatibility reasons. The controller calculates day-of-week value from the given date.

5.4.5 “WB” – Soft Reset

This command restarts the display firmware. Execution of this command is takes 2-3 seconds. During that period the display doesn't communicate, and doesn't propagate frames between its channels.

[version <4.0.0] limitation: Command not supported.

Key exchange may be required before executing this command. (Depends on the value of NotSecureConfig bit in configuration word.)

5.4.6 “WD” – Set communication ID

Used this command to change software-based communication ID. This ID is only used when hardware based ID is set to 0 by setting all switches off on “COMM_ID” DIP switch.

[version <2.2.0] limitation: Command not supported.

5.4.7 “WE” – Set up power on and off times

This command uses FFFF checksum calculation method.

To set up power on and off times, up to four records can be used. Each record is 10 characters long:

- 4 characters of power on time as “HHMM”
- delimiter (usually a ':', but firmware doesn't check this character)
- 4 characters of power off time as “HHMM”
- delimiter (usually a ',' or '.' for the last one but firmware doesn't check this character)

Unused records should be filled with '0' characters.

Periods shouldn't overlap – in that case firmware behaviour is undefined.

Example:

“WE0100:0200,1123:1839,0000:0000,0000:0000.” means:

- Two active records:
  - Switch on at 01:00, switch off at 02:00
  - Switch on at 11:23, switch off at 18:39

[version 1.0.0] limitation: Command not supported.

[version >=2.2.0] feature: Depending on Configuration word settings, key exchange sequence might be required before sending this command!

5.4.8 “WF” – Turn on/off checking of program constraints

Used to switch between play all and play-by-time mode.

[version < 1.1.6] limitation: Command not supported.

[version >=2.2.0] feature: Depending on Configuration word settings, key exchange sequence might be required before sending this command!

This command is always 3 characters long:

- “WFT”: Play-by-time - Switch on checking of program constraints
- “WFA”: Play all - Switch off checking of program constraints

5.4.9 “WN” – Set remaining number of programs

5.4.10 “WL” – System Clear

Deleting all programs, variables and graphics.

“WL” command uses ADD type of checksum calculation method.

Key exchange may be required before executing this command. (Depends on the value of NotSecureConfig bit in configuration word.)

5.4.11 “WP” – Set brightness

This command uses FFFF checksum calculation method.

Set brightness command format:

Field:	command	subcommand	Brightness mode	Time table
Offset:	0	1	2	3
Length:	1	1	1	20
ASCII data:	'W'	'P'	'A', '1'-'8' or 'T'	See below

Time table is only required to be sent when brightness mode is 'T'. (It does no harm to send time table always. When brightness mode is not 'T', it's data will not be checked or used at all.)

[version >=2.2.0] feature: Depending on Configuration word settings, key exchange sequence might be required before sending this command!

5.4.11.1 “WPA” – Set to automatic brightness control

Brightness is set to be controlled by light sensor.

5.4.11.2 “WP” + <number 1 to 8> - Set brightness to the given level

Levels: 1=brightest, 8=darkest

5.4.11.3 “WPT” – Set brightness by time

Brightness is being controlled by time to the given ('1'-'8') level. Up to 4 records can be used. Unused records should have their brightness value set to '0'.

Each record is 5 characters long:

- 4 characters of time as “HHMM”
- 1 character of brightness value ('1' to '8') or '0' to mark unused records

Example:

“WPT06003090011931800000” means:

- Three active records:
  - At 06:00 brightness is set to level '3'

- At 09:00 brightness is set to level '1'
- At 19:31 brightness is set to level '8'

[version 1.0.0] limitation: Set brightness by time is not supported.

[version >= 1.5.7] feature: 0x80..0xFF values also allowed on the brightness defining character positions to set brightness by a finer grain.

5.4.11.4 "WP" + 0x80..0xFF - Set brightness to the given level in finer grains

[version < 1.5.7] limitation: This brightness setup method is not supported.

There are 128 available brightness levels with this method. Examples:

- 0x80 is the darkest, which is a 1/16th of the brightness of "WP8".
- 0x8F is the same brightness as it would be with "WP8"
- 0x9F is the same brightness as it would be with "WP7"

  ...
- 0xFF is the brightest, which is exactly the brightness of "WP1".

Intermediate brightness levels are calculated linearly.

5.4.12 "WY" – Switch between base and expanded mode

Used to switch between base and expanded mode. "WY" command can only switch off checking of program constraints.

[version < 1.1.6] limitation: Command not supported.

[version >=2.2.0] feature: Depending on Configuration word settings, key exchange sequence might be required before sending this command!

This command is always 3 characters long:

- "WY0": Switch to base mode
- "WY1": Switch to expanded mode

5.4.13 "WZ" – LED Parameter Setup

This command is used to set up various display parameters. It can be reached in Multimedia LED software by [CTRL]+[F3] magic keystroke combination.

The command uses ADD type checksum.

[version < 1.5.7] limitation: Command not supported.

[version >=2.2.0] feature: Depending on Configuration word settings, key exchange sequence might be required before sending this command!

Field	Offset	Length	Data
Command	0	1	'W'

Field	Offset	Length	Data
Subcommand	1	1	'Z'
Horizontal display size	2	2	Hexadecimal value – not used by firmware
Flash/RAM mode	4	1	'0'=Flash mode, '1'=RAM mode
Mono/Tricolor mode	5	1	'0'=Monocrome, '1'=Multicolor – not used by firmware
RS232/RS485	6	1	'0'=RS232, '1'=RS485 mode. (It has to do nothing with the communication interface type. More likely it is used to respond all commands or just the properly addressed ones.) Setting not used by firmware and it's always in "RS485" mode – only frames with proper target address will be responded.
Sign display at reset	7	1	Display ('1') or not ('0') a text/logo at startup – not used by firmware.
Fixed font	8	1	'0'=proportional, '1'=fixed fonts – not used by firmware, fixed printing is fonts selected or by font modifiers.
Row spacing	9	1	Usual values: '1'-'4'. Firmware uses this as additional column spacing added between characters. '1'-'4' is translated to 0-3 additional columns respectively. This value both added to minimum and maximum spacing values.
Remote on/off	10	1	Infrared remote control allowed ('1') or not ('0') – not used by firmware.

5.4.14 'W' + 0xFF – Direct digital output

This command supported only by special configurations. Regular displays doesn't have direct digital output.

Command format after 'W' + 0xFF is implementation dependent. Most common implementation is when some external electronic device (buzzer/light/horn) is switched on and off. Examples for this case:

- 'W' + 0xFF + '1': Switching on the external device
- 'W' + 0xFF + '0': Switching off the external device

[version 1.0.0] limitation: Command not supported.

5.5 'R' - Read configuration

Command 'R' uses FFFF checksum calculation method (for most of its subcommands).

This command is used to get all sorts of different information from the display.

Because this command responds with data, it uses long response frame.

Response data always starts with two characters:

command	subcommand
'W'	

As you can see the response alters command code from 'R' to 'W'

5.5.1 'R' + 0x05 – Read key and other information

This command is used to query information that are special features of the mainboard.

[version <2.2.0] limitation: Command not supported.

Command layout:

command	subcommand	information selector character
'R'	0x05	

5.5.1.1 'R' + 0x05 + 'A' – Address information

Return device address. Hardware (COMM_ID DIP switch) and software (from configuration area) address is returned as 2x2 hexadecimal digits.

5.5.1.2 'R' + 0x05 + 'C' – Read configuration area

This command is only used for firmware testing reasons.

Another character in command is required to specify which memory section to be read. That character must be from 0x40 to 0xBF inclusively (there are 128 memory sections).

The response is 16 doublewords of the requested section.

5.5.1.3 'R' + 0x05 + 'D' - Read Display Dimensions

This command is used to get various specifics of the display.

[version <4.0.2] limitation: Command not supported.

Returns the following details as comma separated decimal values:

1. Rows: Number of pixel rows
2. Colums: Number of pixel columns (current setting if firmware supports changing it)
3. Colors: Number of primary colours (Examples: Red only:1, Amber only:1, Red+Green:2, Red+Amber:2, RGB:3)
4. ColorDepth: Number of bits used to support colour gradients. (Values: 1-4)
5. RefreshFreq: Refreshing frequency in Hz. This is the complete refresh of all drive planes (row sets) and doesn't include refreshing of all bits of available colour depth.
6. DrivePlanes: Number of row sets. Usual values: 1, 2, 4, 7, 8, 15, 16 corresponds to static, 1/2, 1/4, 1/7, 1/8, 1/15, 1/16 driving methods.
7. FlashLayer: Whether direct support of flashing is implemented (1) or not (0).
8. PotentialColumns: Number of pixel columns possible by the buffer configuration. (Only useful if firmware supports changing Columns (#2) value. This is the maximum allowed

value.)

9. MinColumns: Minimum number of columns supported by the firmware. This value reported only when **[Version >= 5.0.0].**
10. MaxLineProc: Number of Lines/Boxes supported on a single screen by the firmware. This value reported only when **[Version >= 5.0.0].**

5.5.1.4 'R' + 0x05 + 'E' – Read configuration area flash endurance value

This command returns current endurance value as 4 hexadecimal digits.

Flash configuration area is reused after 2048 changes in configuration. After that the whole area is erased and endurance value is increased. This way the firmware counts how many erases happened for reliability reasons. After 1000 erases the configuration will not be allowed to change when the configuration area becomes completely full. This limitation is applied in the firmware to ensure the reliability of the configuration area.

5.5.1.5 'R' + 0x05 + 'F' – Read font area

This command is used to extract font data.

Two more characters must be supplied to this command to specify which memory section to be read. Both characters must be from 0x40 to 0xBF inclusively. Section address is (<first_character>-0x40)*64+<second_character>-0x40.

The response is 16 doublewords of the requested section.

5.5.1.6 'R' + 0x05 + 'H' – Read heat shock value

This command is used to query heat shock value. Heat shock the maximum temperature that the sensor ever measured. Temperature that is considered as heat shock is above 70ºC. Heat shock above this temperature is stored in increments of about 3ºC. If operating temperature was always below 70ºC, the returned heat shock value will be 0x0000.

Returned data:

command	subcommand	selector	sensor type	delimiter	heat shock value
'W'	0x05	'H'	'0' or '1'	';'	4 hexadecimal digits

Function requires temperature sensor. If there is no thermal sensor, only sensor type '-' is returned without delimiter and heat shock value.
The returned heat shock value is sensor dependent:

- Sensor '0': MCP9700. 70ºC value: 0x0BA3. 1ºC increment: 24.8212121212.
- Sensor '1': MCP9701. 70ºC value: 0x111D. 1ºC increment is: 48.40136363636.

5.5.1.7 'R' + 0x05 + 'I' – Read device firmware specific information

This command returns various information that is specific to the firmware of the

mainboard. Returned data:

command	subcommand	selector	firmware flash upgrade type	configuration set number	firmware version string
'W'	0x05	'I'	1 character	4 hexadecimal digits	6 or more characters

5.5.1.8 'R' + 0x05 + 'J' – Read JEDEC ID

This command can be used to determine by a host whether a flash chip was installed or not on the mainboard. (Without a flash chip it is pointless to turn on flash mode.)

5.5.1.9 'R' + 0x05 + 'K' – Read key

This command is used to request a key. This is the first step in performing key exchange unlocking.

Key is returned as 4 hexadecimal digits. This key must be returned faster than 4 seconds by Enter key ('W' + 0x05) command to unlock configuration changing commands for 12 seconds. After that period, configuration changing commands will be locked automatically.

NotSecureConfig bit of configuration word must be cleared to protect configuration changing commands by key exchange unlock sequence.

Key value is never 0x0000.

5.5.1.10 'R' + 0x05 + 'O' – Read font offset

This command is used to restore extracted font data after firmware upgrade.

Font offset is returned as 4 hexadecimal digits.

5.5.1.11 'R' + 0x05 + 'R' – Read free RAM amount

This command is used to get the current amount of free RAM available.

Free RAM amount is returned as 4 hexadecimal digits.

[version <4.0.1] limitation: Command not supported.

However it is important that this current value is static only in Classic RAM mode. When (periodically) some data from flash memory is getting loaded into RAM, the result read out by this command will be different from time-to-time.

5.5.1.12 'R' + 0x05 + 'T' – Read current temperature

This command used to query current temperature value.

Returned data:

command	subcommand	selector	sensor type	delimiter	temperature value
'W'	0x05	'T'	'0' or '1'	';'	4 hexadecimal digits

Function requires temperature sensor. If there is no thermal sensor, only sensor type '-' is returned without delimiter and temperature value.
The returned temperature value is sensor dependent:

- Sensor '0': MCP9700.

  - 0ºC value: 1241. 0ºF value: 381. 1ºC increment: 24.8212121212.
- Sensor '1': MCP9701.
  - 0ºC value: 993. 0ºF value: 552. 1ºC increment is: 48.40136363636.

5.5.2 'R' + 0x06 – Read configuration word

This command is used to query current value of the configuration word. Check write configuration word command for layout and function of configuration word bits.

[version <2.2.0] limitation: Command not supported.

5.5.3 'R' + 0x07 – Read additional configuration

This command is used to query the current value of an additional configuration setting. Check write additional configuration command for layout and functions of additional configuration bytes and words.

[version <4.0.0] limitation: Command not supported.

5.5.4 “RA” – Read current date and time

This command is used to read current date and time of the LED display

[version < 2.2.3] limitation: Read current date and time command is not supported.

This command has no additional parameter, it's always the following two characters:

command	subcommand
'R'	'A'

Response data is “WAYYYYMMDDHHmmSSw” where:

- “WA”: response leading characters of read current date and time command
- YYYY: current year
- MM: current month
- DD: current day of month
- HH: current hour in 24H format (00-23)
- mm: current minutes
- SS: current seconds
- w: current day-of-week as '1'=Monday, '2'=Tuesday,... '7'=Sunday.

5.5.5 “RF” – Check State

This command is used to read various information about the settings of the LED display

[version < 1.1.0] limitation: Check State command is not supported.

This command has no additional parameter, it's always the following two characters:

command	subcommand
'R'	'F'

Response data after command ('W') and subcommand ('F') character:

Field	Press key voice	Use Program Constraints	Communication ID	Password
Offset	2	3	4	6
Length	1	1	2	1
ASCII data	'0'	'A' or 'T'	Hexadecimal digits	'0'

Use Program Constraints values:

- 'A': Play all (no constraints)
- 'T': Play-by-time (use constraints)

Press key voice and Password fields:
These fields are always '0' because these features are not supported by the hardware or firmware of the controller.

5.5.6 “RS” – Get System Information

Command used to get general system information from the display. Information is provided as plain text (human readable).

- Display configuration name
- Display configuration ID
- Firmware version
- Country code (if specifically set)
- Communication ID
- Communication setup (about firmware assigned communication channels)
- Current date & time
- Communication checksum info - only when disabled(!)
- **[version >= 2.3.0] feature:** RTC chip type (ID)
- **[version >= 4.0.1] feature:** Flash chip ID
- **[version >= 4.0.1] feature:** Size of buffer RAM

5.5.7 “Ra” – Get Program Text Information

Command used to read CRC32 information about a text program (previously loaded)

[version < 1.3.0] limitation: Get Program Text information command is not supported.

Communication error can occur from time-to-time, and there is no other way to know whether the Write Text Program command ('A') went through or not.

command	subcommand	Text program index
'R'	'a'	'0'-'9', 'A'-'Z'

Response data after command ('W') and subcommand ('a') character:

Field	Index	Number of Lines	CRC32
Offset	2	3	5
Length	1	2	8
ASCII data	Graphics index code	Hexadecimal digits	Hexadecimal digits

CRC32: CRC32 used only on program property and text data bytes of the write text program frame. Check section 6 CRC32 calculation for algorithm.
Special value:
When Number of Lines and CRC32 value is all zeroes that means there is currently no text loaded at that index:

- No text program loaded at that index
- Text program has been explicitly deleted
- Text program loads were all unsuccessful to that index (since delete / initialization)

5.5.8 “Rc” – Read Variable

Command used to read variable data

[version < 4.0.0] limitation: Read Variable command is not supported.

Communication error can occur from time-to-time, and there is no other way to know whether the write variable command went through or not.

command	subcommand	Text program index
'R'	'c'	'0'-'9', 'A'-'V'

Response data after command ('W') and subcommand ('c') character:

Field	Index	data length	color	variable data
Offset	2	3	5	6
Length	1	2	1/0	as in data length field
ASCII data	Variable index code	Hexadecimal digits	character	characters

Colour character field is not included in response when data length field is zero.

5.5.9 “Re” – Get Graphics Information

Command used to read some information about a graphics (previously loaded)

[version < 1.2.0] limitation: Get Graphics Information command is not supported.

Communication error can occur from time-to-time, and there is no other way to know whether the graphics load went through or not.

command	subcommand	Graphics index
'R'	'e'	'0'-'9', 'A'-'Z'

Response data after command ('W') and subcommand ('e') character:

Field	Index	Height	Width	Storage info	CRC32
Offset	2	3	5	8	9
Length	1	2	3	1	8
ASCII data	Graphics index code	Hexadecimal digits	Hexadecimal digits	Bit field (see explanation below)	Hexadecimal digits

Storage info byte:

- Storage info byte has 7 bits of data with 64_{10} (0x40) added to it. To get the actual content first subtract 64 from the value.
- Bits:
  - 6: Flash bit extended storage of the image
  - 5: Colour depth extended storage of the image
  - 4-0: Number of bits/pixel (can only be: 1, 2, 4, 8 or 16)

  Examples:

Storage info byte	Actual data bits	Description
64_{10}	0000000	No graphics data
65_{10}	0000001	1 bit / pixel graphics
66_{10}	0000010	2 bits / pixel graphics, R & G bits for each pixel – two colour layers, 1 bit for each colour layer
98_{10}	0100010	2 bits / pixel graphics, Rr bits for each pixel – single colour layer, colour depth of 2 bits

CRC32: CRC32 used only on graphics property and data bytes of the write graphics frame. Check section 6 CRC32 calculation for algorithm.

Special value set:

When height, width, crc32 is all zeroes and storage info is 64_{10} that means there is no graphics loaded at that index:

- No graphics loaded at that index
- Graphics has been explicitly deleted by having height or width of zero loaded at index
- Graphics loads were all unsuccessful to that index (since delete / initialization)

6 CRC32 calculation

Initial seed is 0. The following code snippet can be used for fast calculation of CRC32:

```
const unsigned long c_Crc32Table[256]={
  0x00000000, 0x04c11db7, 0x09823b6e, 0x0d4326d9, 0x130476dc, 0x17c56b6b, 0x1a864db2, 0x1e475005,
  0x2608edb8, 0x22c9f00f, 0x2f8ad6d6, 0x2b4bcb61, 0x350c9b64, 0x31cd86d3, 0x3c8ea00a, 0x384fbdbd,
  0x4c11db70, 0x48d0c6c7, 0x4593e01e, 0x4152fda9, 0x5f15adac, 0x5bd4b01b, 0x569796c2, 0x52568b75,
  0x6a1936c8, 0x6ed82b7f, 0x639b0da6, 0x675a1011, 0x791d4014, 0x7ddc5da3, 0x709f7b7a, 0x745e66cd,
  0x9823b6e0, 0x9ce2ab57, 0x91a18d8e, 0x95609039, 0x8b27c03c, 0x8fe6dd8b, 0x82a5fb52, 0x8664e6e5,
  0xbe2b5b58, 0xbaea46ef, 0xb7a96036, 0xb3687d81, 0xad2f2d84, 0xa9ee3033, 0xa4ad16ea, 0xa06c0b5d,
  0xd4326d90, 0xd0f37027, 0xddb056fe, 0xd9714b49, 0xc7361b4c, 0xc3f706fb, 0xceb42022, 0xca753d95,
  0xf23a8028, 0xf6fb9d9f, 0xfbb8bb46, 0xff79a6f1, 0xe13ef6f4, 0xe5ffeb43, 0xe8bccd9a, 0xec7dd02d,
  0x34867077, 0x30476dc0, 0x3d044b19, 0x39c556ae, 0x278206ab, 0x23431b1c, 0x2e003dc5, 0x2ac12072,
  0x128e9dcf, 0x164f8078, 0x1b0ca6a1, 0x1fcdbb16, 0x018aeb13, 0x054bf6a4, 0x0808d07d, 0x0cc9cdca,
  0x7897ab07, 0x7c56b6b0, 0x71159069, 0x75d48dde, 0x6b93dddb, 0x6f52c06c, 0x6211e6b5, 0x66d0fb02,
  0x5e9f46bf, 0x5a5e5b08, 0x571d7dd1, 0x53dc6066, 0x4d9b3063, 0x495a2dd4, 0x44190b0d, 0x40d816ba,
  0xaca5c697, 0xa864db20, 0xa527fdf9, 0xa1e6e04e, 0xbfa1b04b, 0xbb60adfc, 0xb6238b25, 0xb2e29692,
  0x8aad2b2f, 0x8e6c3698, 0x832f1041, 0x87ee0df6, 0x99a95df3, 0x9d684044, 0x902b669d, 0x94ea7b2a,
  0xe0b41de7, 0xe4750050, 0xe9362689, 0xedf73b3e, 0xf3b06b3b, 0xf771768c, 0xfa325055, 0xfef34de2,
  0xc6bcf05f, 0xc27dede8, 0xcf3ecb31, 0xcbffd686, 0xd5b88683, 0xd1799b34, 0xdc3abded, 0xd8fba05a,
  0x690ce0ee, 0x6dcdfd59, 0x608edb80, 0x644fc637, 0x7a089632, 0x7ec98b85, 0x738aad5c, 0x774bb0eb,
  0x4f040d56, 0x4bc510e1, 0x46863638, 0x42472b8f, 0x5c007b8a, 0x58c1663d, 0x558240e4, 0x51435d53,
  0x251d3b9e, 0x21dc2629, 0x2c9f00f0, 0x285e1d47, 0x36194d42, 0x32d850f5, 0x3f9b762c, 0x3b5a6b9b,
  0x0315d626, 0x07d4cb91, 0x0a97ed48, 0x0e56f0ff, 0x1011a0fa, 0x14d0bd4d, 0x19939b94, 0x1d528623,
  0xf12f560e, 0xf5ee4bb9, 0xf8ad6d60, 0xfc6c70d7, 0xe22b20d2, 0xe6ea3d65, 0xeba91bbc, 0xef68060b,
  0xd727bbb6, 0xd3e6a601, 0xdea580d8, 0xda649d6f, 0xc423cd6a, 0xc0e2d0dd, 0xcda1f604, 0xc960ebb3,
  0xbd3e8d7e, 0xb9ff90c9, 0xb4bcb610, 0xb07daba7, 0xae3afba2, 0xaafbe615, 0xa7b8c0cc, 0xa379dd7b,
  0x9b3660c6, 0x9ff77d71, 0x92b45ba8, 0x9675461f, 0x8832161a, 0x8cf30bad, 0x81b02d74, 0x857130c3,
  0x5d8a9099, 0x594b8d2e, 0x5408abf7, 0x50c9b640, 0x4e8ee645, 0x4a4ffbf2, 0x470cdd2b, 0x43cdc09c,
  0x7b827d21, 0x7f436096, 0x7200464f, 0x76c15bf8, 0x68860bfd, 0x6c47164a, 0x61043093, 0x65c52d24,
  0x119b4be9, 0x155a565e, 0x18197087, 0x1cd86d30, 0x029f3d35, 0x065e2082, 0x0b1d065b, 0x0fdc1bec,
  0x3793a651, 0x3352bbe6, 0x3e119d3f, 0x3ad08088, 0x2497d08d, 0x2056cd3a, 0x2d15ebe3, 0x29d4f654,
  0xc5a92679, 0xc1683bce, 0xcc2b1d17, 0xc8ea00a0, 0xd6ad50a5, 0xd26c4d12, 0xdf2f6bcb, 0xdbee767c,
  0xe3a1cbc1, 0xe760d676, 0xea23f0af, 0xeee2ed18, 0xf0a5bd1d, 0xf464a0aa, 0xf9278673, 0xfde69bc4,
  0x89b8fd09, 0x8d79e0be, 0x803ac667, 0x84fbdbd0, 0x9abc8bd5, 0x9e7d9662, 0x933eb0bb, 0x97ffad0c,
  0xafb010b1, 0xab710d06, 0xa6322bdf, 0xa2f33668, 0xbcb4666d, 0xb8757bda, 0xb5365d03, 0xb1f740b4};

unsigned long CRC32Calc(const unsigned long seed, const unsigned char chr)
{
        return (c_Crc32Table[(seed>>24)^chr]^(seed<<8));
}

SendGraphicsCharacter(unsigned char chr)
{

        ...

        if (FirstCharacter) Graphics.crc32=CRC32Calc(0,chr);
        else Graphics.crc32=CRC32Calc(Graphics.crc32,chr);

        ...

}
```

Slow calculation variant (not table based):

```
unsigned long CRC32Calc(const unsigned long seed, const unsigned char chr)
{
        unsigned long l;
        unsigned long j;
        l=((seed>>24)^(chr))<<24;
        for(j=0;j<8;j++) {
                if (l&0x80000000) l=(l<<1)^0x04c11db7;
                else l<<=1;
        }
        return ((seed<<8)^l);
}
```

7 Firmware version history

- V5.0.1
  - Bugfix: Boxed mode limited entry mode of coordinates wasn't working as described. Affected version: V5.0.0.
- V5.0.0
  - Change: MinColumns and MaxLineProc value added to the output of 'R' + 0x05 + 'D' - Read Display Dimensions command.
  - Change: Rolling screen update optimised: all empty rows are getting skipped
  - Change: Program update will invalidate current screen instantly (regardless of remaining hold time)
  - Added: Boxed printing mode (screen partitioning)
  - Added: Effects (disabled by default for compatibility reasons)
  - Change: Variable update will update screen and not causing screen invalidation (except when height changes in normal mode)
  - Change: Graphics update will update screen and not causing screen invalidation (except when height changes in normal mode)
  - Change: Temperature, date and time values are updated on the screen (not instantly on a rolling line)
  - Added: Possibility to support multiple display widths with one firmware (Display width adjust by 'W' + 0x07 – Change Additional Configuration command)
  - Bugfix: It was possible that not the original line flags had been saved in flash mode. This may lead to display errors in flash mode. Affected versions: V4.0.0 – V4.0.5.
  - Bugfix: Simple brightness settings were always stored in Flash Configuration Area regardless its Configuration Flag setting. Affected versions: V4.0.0 – V4.0.5.
  - Bugfix: Writing single word custom parameters leaded to writing additional data words (garbage). Affected versions: V4.0.0 – V4.0.5.
  - Change: Lots of code size/speed optimisation
  - Added: Variables can contain Font Selection or Modifier, Color Selection or reference to Graphics
  - Added: More country specific settings available for printing date/time
  - Added: Preferred roll restart position can be changed/stored in Flash Configuration Area
  - Added: Roll restart position can be set on any line (TAB + 'r')
  - Added: Any line can be forced to roll (a variant of line roll restart position setting)(TAB + '<')
  - Added: Current vertical print position can be offset (TAB + 'y')

  - Added: Current horizontal print position can be offset (TAB + 'x')
  - Added: Temperature sensor calibration constants (multiply and add) can be changed/stored in Flash Configuration Area. (See Assigned system IDs.)
  - Change: Default fonts are changed to those where the zero ('0') character is not crossed
  - Bugfix: Date display 'G' has been corrected to display month with short name instead of number. Affected versions: V1.0.0 – V4.0.5.
  - Added: Support for “WL” – System Clear command (Deleting all programs, variables and graphics)
  - Added: Flash Chip JEDEC ID can be read ('R' + 0x05 + 'J' – Read JEDEC ID)
  - Bugfix: Flash saved variables with content but marked with zero text length caused problems at next startup. Affected versions: V4.0.0 – V4.0.5.
  - Added: Auto color mode. Can be enabled by setting the corresponding bit of Legacy Flags. (See in Assigned system IDs.)
  - Added: Special character high variant can be disabled by setting corresponding bits of Legacy Flags. (See in Assigned system IDs.)
- V4.0.5
  - Change: Code size/speed optimised by not using Peripheral Library of Microchip C32.
  - Bugfix: Boot loader marker cannot be written to Flash Configuration Area, preventing one step flash upgrades. Affected versions: V4.0.0-V4.0.4.
  - Change: Boot loader allows direct connection to it during power up.
  - Bugfix: Brightness setting was not loaded from Flash Configuration Area after power up. Affected versions: V4.0.0-V4.0.4.
  - Bugfix: Deleted records was counted as user records in Flash Configuration Area. Affected versions: V4.0.0-V4.0.4.
- V4.0.4
  - Bugfix: Some values cannot be written to Flash Configuration Area. Affected versions: V4.0.0-V4.0.3
- V4.0.3
  - Bugfix: At startup Soft Communication ID value was not used in some cases (01 used instead.) Affected versions: V2.2.0-V4.0.2.
- V4.0.2
  - Added: Read Display Dimensions command ('R' + 0x05 + 'D'). Returns various details: Rows,Colums,Colors,ColorDepth,RefreshFreq,DrivePlanes,FlashLayer, PotentialColumns
- V4.0.1
  - Added: Read Free RAM ('R' + 0x05 + 'R') command

  - Added: RAM buffer size is included in "Get System Information" command result.
  - Added: Flash functionality/chip ID information is included in "Get System Information" command result.
- V4.0.0
  - Added: Flash memory support
  - Added: Supporting temperature value display only without putting the degree sign and C/F character behind.
  - Added: Supporting time display mode 'K' - countdown time
  - Added: Reading out variables is supported (Command: "Rc")
  - Added: Supporting Soft reset command (WB)
  - Added: Custom parameters can be stored in Flash Config Area (Command: 'W' + 0x07)
  - Added: Reading parameters from Flash Config Area (Command: 'R' + 0x07)
  - Added: Invalid time replacement character can be changed (Command: 'W' +0x07 + “=8004” + <character_code>)
  - Added: Communication timeout settings can be changed (Command: 'W' + 0x07 + “=88” + <timeout_setting>)
  - Added: RTC calibration byte can be changed (Command: 'W' + 0x07 “=8003” + <calibration_byte>)
- V3.3.0
  - Added: More font modifiers added to support line vertical drop and transparency for printing
  - Bugfix: Bad default font was selected for index 'N'. Affected versions: V2.1.0-V3.2.2. With default font set. Since it depends on font configuration, therefore a direct test required to determine exactly.
  - Bugfix: Different font rolling rows had font selection lost. Affected versions: all previous.
  - Bugfix: Rolling line start was blurry when there are font modifiers at the start of the line. Affected versions: V1.5.6-V3.2.2.
- V3.2.2
  - Bugfix: Using a character code that has no image in the selected character set at the end of a line causes infinite loop. Affected versions: almost all previous versions. (Cannot be determined without directly testing for it. Only very old versions may not have this bug.)
- V3.2.1
  - Interim release - Should be handled same as V3.2.0
- V3.2.0
  - Added: Communication timeout screen feature (empty screen or with fixed static text(s))

    - Added: All communication failures (framing and CRC errors) can instantly trigger timeout screen
- V3.1.1
    - Change: Splash Screen can handle multiple additional text lines
    - Added: Splash Screen configuration information will split into two lines if Rows>14 and Columns<80
- V3.1.0
    - Added:Splash Screen with configuration information
- V3.0.1
    - BugFix: Minor bugfixes in preparation to flash mode. Affected versions: none
- V3.0.0
    - Added: Boot loader for easy remote firmware updating
    - Added: Firmware or font update support V4 allows writing update trigger required in flash Config area to invoke boot loader.
    - Change: Internal flash handling robustness increased
    - Added: All commands can accept checksums calculated by any of the three checksum calculation methods.
- V2.3.0
    - Added: RTC chip always checked (including no RTC chip configurations). Chip type gets automatically detected.
    - Added: MCP7940x RTC chip (type #2) is also supported.
    - Added: RA and RS (Get System Information) command shows if clock has been stopped(invalid) and RS shows chip type too.
    - Added: Date/time displayments are filled with '?' characters when RTC chip clock is stopped
- V2.2.6
    - Added:Flash version V3 allows simpler font change
- V2.2.5
    - Added:Digital input PCB version and program gating function support
- V2.2.4
    - Added:Processor goes to sleep mode whenever possible to reduce power consumption
- V2.2.3
    - Bugfix: Unnecessary erase on empty flash configuration area page
    - Bugfix: Firmware or font update support V2

    - Added: Current date and time can be read via RA command
    - Change: Code efficiency increased
- V2.2.2
    - Bugfix: Heat shock was recorded with improper ID to flash configuration area. Affected versions: V2.2.0-V2.2.1.
- V2.2.1
    - Bugfix: Config word bits StoreBrightness and StorePowerOnOff were not checked. (Always storing changes in brightness and power on/off settings.) Affected version: V2.2.0.
- V2.2.0
    - Added: Configuration data can be saved to Program flash Configuration area
    - Added: Configuration word with many new settings stored in Program flash Configuration area
    - Added: Configuration can be protected by key exchange + change time window mechanism
    - Added: Possibility to record heat shocks to Program flash Configuration area (requires heat sensor)
    - Added: Fonts can be downloaded from the mainboard
- V2.1.0
    - Added: Flash memory sections separated for configuration data and fonts
- V2.0.2
    - Added: Internal function: Conditional initial stream can be switched by DIGTEMP pin
- V2.0.1
    - Added: Fine grain hold time settings with using >=0x80 characters (by *50ms and *500ms)
- V2.0.0
    - Added: Firmware or font update support version V0
- V1.5.7
    - Added: Minimum and maximum spacing font modifiers
    - Added: Led Parameter Setup command (WZ)
    - Bugfix: Set Date/Time data field checking sometimes falsely dropped good frames
- V1.5.6
    - Added: Font print mode modifiers
- V1.5.5

    - Bugfix: Control character combos at the end of a rolling row are getting displayed sometimes
- V1.5.4
    - Bugfix: Response propagation end wasn't sensed
    - Bugfix: Had incorrect hold times when refresh rate changed from usual 1kHz
- V1.5.3
    - Bugfix: Target address selection wasn't properly handled addresses having any of 'A'-'F' hexadecimal digits
- V1.5.2
    - Bugfix: Empty line write program text frame handling causes endless loop then Watchdog restart event
- V1.5.1
    - Added: Keep text colour can be used with variables by colour selection ID ' ' (space)
- V1.5.0
    - Added: Supporting variables
    - BugFix: Rolling row variable content field may step 1 pixel because SpacerMask was not restored on incomplete print.
    - BugFix: Rolling row sometimes wasn't restarted after all pixels at the end rolled off the screen. (Affected firmware versions only since V1.2.0)
- V1.4.2
    - Change: Invalidate ASAP the current screen that has overwritten program text.
- V1.4.1
    - Bugfix: Temperature sensor constants changed, integration time has been increased.
- V1.4.0
    - Added: Supporting short Write Text Program command properties format without sending program constraints (start & end date + time, and Days-of-week mask.)
- V1.3.1
    - Bugfix: RS485 response was missing because of too early switching off of driver chip. (Bug appeared in V1.3.0 only.)
- V1.3.0
    - Change: Reworked communications to work on both 1 and 2 stop bits.
    - Bugfix: Long frame responses were not fully compatible
- V1.2.0c (pre release!)
    - Added: Support of Graphics (loaded with color extended V2006 protocol)

  - Added: Single screen/single program - roll can run forever without interruption (restart)
  - Bugfix: Frames above the size of 255 bytes was rejected
- V1.1.8
  - Bugfix: Default communication with 2 stop bits not 1 (9600,N,8,2)
  - Added: Support for extreme screen refresh lengths (>512 pixels) via c_RowBatches
- V1.1.7
  - Added: Support of various "rainbow" color selections
  - Added: Support of "custom date/time" of the 2007 protocol
  - Added: RGB color selection by protocol V2007
  - BugFix: Skip printing of characters where the x position is less than 0 (Possible only with variables, date/time, etc.)
- V1.1.6
  - Added: Automatic seasonal time change (with RTC always being in wintertime)
  - Added: Handle settings of Base vs. Expand mode and Play All vs. by Time
  - Added: Support of using program start and end date+time as composite constraints when DoW constraint value >=80h or ==00h
- V1.1.5
  - Bugfix: After TAB positioning collision was still checked - contprint now gets set to false after TAB
- V1.1.4
  - Bugfix: Empty program caused trap - Empty program wasn't always invalidated properly
  - Bugfix: Communication propagation missed EOT because bad checksum calculation of uninterested frames
- V1.1.3
  - Added: NextScreenProgram (index) allows post-invalidation of the already prepared next screen
  - Bugfix: CopyRollingRows boundary check was below screen height causing false row shiftings
  - Bugfix: CopySteadyRows boundary check was below screen height causing false row copyings
  - Bugfix: ClearEmptyRows boundary check was below screen height causing false row clearing

- V1.1.2
  - Bugfix: GetLineActualWidth was miscalculated for double strike fonts. (Roll mode sometimes wasn't selected for a line longer than display width - such error have happened only if a double strike font was used the line.)
  - Bugfix: Starting colour after line breaks was sometimes incorrect
  - Change: Last empty line gets removed from programs
  - Added: Smart half screen pull-down - for half display height sized line starting at half of the display
  - Added: Exact screen height font sets
- V1.1.1
  - Change: Program storage is static - preparation of flash storage
  - Bugfix: Repeats were not working, programs got repeated only once
- V1.1.0
  - Added: Support for both V2006 & V2007+ text program header (with automatic selection)
  - Added: Support for RTC
  - Added: Initial temperature support with analogue sensors (only)
  - Added: Supporting screen refresh length up to 768 pixels (on DMA channels 0..2)
  - Added: Support for Date, time & day-of-week constraints of programs
  - Added: Support for program repeats value (“circle”)
  - Added: Support for timed brightness setting
  - Added: Support for screen on/off times setting
  - Added: Internal clock get repeatedly synced to RTC (every second if RTC & communications are idle)
  - Bugfix: GetLineActualWidth missed some spacings when line is not text only. This have caused column fall off especially when text was not left aligned
- V1.0.0
  - Initial version with system info + date/time support

8 Using DisplayTool utility

DisplayTool utility has been developed to simplify managing a display. The utility can be used on Windows Console (CMD / Command Prompt).

Parameters supplied will be used to select management tasks or alter the selected action. Parameter data should be written immediately after (without any space between) the selector character!

Parameters can be supplied in any order.

DisplayTool provides three management tasks:

- 'c' – Executing a single command
- 'f' – Sending batch of commands supplied in a file
- 'u' – Update firmware

8.1 Detailed list of parameters and their default values:

Parameter	Default value	Parameter description
c		Selection of single command execution. Command should be written immediately after the selector ('c') character. Special characters that would cause problems in a Command Prompt parameter (' ','<','|','>','&' and characters having ASCII code less than 32) can be replaced by "\xHH" sequence where HH is the hexadecimal value of the desired ASCII code. It is not required to supply anything else from a valid display command frame, but the data payload between the STX and ETX characters. (It is allowed to include the rest of the frame fully or partially with the special characters between the fields provided too.)
f		Sending batch of commands supplied in a file. File name and path should be written immediately after the selector ('f') character. File has to contain commands as binary data. End of command marker is the NUL character. It is not required to supply anything else from a valid display command frame, but the data payload between the STX and ETX characters. (It is allowed to include the rest of the frame fully or partially with the special characters between the fields provided too.) File execution is not interrupted by the fact of not being acknowledged by the display. Always all commands in the file will be sent.
u		Performing firmware update supplied in a file. File name and path should be written immediately after the selector ('u') character. Update will be executed by issuing multiple commands to the selected

Parameter	Default value	Parameter description
		display. Phases of update: **1.** Checking connection to the display **2.** Checking whether the supplied update file is suitable for the selected display **3.** Informing the user about the result, and requesting for continue **4.** Preparing for update process **5.** Sending update blocks **6.** Closing update process
k		Performing key exchanges before processing the requested task. Key value (4 digit hexadecimal number) can be supplied immediately after the selector ('k') character. However this is normally not required! Only needed in the unlikely event of the update process failing partially in the middle of transmission, and the display cannot exit from its boot loader.
d	0	Delay between commands in milliseconds. This is to have gaps between sending the next command after the response receiving phase of the previous command has been ended.
h		Hinted receive mode. Using this parameter speeds up sending many commands by using the fact that known display commands having an expected end result. In this mode receiving the expected end result will quickly end the receiving phase without waiting for another character to arrive.
m	0	Selecting checksum calculation method. This can be used to override the default calculation method valid for the particular command. Some of the method selectors are require that the supported command frames should have checksum field supported too. Method selecting characters: • 0,d,-,s or u: Use source if supplied, calculate if not. • 1, a or +: Use ADD checksum method • 2, x or ^: Use XOR checksum method • 3 or f: Use FFFF checksum • 4 or c: Always calculate
p	COM1	Use the selected COM port. Above COM9 use “\\.\COMxx” to access through COM port number xx.
t	1	Select Communication ID of target display.
r	1	Number of repeats for the batch of commands in a file.
s		Silent update. Update process will not wait for pressing 'Y' key for the user.

Parameter	Default value	Parameter description
v		Verbose mode. Display all details of the communication. Mostly useful only for testing connection reliability or debug communication issues of a firmware.
w		Wait key press before exiting from DisplayTool. This is useful to see the results if not started from the command prompt but from a CMD file.
y	3	Wait amount (in seconds) for the first character to arrive in response phase.
z	200	Wait amount (in milliseconds) for any next character to arrive in response phase after the fist character arrived. Value cannot be more than 900.
?		Print parameter help information.

8.2 Examples:

- DisplayTool uUpdate.fwu

  Firmware updating display #1 attached to COM1 with update file “Update.fwu”.

- DisplayTool uUpdate245.fwu **t23 pCOM5**

  Firmware updating **display #23 attached to COM5** with update file “Update245.fwu”.

- DisplayTool fc:\test500.txt

  Sending batch of commands to display #1 attached to COM1 from command batch file “c:\test500.txt”.

- DisplayTool cWN00

  Deleting all programs of display #1 attached to COM1. (Executing command “WN00”.)

- DisplayTool cA0C12N13\x0FE\x10B1234

  Loading program to location #0 of display #1 attached to COM1. Program details: Effect=Hold ('C'), Fastest effect/roll speed ('1'), 2 seconds of hold time, No running constraints ('N'), 1 repeat, centre align ('3'), Font select: 5x7 normal (“\x0FE”), Color select: red (“\x10B”), text=”1234”.

- DisplayTool cW\x06^2000 k

  Setting configuration word of display #1 attached to COM1. “^2000” means toggling the **checksum test** bit. (k is to make the key exchange sequence before the command - required to make the change at all.)

- DisplayTool cW\x0610080 k

  Setting configuration word of display #1 attached to COM1. “10080” means setting the **RAM mode** bit. (k is to make the key exchange sequence before the command - required to make the change at all.)

- DisplayTool cW\x07=890040 k

  Setting additional configuration “89” to value “0040” of display #1 attached to COM1. “89” is **display width** in pixels, therefore 0040 means 64 pixels. (k is to make the key exchange sequence before the command - required to make the change at all.)

- DisplayTool cW\x07=800541 k

  Setting additional configuration “8005” to value “41” of display #1 attached to COM1. “8005” is auto effect handling/effect enable parameter, therefore 41 means 'A' effect, so this will **turn on effects** and selecting 'A' effect to a program will perform the **random effect** feature. (k is to make the key exchange sequence before the command - required to make the change at all.)

- DisplayTool cW\x07=800604 k

  Setting additional configuration “8006” to value “04” of display #1 attached to COM1. “8006” is the Legacy flags parameter, therefore 04 means **enabling random colour** (auto colour) selection effect when colour selection character 'A' is being used. (k is to make the key exchange sequence before the command - required to make the changes.)

- DisplayTool cRc0

  Reading content of variable #0 from display #1 attached to COM1.