UnicView AD v1.6.0
Development Guide
1.1



Proculus Technologies Limited

2/19/2019

FIRST Components s.r.l. Via Cavour, 69 21020 Buguggiate (VA) Italy Tel. +39 0332 1888005 Fax +39 0332 320856
web: www.first-components.com

Proculus Technologies Limited provides this document to its customers with a product purchase to use in the product operation. This document is copyright protected and any reproduction of the whole or any part of this document is strictly prohibited, except with the written authorization of Proculus Technologies Limited.

The contents of this document are subject to change without notice. All technical information in this document is for reference purposes only. System configurations and specifications in this document supersede all previous information received by the purchaser.

Proculus Technologies Limited makes no representations that this document is complete, accurate or error-free and assumes no responsibility and will not be liable for any errors, omissions, damage or loss that might result from any use of this document, even if the information in the document is followed properly.

This document is not part of any sales contract between Proculus Technologies Limited and a purchaser. This document shall in no way govern or modify any Terms and Conditions of Sale, which Terms and Conditions of Sale shall govern all conflicting information between the two documents.

For Research Use Only. Not for use in diagnostic procedures.

PATENT PENDING

Contents

1 INTRODUCTION 7
1.1 Document Overview 7
1.2 Conventions Used in this Document 7
1.2.1 Notations 7
1.2.2 Information, Caution and Warning Statements 8

2 AD FIRMWARE OVERVIEW 9

3 HARDWARE 10
3.1 LCD 10
3.2 Touch Panel 11
3.3 RTC 11
3.4 Buzzer 11
3.5 Audio Output 11
3.6 USB Connectors 11
3.7 Power 12
3.8 Serial Communication 12

4 AD FIRMWARE STRUCTURE 13
4.1 Color System 13
4.2 Touch Panel Calibration 13
4.3 Processing Core Operation 13
4.4 Memory Spaces 14
4.4.1 System Configuration Space 14
4.4.2 Register Space 15
4.4.3 RAM Space (VP and PP) 16
4.4.4 FLASH Space 17
4.4.5 Trend Curve Buffer Space 19
4.5 File Structure 22

5 SYSTEM CONFIGURATION (LCM CONFIGURATION) 23
5.1 Frame Header (R3, RA) 23
5.2 Baud rate (R1, R5, 59) 23
5.3 Configuration 1 (R2) 24
5.4 Screen Rotation (R4) 25
5.5 Configuration 2 RC 25

5.6 Backlight Automatic Control (R6, R7, R8) 26
5.7 Initial Screen (RD, RE) 26

6 CONTROL REGISTERS 27
6.1 Control Register Table 28
6.2 Software Control Activation 31
6.3 Audio Reproduction 31
6.4 Video Reproduction 32
6.5 Touch Panel Calibration 33
6.6 Configuration Registers Overwriting 34
6.7 RTC Reading and Adjustment 34
6.8 Trend Curve Buffer Clearing 35

7 SERIAL COMMUNICATION PROTOCOL 36
7.1 Introduction 36
7.2 Control Register Commands 37
7.2.1 Write Registers (0x80) 37
7.2.2 Read Registers (0x81) 38
7.3 VP (RAM) Commands 40
7.3.1 Write VPs (0x82) 40
7.3.2 Read VPs (0x83) 41
7.4 Trend Curve Buffer Commands 43
7.4.1 Write Trend Curve Buffer (0x84) 43
7.4.2 Curve Clearing 44
7.5 CRC 45

8 INTERFACE OBJECTS 48
8.1 VP and PP Distribution 48
8.2 Controls 51
8.2.1 Basic Touch 52
8.2.2 Set Value 53
8.2.3 Touch Status 54
8.2.4 Numeric Input 56
8.2.5 Text Input 58
8.2.6 Incremental Input 60
8.2.7 Slider Input 61
8.2.8 RTC Input 62
8.2.9 Popup 63
8.3 Display Variables 64

8.3.1 Dynamic Icon 65
8.3.2 Animated Icon 66
8.3.3 Slider Display 67
8.3.4 Rotating Icon....69
8.3.5 Bitwise Icon....71
8.3.6 Numeric Art 73
8.3.7 Numeric Display....74
8.3.8 Text Display....75
8.3.9 Image Animation 76
8.3.10 Hex Display 77
8.3.11 RTC Display 78
8.3.12 Analog Clock 79
8.3.13 Table Display....80
8.3.14 Trend Curve Display....83
8.3.15 Graphic Primitives Display (GPD)....84
8.3.16 QR Code Display 88

9 AD ASSEMBLY....89

10 MODBUS 90
10.1 Master 90
10.2 Slave....92

1 Introduction

This section contains important information on how to read this document.

1.1 Document Overview

This document provides a general overview of AD firmware for Proculus LCMs, its features and utilities, instructions on how to use it, and descriptions of all functionalities. It assumes the user has basic prior knowledge about microcontroller or computer programming and binary and hexadecimal numeric representations.

If you are a new user of AD LCMs, we recommend reading this document sequentially, from start to end. However, this document's sections are structured to be used as a reference guide, where you can look for specific information on each subject quickly.

LCM Family

This document applies only to the **V Family** of Proculus LCMs.

1.2 Conventions Used in this Document

This section presents the textual conventions and notations used in this document. Knowing these conventions will make it easier to read this document.

1.2.1 Notations

Notation	Description	Examples
(button+…), **([button/button])**	Keyboard or Mouse button or button combination. Buttons inside square brackets represent multiple options.	**(CTRL+A)** = Control + A **(ALT+LMB)** = Alt + Left Mouse Button **(ESC)** = Escape **(SHIFT+[UP/DOWN])** = **(SHIFT+UP)** or **(SHIFT+DOWN)**
0x0000	Number in hexadecimal notation. The number of digit pairs indicates the number of bytes considered.	0x0064, 0x01
0b00000000	Number in binary notation.	0b01001011, 0b11110000
(0,0), [0,0]	**In value range context** – represents a range of values. **Rounded brackets** symbols indicate the value is not included in the range, and **square brackets** indicate the value is included in the range.	(0,3) = Range from 1 to 2. [0,3) = Range from 0 to 2. (0,3] = Range from 1 to 3. [0,3] = Range from 0 to 3.
(0,0)	**In coordinates context** – represents a pair of values, usually coordinates.	(23, 172) – Coordinates on the LCM screen.
"X - Section Title"	Link to a section of this document.	Section "1.2 - Conventions Used in this Document".
Section Title	Link to a section of this document.	Insert a Basic Touch.
<FieldName>FixedText	Represents variable and fixed text fields.	<FileName>_AD.bin = could be Controls_AD.bin, Images_AD.bin, System_AD.bin, etc.

1.2.2 Information, Caution and Warning Statements

This document may contain Information, Caution and Warning statements.

	Info	This is an Information statement. It draws attention to certain key aspects about the current topic.
	Caution	This is a Caution statement. It describes a situation that could potentially damage your software, equipment or cause data loss.
	Warning	This is a Warning statement. It describes a situation that could potentially cause harm or injury to you.

The information in Caution and Warning statements is provided for your protection. Read each Caution and Warning statement carefully.

2 AD Firmware Overview

The **AD (Advanced Design)** firmware for Proculus LCMs is the main solution for Graphical User Interface (GUI) design. It provides a comprehensive collection of tools that makes the process of creating and integrating a GUI in an application very quick and easy.

AD firmware communicates with external devices through a simple 5-Command serial protocol. It also has Modbus Protocol-ready variants, to enable Modbus RTU Master or Slave communication.

There's also a basic version of AD firmware, called **MiniAD**. Some functionalities from AD are removed from MiniAD firmwares, making it adequate for some niche applications of Proculus LCMs.

This documentation primarily addresses the standard AD firmware, and some information may not be applicable to MiniAD firmwares. While differences between AD and MiniAD are often pointed out in the relevant sections, if you're using a MiniAD firmware, please consult the "UnicView MiniAD Development Guide" document for complete details.



Info — Refer to the "**UnicView MiniAD Development Guide**" document for a complete description of the differences between AD and MiniAD firmwares.

3 Hardware

This section describes the hardware (physical) components of a Proculus Liquid Crystal Module (Proculus LCM).

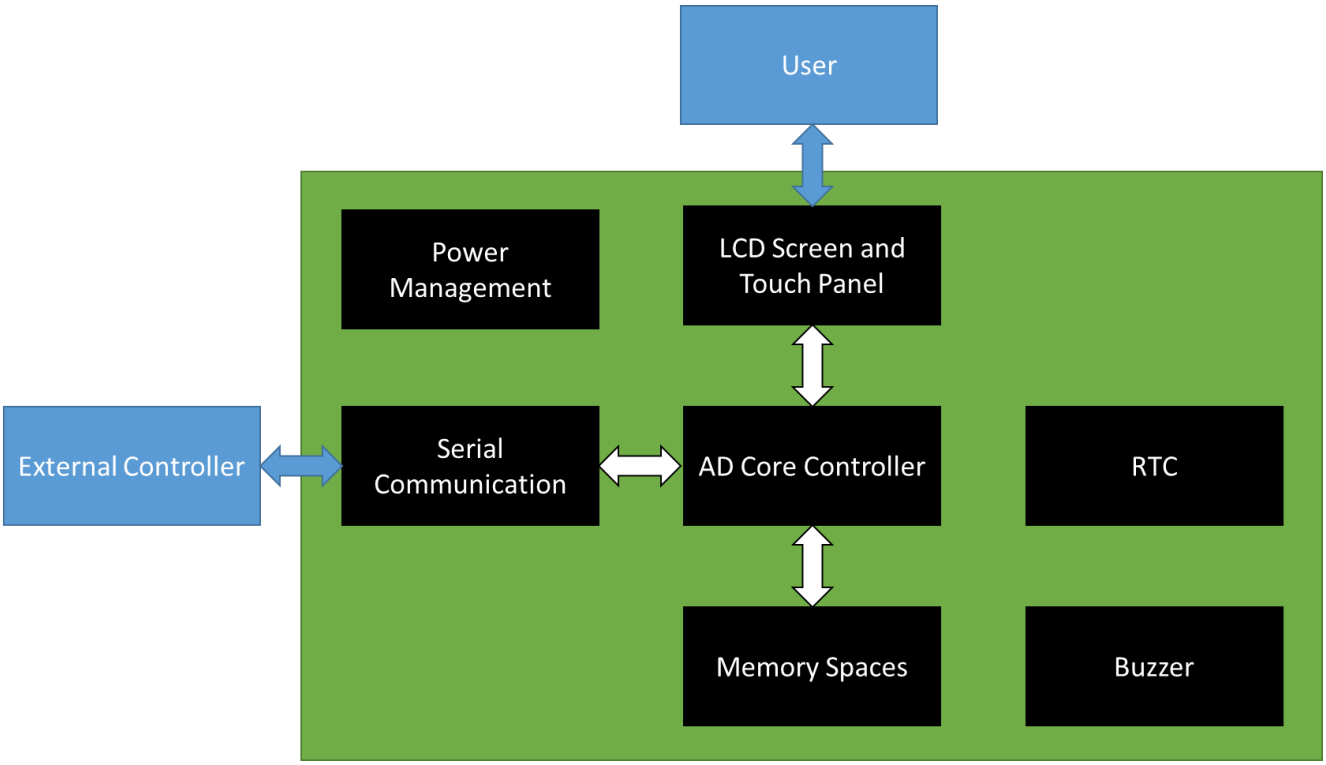
The LCM is composed of three main components:

- Screen and Touch Panel
- Processing Core
- Control PCB (Printed Circuit Board)

The Screen and Touch Panel are the user interaction medium. They are the actual Human-Machine interface elements. The Screen is a LCD display covered by a glass layer, which has a Touch Panel attached to it.

The Processing Core is the central controller, responsible for all the graphical processing and control of all peripherals. The AD firmware runs on the Core.

The Control PCB is the attachment surface for the Screen and the Core. It's also where all the peripherals (RTC, battery holder, Serial Port connectors, etc.) are located.



3.1 LCD

The screen of a LCM is a Thin Film Transistor Liquid Cristal Display (TFT LCD). Its light source is a Light Emitting Diode (LED) strip, positioned right behind the display itself. This LED strip is referred to as **backlight**. The backlight brightness can be adjusted for greater or smaller light levels.

Different LCM models may have different maximum brightness specifications.

3.2 Touch Panel

All Proculus LCMs have touch-screen capabilities. There are three available options:

- **No Touch Panel**: For when there's no need for user direct interaction, and a brighter screen is desirable.
- **Resistive Touch Panel**: Most common option. Offers great sensitivity, while being very stress-resistant. Works with gloves, oily or wet skin, and dirt. Single touch point.
- **Capacitive Touch Panel**: Offers exceptional sensitivity. Higher screen brightness than Resistive. Slightly less stress-resistant. Doesn't work with gloves or wet skin. Capable of five touch points.

3.3 RTC

The built-in Real Time Clock (RTC) provides the ability to display time and date on the LCM without the need for external components.

It has a typical drift of about ±1 second per month.

The RTC requires a battery to retain its value when the LCM is powered-down.

3.4 Buzzer

The built-in buzzer provides audible feedback for the user when they touch a button on the Screen. It has fixed frequency and volume.

The automatic sound feedback can be turned off.

The buzzer can also be activated by Serial Communication.

3.5 Audio Output

Some LCM models have an audio output connector for speakers, to play audio files stored in memory.

3.6 USB Connectors

Proculus LCMs (**V Family**) have both a USB Type A and a USB Mini-B connector. Their functions are:

USB Mini-B

- Project download via USB cable connected to a host PC running UnicView AD software.
- Doesn't supply power to the LCM.

USB Type-A

- Firmware upgrading and project download via USB Flash drives.
- Doesn't supply power to the LCM.



Info — For additional information on the physical interface of Proculus LCMs, please refer to the "Proculus UnicView AD LCMs Connection Guide" document.

3.7 Power

Depending on the LCM model, it may have a built-in voltage regulating system, which grants a wide supply voltage range. On models that don't have regulated supply inputs, the LCM usually accepts a narrower voltage range.

If not enough power is supplied, the LCM won't start, or will flicker (repeatedly turn on then turn off).

3.8 Serial Communication

All Proculus LCMs have one or more connectors for Serial Communication with external controllers. The supported electrical protocols are:

- TTL/CMOS: 0V/3.3V, compatible with 0V/5V devices.
- RS232: Compliant up to -15V/15V.
- RS485: Compliant up to -6V/+6V.

The Serial Port operates at baud rates up to 892900 bps, within a frequency tolerance of ±2%

The packet format is 8N1 (8 data bits, no parity check, 1 stop bit).

4 AD Firmware Structure

4.1 Color System

Most Proculus LCMs use a 16-bit color palette. Colors are coded in the RGB565 format, with 5 bits for the RED channel, 6 bits for the GREEN channel, and 5 bits for the BLUE channel.

	16-bit Color RGB565															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Max. Value	Red 0xF800					Green 0x07E0						Blue 0x001F				

Accepted file format: JPEG, any color specifications, same resolution as the target LCM.

	Info	For best results, use RGB565 color palettes on your graphical design software.

4.2 Touch Panel Calibration

To enter the Touch Panel calibration routine, send the appropriate Serial Command (see sections 6.5 - Touch Panel Calibration and 7.2.1 - Write Registers (0x80) for details).

Touch the three (or five, depending on the LCM Configuration) crosses that will appear on the screen. After the calibration routine is complete, the LCM is reset.



4.3 Processing Core Operation

The AD firmware operates in a cyclic fashion, much like PLCs (Programable Logic Controllers). The binary code of a UnicView AD project is written into the Core memory, and is executed repeatedly in **Operation Cycles**.

The **Operation Cycle Period** is the amount of time that the Core has to execute all of its instructions, and influences the update frequency of the interface. Smaller Operation Cycle Periods increase the perceived "speed" of the interface., while longer Operation Cycles tend to produce a "slower" looking response.

At the end of each Operation Cycle, the screen is refreshed, updating its graphics.

4.4 Memory Spaces

AD LCMs have several memory addressing spaces, each with a specific purpose. This section describes each memory space.

4.4.1 System Configuration Space

The System Configuration space stores basic configuration settings, which determine how the LCM operates.

Characteristics:

- Non-volatile.
- Addressing: 8-bit.
- Address range: [0x00,0x0C] (AD), [0x00,0x0B] (MiniAD).
- Data length: 8-bit.

Each address is referred to as a **Configuration Register**, and is designated as R# (R0, R1, and so on).



This space is usually modified by the CONFIG.txt file, but can be edited via Serial Communication.

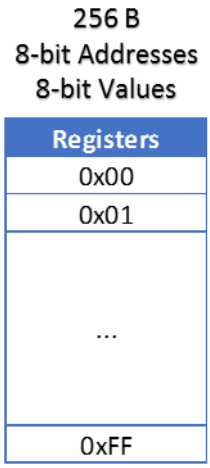
Info	More details in section “5 - System Configuration (LCM Configuration)”.

4.4.2 Register Space

The Control Register space stores run-time configuration settings, and is also used to trigger special procedures, like database access and RTC adjustment.

Characteristics:

- Volatile.
- Addressing: 8-bit.
- Address range: [0x00,0xFF].
- Data length: 8-bit.



This space is usually accessed via Serial Communication.

Info	More details in section “6 - Control Registers”.

4.4.3 RAM Space (VP and PP)

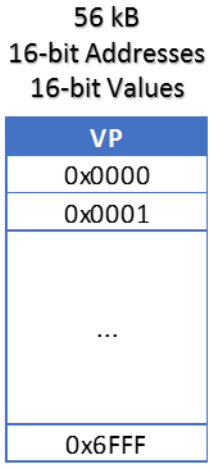
The RAM space, also called VP space, stores all application and user variable data used by Interface Objects. This is the main data space accessed by external controllers.

Each address is referred to as **Variable Pointer** (VP), and the actual data stored in each VP is called **Variable Pointer Content** (VPC), usually denoted in C-Language-Style syntax: *VP.

Parameter Pointer (PP) is the denomination of a VP used to store run-time modifiable parameters of Display Variables.

Characteristics:

- Volatile.
- Addressing: 16-bit.
- Address range: [0x0000,0x6FFF] (AD), [0x0000,0x07FF] (MiniAD).
- Data length: 16-bit.



4.4.4 FLASH Space

The FLASH space stores Icon Libraries, Font Libraries and Configuration Files.

Characteristics:

- Non-volatile.
- Divided into 127 indexes (LibIds), with 256 kB each.
- Addressing (per LibId): 32-bit.
- Address range (per LibId): [0x00000000,0x0001FFFF].
- Data length: 16-bit.

32 MB
32-bit Addresses
16-bit Values

LibId	Sub-Address
0x00	0x00000000
	0x00000001
	...
	0x0001FFFF
0x01	0x00000000
	0x00000001
	...
	0x0001FFFF
...	0x00000000
	0x00000001
	...
	0x0001FFFF
0x7F	0x00000000
	0x00000001
	...
	0x0001FFFF

This space is divided into 127 sub-spaces, each designated by a Library Index, or **LibId**. All Configuration files, Icon and Font Libraries and other standard files are allocated to an individual LibId.



Info If a file occupies more than 256 kB, it will extend to adjacent LibIds, occupying them.

The following table describes the distribution of the FLASH memory, according to file type.

LibId	Content
0	Default Font
11	
12	QR Code
13	Control Configuration
14	Display Variable Configuration
21	
22	RAM Initialization
23	AD Assembly
24	Icons and Fonts
127	

4.4.5 Trend Curve Buffer Space

This is a volatile memory, composed of 8 curve buffer channels. Each channel functions as a FIFO (First-In, First-Out) buffer, that holds the data to be plotted on Trend Curve Displays. Data is **16-bit signed Integer**.

When data is written on a channel, usually via Serial Commands, all visible Trend Curve Displays allocated to that channel will update their plots, showing the data from latest to oldest, right to left. Only the most recent data is shown.

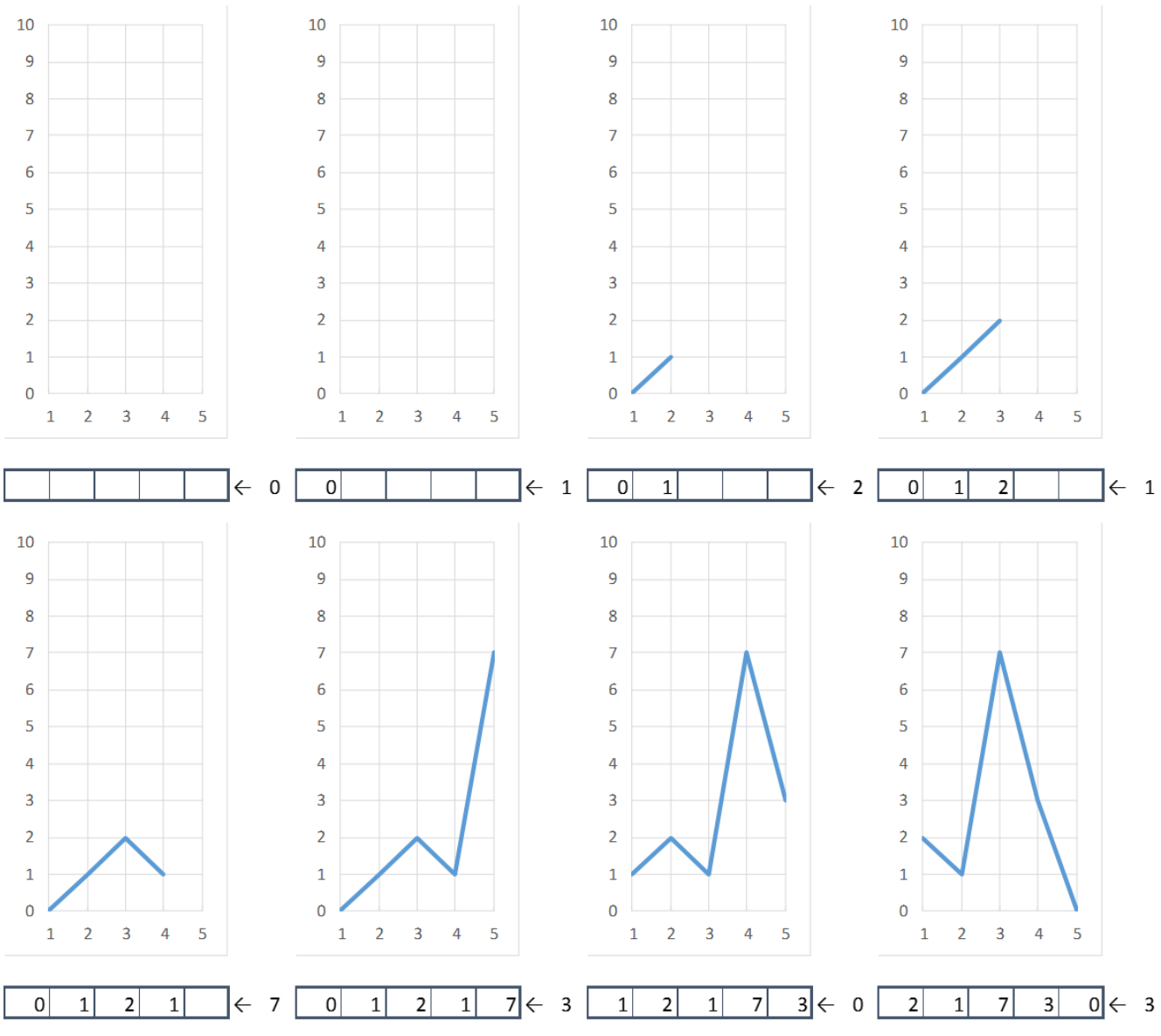
The curves are line graphs, with constant spacing on the X axis.

	Info	For MiniAD, only 2 curve channels are available.

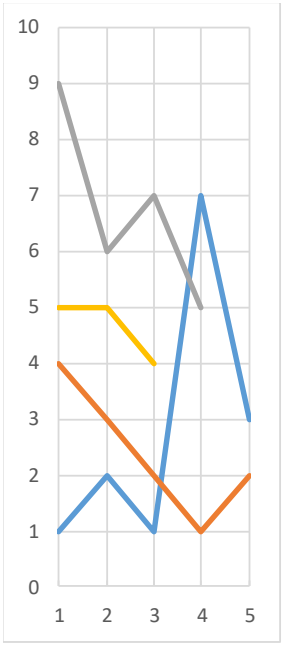
Consider an empty curve buffer, on channel 0, for instance, and a Trend Curve Display.

Initially, the plot is empty. As data is written in the buffer, the plot is continuously updated.

When the plot reaches the right end of the plot area, the plot starts to “move” to the left, making room for new data points.



By using multiple Trend Curve Displays, you can show superimposed plots. As the curve buffers are independent from each other, the data doesn't need to be aligned.



1	2	1	7	3	← C0
4	3	2	1	2	← C1
9	6	7	5		← C2
5	5	4			← C3

The curve buffer channels are numbered from 0 to 7, and are addressed in a one-hot encoding:

Channel	Binary	Hexadecimal
0	00000001	0x01
1	00000010	0x02
2	00000100	0x04
3	00001000	0x08
4	00010000	0x10
5	00100000	0x20
6	01000000	0x40
7	10000000	0x80

To clear a curve buffer channel, write (0x56 + Channel) in the **Control Register** 0xEB. Write 0x55 to clear all channels.

Examples:

- 0x56: Clear channel 0.
- 0x57: Clear channel 1.
- 0x5D: Clear channel 7.
- 0x55: Clear all channels.

	Info	When using MiniAD, write 0x5A to clear channel 0, and 0x5B to clear channel 1.

	Info	Refer to section “7.4 - Trend Curve Buffer” for more information on how to write into the Trend Curve Buffers.

	Info	Refer to section “6 - Control Registers” for more information on Control Registers

4.5 File Structure

The following table describes the file types and naming rules accepted by the AD firmware.

File Type	Naming Rule	Example	Description
Pictures	PicId.jpg	0.jpg	JPEG file, with the same resolution as the target LCM.
Fonts	LibId.dzk	32.dzk	Custom Font Libraries, generated on UnicView AD.
Icon Libraries	LibId.ICO	41. ICO	Icon Libraries, generated on UnicView AD.
Default Font Library	0.hzk	-	Default Font Library, built-in on UnicView AD.
Audio	AudioId.wav	0.wav	WAV file, 32KHz, 16-bit, Mono.
QR Code Library	12.bin		Contains the codification for QR Code.
Control and Display Variable Configuration, RAM Initialization	UnicViewAD.bin	-	Controls Configuration File. Contains all information regarding the Controls and Display Variables on each Screen. Also contains the data that is loaded into the VP Space of the LCM on power-up.
~~**AD Assembly**~~	~~23.BIN~~	~~-~~	~~AD Assembly Binary File.~~
System Configuration (LCM Settings)	CONFIG.txt	-	LCM Basic Configuration Options.

5 System Configuration (LCM Configuration)

The System Configuration is a set of registers that control several aspects of the LCMs operation.

Each **Configuration Register** stores an 8-bit value, and they must be written to a text file, named "CONFIG.txt". Only one Configuration Register is allowed by line. You can insert comments after a semicolon ";" character.

Example of System Configuration file, generated by UnicView AD software:

```
; # UnicView AD System Configuration File
; # WARNING : Do not modify this file manually! This may render the LCM unusable.
R1=07          ; Baudrate: 115200 bps
R5=00          ; Custom baudrate divider, high byte
R9=36          ; Custom baudrate divider, low byte
R3=5A          ; Frame header, high byte
RA=A5          ; Frame header, low byte
R6=3F          ; ON Brightness
R7=00          ; OFF Brightness
R8=05          ; SLEEP Time, in seconds
R2=0F
R4=03          ; Rotation: Clockwise270
RC=12
RD=00
RE=00
```



Caution Avoid manually writing the CONFIG.txt file. Use UnicView AD to generate it.

5.1 Frame Header (R3, RA)

The Serial Communication Frame Header is defined by R3:RA.

The default factory value is R3 = 0x5A, RA = 0xA5.

5.2 Baud rate (R1, R5, 59)

These three Configuration Registers control the Serial Port baud rate.

R1 Value	Baud Rate (bps)
0x00	1200
0x01	2400
0x02	4800
0x03	9600
0x04	19200
0x05	38400
0x06	57600
0x07	115200
0x08	28800
0x09	76800
0x0A	62500
0x0B	125000
0x0C	250000
0x0D	23400
0x0E	345600
0x0F	691200
0x10	892900
0x11	Custom Baud Rate: 6250000/(R5:R9)

5.3 Configuration 1 (R2)

R2 Bit	Binary	Name	Value
7	0x80	Undefined	Write 0.
6	0x40	Undefined	Write 0.
5	0x20	BACKLIGHT	0 = Disable Backlight Automatic Control. 1 = Enabled Backlight Automatic Control.
4	0x10	CRC_CTRL	0 = Disable CRC16 verification on serial port. 1 = Enable CRC16 verification on serial port.
3	0x08	AUTOSEND	0 = Disable Control Auto-Send. 1 = Enable Control Auto-Send.
2	0x04	RAM_INIT	0 = Clear RAM at power-up. 1 = Initialize RAM at power-up according to initial values.
0:1	0x02:0x01	OP_CYCLE	Operation Cycle Period: 00 = 200ms 01 = 160ms 10 = 120ms 11 = 80ms Obs.: For resolutions greater than 1024x768, 120ms or above is recommended.

5.4 Screen Rotation (R4)

This Configuration Register controls the clockwise rotation of the project, relative to an upright LCM facing the user.

R4 Value	Phase
0x00	0°
0x01	90° Clockwise
0x02	180° Clockwise
0x03	270° Clockwise

It can also be understood as a counter-clockwise rotation of the LCM in relation to an upright project:



5.5 Configuration 2 RC

RC Bit	Binary	Name	Value
7	0x80	Undefined	Write 0.
6	0x40	AD_ASM	0 = Disable AD Assembly. 1 = Enable AD Assembly.
5	0x20	BUZZER	0 = Enable Buzzer feedback. 1 = Disable Buzzer feedback.
4	0x10	DVPP	0 = Not used. 1 = 128 Display Variables per Page.
3	0x08	CRC_RESP	0 = Disable CRC16 on LCM responses. 1 = Enable CRC16 on LCM responses (R2.4 must be enabled).
2	0x04	CAL_POINTS	0 = Three-point Touch Panel calibration. 1 = Five-point Touch Panel calibration.
1	0x02	Undefined	Write 1.
0	0x01	Undefined	Write 0.



Info This Configuration Register is not available in MiniAD.

5.6 Backlight Automatic Control (R6, R7, R8)

The backlight brightness can be set to automatically enter a low power mode. This behavior is enabled or disabled through the "**BACKLIGHT**" bit of Configuration Register R2.

	Range	Description
R6	[0x00,0x3F]	Brightness level at normal mode.
R7	[0x00,0x3F]	Brightness level at low-brightness mode.
R8	[0x00,0xFF]	The period, in seconds, after which the LCM will enter low-brightness mode. Counted after the last touch on the Touch Panel.

If Backlight Automatic Control is enabled, after a set amount of time has passed without any touches detected on the Touch Panel, the LCM will "sleep", reducing its backlight level to the desired value.

5.7 Initial Screen (RD, RE)

These Configuration Registers (RD:RE) select the Screen to be shown when the LCM is powered on.

	Range	Description
RD	[0x00,0xFF]	PicId at power-up, high byte.
RE	[0x00,0xFF]	PicId at power-up, low byte.

6 Control Registers

The **Control Registers**, often referred to as **Registers**, reside in a volatile 8-bit memory space, which controls run-time configurations and triggers certain operations. For example, some Registers control the current Screen displayed, or backlight brightness level. Other Registers start data-moving procedures, or reset the LCM.

These Registers can be accessed via Serial Protocol, using the Write Registers (0x80) and Read Registers (0x81) commands.



Caution Wait at least one Operation Cycle Period before writing to the same Register again.

The table on the following section lists all Control Registers. Detailed information on certain operations is provided in the subsequent sections.

6.1 Control Register Table

Address	Name	Length (Bytes)	Description	Range	Read Write
0x00	VERSION_INFO	1	AD Firmware Version. BCD format (0x71 = Version 7.1).	[0x00,0xFF]	R
0x01	CURRENT_BACKLIGHT	1	Backlight brightness level.	[0x00,0x3F]	R/W
0x02	ACTIVATE_BUZZER	1	Activates the buzzer.	[0x00,0xFF] * 10ms	W
0x03 - 0x04	PIC_ID	2	Read: Returns the current Screen (PicId). Write: Jumps to a new Screen.	[0x0000,0xFFFF]	R/W
0x05	TP_FLAG	1	Indicates when a touch has been detected. The coordinates are stored at Registers "TP_COORDINATES". Read: 0x5A = The coordinates have been updated. You must clear this Register, otherwise coordinates are no longer updated. Write: 0x00 = Clears the flag, freeing it to be activated again by the Touch Panel.	-	R/W
0x06	TP_STATUS	1	0x00: The Touch Panel has not been touched yet. 0x01: "Press" state. Indicates that the user has just pressed the Touch Panel. 0x02: "Release" state. Indicates that the user has just released the Touch Panel. This is also the idle state. 0x03: "Hold" state. Indicates that the user is holding the Touch Panel pressed down.	-	R
0x07 - 0x0A	TP_COORDINATES	4	Last touch coordinates captured. 0x07:0x08 = X, 0x09:0x0A = Y. Captures only the first coordinates detected after "TP_FLAG" has been cleared.	-	R
0x0B	TP_ENABLE	1	0x00: Disables the Touch Panel. 0xFF: Enables the Touch Panel.	-	R/W
0x0C - 0x0F	RUN_TIME	4	Counts the time since the LCM was powered-on. BCD Format (0x10002233 = 1000h 22m 33s).	[0000:00:00,9999:59:59]	R
0x10 - 0x1C	R0 - RC_MIRROR	13	Mirrors the R0 to RC SCRs. Writing to these registers doesn't modify the SCRs (use Register OVERWRITE_CONFIG to modify).	[0x00,0xFF]	R/W
0x1D	OVERWRITE_CONFIG	1	0x5A: Save R0 - RC_MIRROR into R0 - RC Registers. 0xA5: Discard changes in R0 - RC_MIRROR.	-	W
0x1E	READ_BACKLIGHT	1	Returns the current backlight brightness level.	[0x00,0x3F]	R
0x1F	OVERWRITE_RTC	1	0x5A: Starts RTC date-time overwriting.	-	W
0x20 - 0x26	RTC_VALUE	7	Date and time from the RTC, in BCD format (YY:MM:DD:WW.HH:MM:SS). Week days = 0 - 7, Sunday - Saturday. Writing to these registers doesn't modify the RTC (use Register OVERWRITE_RTC to modify).	-	R/W
0x27 - 0x3F	-	25	-	-	-

Address	Name	Length (Bytes)	Description	Range	Read Write
0x40 - 0x49	-	10	-	-	-
0x4A - 0x4B	TIMER_0	2	16-bit timer. Counts down to 0. Maximum error is ±4 ms. Read: Returns the current count. Write: Sets the timers to a specific value (starts counting down immediately).	[0x0000,0xFFFF] * 4ms	R/W
0x4C	TIMER_1	1	8-bit timer. Counts down to 0. Maximum error is ±4 ms. Read: Returns the current count. Write: Sets the timers to a specific value (starts counting down immediately).	[0x00,0xFF] * 4ms	R/W
0x4D	TIMER_2	1	8-bit timer. Counts down to 0. Maximum error is ±4 ms. Read: Returns the current count. Write: Sets the timers to a specific value (starts counting down immediately).	[0x00,0xFF] * 4ms	R/W
0x4E	TIMER_3	1	8-bit timer. Counts down to 0. Maximum error is ±4 ms. Read: Returns the current count. Write: Sets the timers to a specific value (starts counting down immediately).	[0x00,0xFF] * 4ms	R/W
0x4F	ACTIVATE_SOFT_CTRL	1	Activates a Software Control. Write the Software Control Code of the Control you want to activate. It simulates a user touch, and the Control must be in the current Screen.	[0x00,0xFF]	W
0x50	AUDIO_PLAY_STOP	1	0x5B: Plays the current audio. 0x5C: Stops the current audio.	-	W
0x51 - 0x52	AUDIO_ID	2	Audio ID to play.	[0x0000,0x0FFF]	W
0x53	OVERWRITE_AUDIO_VOLUME	1	0x5A: Updates the audio volume settings, according to 0x54.	-	W
0x54	AUDIO_VOLUME	1	Audio volume level. Default value = 0x40.	[0x00,0x40]	W
0x55	AUDIO_STATUS	1	0x00: Stopped. 0x01: Playing.	[0x00,0x01]	R

Address	Name	Length (Bytes)	Description	Range	Read Write
0x56 - 0x5F	-	10	-	-	-
0x60	OVERWRITE_VIDEO	1	0x5A: Updates the video reproduction settings, according to 0x61 - 0x67.	-	R/W
0x61	VIDEO_MODE	1	0x00: Play single video in LCM memory. 0x01: Loop single video in LCM memory. 0x02: Repeat all videos in LCM memory. 0x03: Play single video in USB drive. 0x04: Loop single video in USB drive. 0x05: Repeat all videos in USB drive.	[0x00,0x05]	W
0x62 - 0x65	VIDEO_POSITION	4	Top-left coordinates of the video window (X:H, X:L, Y:H, Y:L). 0x00000000: Display centralized.	[0x00000000, 0xFFFFFFFF]	W
0x66 - 0x67	VIDEO_ID	2	Video ID to play, when using mode 0x00 or 0x03 in Register VIDEO_MODE.	[0x0000,0xFFFF]	W
0x68	OVERWRITE_VIDEO_VOLUME	1	0x5A: Updates the video volume settings, according to 0x69.	-	W
0x69	VIDEO_VOLUME	1	Video volume level. Default value = 0x3F.	[0x00, 0x3F]	W
0x6A	VIDEO_PLAY_PAUSE	1	0x5A: Toggles play/pause on the current video.	-	W
0x6B	VIDEO_STOP	1	0x5A: Stops the reproduction of the current video.	-	W
0x6C	VIDEO_NEXT	1	0x5A: Plays next video.	-	W
0x6D	VIDEO_PREVIOUS	1	0x5A: Plays previous video.	-	W
0x6E	VIDEO_STATUS	1	0x00: Idle. 0x01: Playing. 0x02: Paused.	[0x00, 0x02]	R
0x6F - 0xE8		122	-		
0xE9	KEYBOARD_STATUS	1	Read: Returns 0x01 if there's an open keyboard, or 0x00 if there's no keyboard open.	[0x00,0x01]	R
0xEA	ACTIVATE_CALIBRATION	1	0x5A: LCM enters calibration routine.	-	W
0xEB	CLEAR_TREND_CURVE	1	0x55: Clear all channel buffers. 0x56 + #: Clear channel C#, from C0 to C7. Example: 0x5A = Clear C4.	-	W
0xEC - 0xED	-	2	-	-	-
0xEE - 0xEF	ACTIVATE_RESET	2	0x5AA5: Resets the LCM.	-	W
0xF0 - 0xFF	-	16	-	-	-

6.2 Software Control Activation

To activate a **Software Control**, write the **Software Control Code** of the Control in Register 0x4F.

	Info	The Software Control must be in the current PicId displayed.

Serial Command example: Activate Software Control 0x03.

```
5AA5 0480 4F 03
```

6.3 Audio Reproduction

Audio files must be stored in the LCMs internal memory. The files must be named "**#.mp3**" or "**#.wav**", where # denotes any decimal integer number (**without zero padding**) from **0** to **4095**.

File Type	.mp3 or .wav

Audio settings and control is achieved by writing the corresponding **Control Registers**. The next table shows those Registers.

	Info	Only specific LCM models have audio output connections.

Address	Name	Length (Bytes)	Description	Range	Read Write
0x50	AUDIO_PLAY_STOP	1	0x5B: Plays the current audio. 0x5C: Stops the current audio.	-	W
0x51 - 0x52	AUDIO_ID	2	Audio ID to play.	[0x0000, 0x0FFF]	W
0x53	OVERWRITE_AUDIO_VOLUME	1	0x5A: Updates the audio volume settings, according to 0x54.	-	W
0x54	AUDIO_VOLUME	1	Audio volume level. Default value = 0x40.	[0x00, 0x40]	W
0x55	AUDIO_STATUS	1	0x00: Stopped. 0x01: Playing.	[0x00, 0x01]	R

Serial Command Example 1: Play audio file **6.wav** at **100% volume**.

```
5A A5 07 80 50 5B 0006 5A 40
```

Serial Command Example 2: Play audio file **3.wav** at **50% volume**.

```
5A A5 07 80 50 5B 0003 5A 20
```

Serial Command Example 3: Stop audio file **3.wav**.

```
5A A5 05 80 50 5C 0003
```

6.4 Video Reproduction

Video files can be stored either in the LCMs internal memory, or in an external USB flash drive. The files must be named "**#.avi**", where # denotes any decimal integer number (**without zero padding**) from **0** to **65535**.

LCM models with audio output can also play audio from the video.

File Type	.AVI
Video Encoding	MJPEG
Video Resolution	Equal or smaller than the LCM's resolution
Audio Encoding	MP3, 16 kHz sampling



Info When playing video, Display Variables stop refreshing, and Audio reproduction (not from video) is stopped.

Video settings and control is achieved by writing the corresponding **Control Registers**. The next table shows those Registers.

Address	Name	Length (Bytes)	Description	Range	Read Write
0x60	OVERWRITE_VIDEO	1	0x5A: Updates the video reproduction settings, according to 0x61 - 0x67.	-	R/W
0x61	VIDEO_MODE	1	0x00: Play single video in LCM memory. 0x01: Loop single video in LCM memory. 0x02: Repeat all videos in LCM memory. 0x03: Play single video in USB drive. 0x04: Loop single video in USB drive. 0x05: Repeat all videos in USB drive.	[0x00,0x05]	W
0x62 - 0x65	VIDEO_POSITION	4	Top-left coordinates of the video window (X:H, X:L, Y:H, Y:L). 0x00000000: Display centralized.	[0x00000000, 0xFFFFFFFF]	W
0x66 - 0x67	VIDEO_ID	2	Video ID to play, when using mode 0x00 or 0x03 in Register VIDEO_MODE.	[0x0000,0xFFFF]	W
0x68	OVERWRITE_VIDEO_ VOLUME	1	0x5A: Updates the video volume settings, according to 0x69.	-	W
0x69	VIDEO_VOLUME	1	Video volume level. Default value = 0x3F.	[0x00, 0x3F]	W
0x6A	VIDEO_PLAY_PAUSE	1	0x5A: Toggles play/pause on the current video.	-	W
0x6B	VIDEO_STOP	1	0x5A: Stops the reproduction of the current video.	-	W
0x6C	VIDEO_NEXT	1	0x5A: Plays next video.	-	W
0x6D	VIDEO_PREVIOUS	1	0x5A: Plays previous video.	-	W
0x6E	VIDEO_STATUS	1	0x00: Idle. 0x01: Playing. 0x02: Paused.	[0x00, 0x02]	R

Serial Command Example 1: Play vide file **1.avi** at **100% volume**.

```
5AA5 0D80 60 5A 00 0000 0000 0001 5A 3F 5A
```

Serial Command Example 2: Play/pause the current video.

```
5AA5 0380 6A 5A
```

Serial Command Example 3: Stop current video.

```
5AA5 0380 6B 5A
```

6.5 Touch Panel Calibration

To start the touchscreen calibration routine, write the value **0x5A** into the Control Register **0xEA**.

	Info	The current PicId before starting the calibration routine is restored after calibration.

Serial Command example: Start touchscreen calibration routine.

```
5AA5 0480 EA 5A
```

6.6 Configuration Registers Overwriting

To modify the Configuration Registers, write their new values in Registers 0x10 to 0x1C, then write 0x5A in Register 0x1D to save.

6.7 RTC Reading and Adjustment

To read the current date and time, read 7 Registers, starting from register 0x20.

To write new date and time, write 8 Registers, starting from register 0x1F. Register 1F must be written as 0x5A.

Serial Command example: Read the RTC. Current time is 25/10/2012, Thursday, 12:00:01.

```
5A A5 03 81 20 07
```

LCM answer:

```
5A A5 0A 81 20 07 12 10 25 04 12 00 01
```

Serial Command example: Modify the RTC to 25/10/2012, Thursday, 12:00:01.

```
5A A5 0A 80 1F 5A 12 10 25 04 12 00 01
```

6.8 Trend Curve Buffer Clearing

To clear a curve buffer channel, write (0x56 + Channel) in Register 0xEB. Write 0x55 to clear all channels.

5AA5 0380 **EB <55 + CH>**

Serial Command example: Clear channel 0.

5AA5 0380 EB 56

Serial Command example: Clear channel 7.

5AA5 0380 EB 5D

Serial Command example: Clear all channels.

5AA5 0380 EB 55



Info When using MiniAD, write 0x5A to clear channel 0, and 0x5B to clear channel 1.

7 Serial Communication Protocol

7.1 Introduction

	Info	This section uses the following notation: < >: One byte. [ ]: Optional fields. Numbers in Serial Commands are in **hexadecimal format**.

The native Proculus Protocol is composed of 5 commands:

- 0x80: Write Control Registers
- 0x81: Read Control Registers
- 0x82: Write VPs (RAM)
- 0x83: Read VPs (RAM)
- 0x84: Write Trend Curve Buffer

A **Frame** (or packet) structure follows this format:

```
<Frame Header H> <Frame Header L> <Byte Count> <Command> [<Data>...] [<CRC H> <CRC L>]
```

Or, in abbreviated notation:

```
<FHH> <FHL> <BC> <CMD> [<DATA>...] [<CRCH> <CRCL>]
```

- Frame Header: Identifies the start of a new Proculus Protocol packet. Can be used to uniquely identify a LCM on a communication bus. Default value = 0x5AA5.
- Byte Count: Counts the number of bytes in the packet, excluding the Frame Header and this byte, i.e., counts all the bytes starting from the Command byte.
- Command: Defines the Command to be executed.
- Data: Includes addresses, lengths and values.
- CRC: Optional error detection value.

	Info	For simplicity, the CRC field is omitted in the Serial Commands in the following sections.

7.2 Control Register Commands

7.2.1 Write Registers (0x80)

This Command writes one or more Control Registers. You can write multiple Registers at once, if they are sequential.

- **Format**

<FHH> <FHL> <BC> 80 <RG> <VL1> [<VL2> <VL3> ...]

<RG>: Register Address.

<VL#>: Value(s) to write.

- **Examples**

Write the value 3 in Register 0x01 (**same as setting the backlight level**):

5AA5 0380 01 03



Write values on 2 sequential Registers, starting from Register 0x03 (**same as jumping to PicId**):

5AA5 0480 03 0001



	Caution	It is good practice to wait at least one Operation Cycle before writing to the same Control Register again.

7.2.2 Read Registers (0x81)

This Command reads one or more Control Registers. You can read multiple Registers at once, if they are sequential.

- **Format**

<FHH> <FHL> <BC> 81 <RG> <LEN>

<RG>: Register Address.

<LEN>: Number of Registers (bytes) to read.

Answer from LCM:

<FHH> <FHL> <BC> 81 <RG> <LEN> <VL1> [<VL2> <VL3> ...]

<VL#>: Value(s) read;

- **Examples**

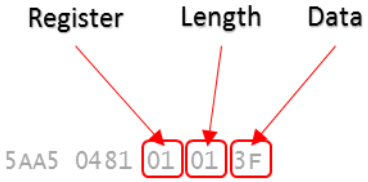
Read the value in Register 0x01 (**same as reading the backlight level**):

5AA5 0381 01 01



Answer from LCM:

5AA5 0481 01 01 3F



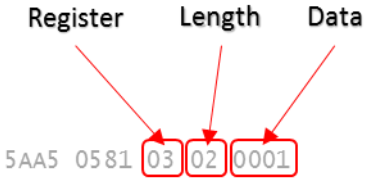
Read values on 2 sequential Registers, starting from Register 0x03 (**same as reading current PicId**):

5AA5 0381 03 02



Answer from LCM:

5AA5 0581 03 02 0001



7.3 VP (RAM) Commands

7.3.1 Write VPs (0x82)

This Command writes one or more VPs. You can write multiple VPS at once, if they are sequential.

- **Format**

<FHH> <FHL> <BC> **82 <VP><VP> <VL1><VL1> [<VL2><VL2> <VL3><VL3> ...]**

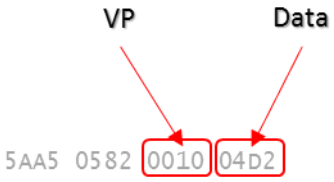
<VP><VP>: RAM Address.

<VL#><VL#>: Value(s) written.

- **Examples**

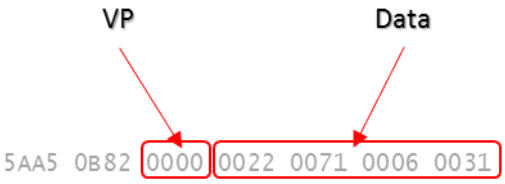
Write the value 1234 in VP 0x0010:

5AA5 0582 0010 04D2



Write values on 4 sequential VPs, starting from VP 0x0000:

5AA5 0B82 0000 0022 0071 0006 0031



7.3.2 Read VPs (0x83)

This Command reads one or more VPs. You can read multiple VPs at once, if they are sequential.

- **Format**

<FHH> <FHL> <BC> **83** **<VP><VP>** **<LEN>**

<VP><VP>: RAM Address.

<LEN>: Number of VPs (words) to read.

Answer from LCM:

<FHH> <FHL> <BC> **83** **<VP><VP>** **<LEN>** **<VL1><VL1>** **[<VL2><VL2>** **<VL3><VL3>** **...]**

<VL#><VL#>: Value(s) read.

- **Examples**

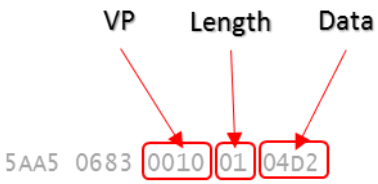
Read the value in VP 0x0010:

5AA5 0483 0010 01



Answer from LCM:

5AA5 0683 0010 01 04D2



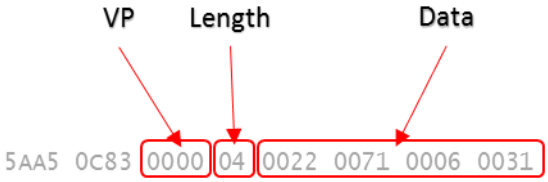
Read values on 4 sequential VPs, starting from VP 0x0000:

5AA5 0483 0000 04



Answer from LCM:

5AA5 0C83 0000 04 0022 0071 0006 0031



7.4 Trend Curve Buffer Commands

This section describes how to write and clear data in the Trend Curve Buffers.

Info	Refer to section “4.4.5 - Trend Curve Buffer Space” for more information.

7.4.1 Write Trend Curve Buffer (0x84)

This Command writes data into one or more Trend Curve Buffer channels.

Info	AD firmware has 8 curve channels. MiniAD firmware has 2 curve channels.

- **Format**

<FHH> <FHL> <BC> 84 <CH> <VL><VL> [<VL><VL> ...]

<CH>: Sum of the binary representation of the channels that will receive the data.

<VL><VL>: Value(s) written. When addressing multiple channels, the data is alternated between channels.

Channel	Binary	Hexadecimal
0	00000001	0x01
1	00000010	0x02
2	00000100	0x04
3	00001000	0x08
4	00010000	0x10
5	00100000	0x20
6	01000000	0x40
7	10000000	0x80

- **Examples**

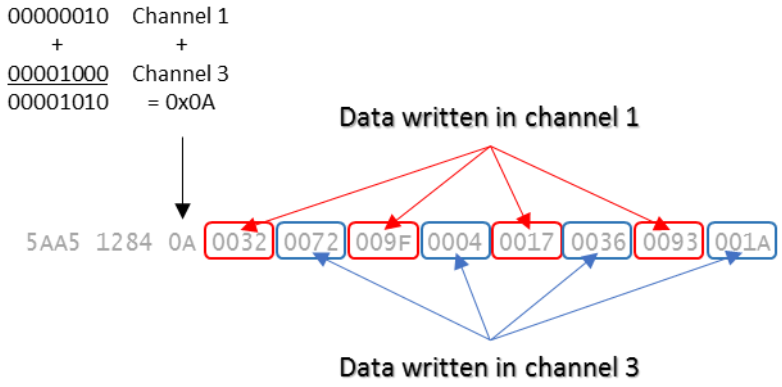
Write 8 values in channel 0:

5AA5 1284 01 0032 0072 009F 0004 0017 0036 0093 001A



Write 4 values in channel 1 and 4 values in channel 3:

5AA5 1284 0A 0032 0072 009F 0004 0017 0036 0093 001A



7.4.2 Curve Clearing

To clear a curve buffer channel, use a Write Registers (0x80) Command to write (0x56 + Channel) in the **Control Register** 0xEB. Write 0x55 to clear all channels.

5AA5 0380 EB <55 + CH>

- **Examples**:

Clear channel 0:

5AA5 0380 EB 56

Clear channel 7:

5AA5 0380 EB 5D

Clear all channels:

5AA5 0380 EB 55

Info	When using MiniAD, write 0x5A to clear channel 0, and 0x5B to clear channel 1.

Info	Refer to section “6 - Control Registers” for more information on Control Registers

7.5 CRC

AD firmware uses Cyclic Redundancy Check (CRC) to verify data integrity during communication. The specific variation used is **CRC-16 Modbus**.



Info — Traditional CRC calculation yields a swapped value. Swap the high and low bytes to get the correct value.

The following pseudo-code explains how to calculate the CRC (already swapped).

```
start
   CRC ← 0xFFFF

   for each byte, do:
   {
        CRC ← CRC xor byte
        repeat
        {
             if CRC.bit0 = 1, then
             {
                  shift CRC right once
                  CRC ← CRC xor 0xA001
             }
             else
                  shift CRC right once
        }
        until 8 right shifts have been performed
   }

   Swap CRC
end
```

Some real code examples (already swapped):

- C Language:

```
// Compute the MODBUS RTU CRC
UInt16 ModRTU_CRC(byte[] buf, int len)
{
   UInt16 crc = 0xFFFF;

   for (int pos = 0; pos < len; pos++)
   {
        // XOR byte into least sig. byte of crc
        crc ^= (UInt16)buf[pos];

        for (int i = 8; i != 0; i--) // Loop over each bit
        {
             if ((crc & 0x0001) != 0) // If the LSB is set
             {
                  crc >>= 1; // Shift right and XOR 0xA001
                  crc ^= 0xA001;
             }
             else // Else LSB is not set
                  crc >>= 1; // Just shift right
        }
   }

   // Swap high and low bytes
   crc = ((crc & 0xFF00) >> 8) | ((crc & 0x00FF) << 8);

   return crc;
}
```

- C# Language:

```
/// <summary>
/// Calculates CRC16 for a input byte array.
/// </summary>
/// <param name="dataArray">Input data.</param>
/// <returns>The calculated CRC16.</returns>
public static ushort CalculateCrc16(byte[] dataArray)
{
    // Compute the MODBUS RTU CRC
    ushort crc = 0xFFFF;

    foreach (var data in dataArray)
    {
        crc ^= data; // XOR byte into least significative byte of CRC

        for (var i = 8; i > 0; i--)
        {
            // Loop over each bit
            if ((crc & 0x0001) != 0)
            {
                // If the LSB is set
                crc >>= 1; // Shift right and XOR 0xA001
                crc ^= 0xA001;
            }
            else // Else LSB is not set
                crc >>= 1; // Just shift right
        }
    }

    var highByte = (crc & 0xFF00);
    var lowByte = crc & 0x00FF;

    crc = (ushort) ((highByte >> 8) | (lowByte << 8));

    return crc;
}
```

8 Interface Objects

On AD LCMs, direct user interaction is provided by **Interface Objects** (or, simply, **Objects**). There are two kinds of Interface Objects:

- Controls – Provide user input.
- Display Variables – Provide visual output to the user.

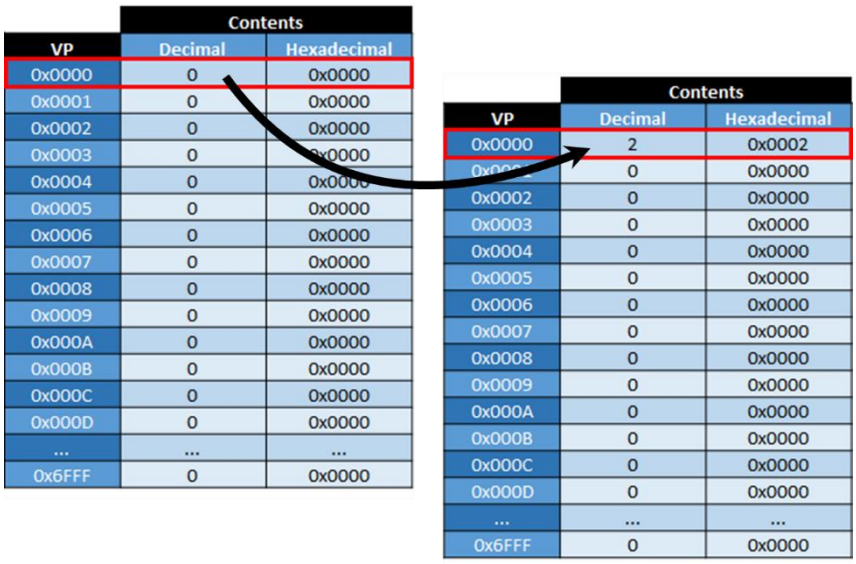
Using both kinds of Interface Objects, you can layout and compose a great human-machine interface (HMI). Each Screen on the LCM has its own set of Objects.

8.1 VP and PP Distribution

Most Interface Objects must be assigned to a **Variable Pointer (VP)**. A VP is an address on the RAM space. Each VP points to a 2-byte (1 word) value.

	Info	The value stored on a VP is called **VPC (Value Pointer Content)**. It is also denoted as ***VP**.

For example, if an Incremental Input is assigned to VP 0x0000, when it is activated, it will increment the value stored at this VP. Assuming the initial value is 0, after two activations (touches on the Touch Panel), the new value stored on VP x0000 is 2.



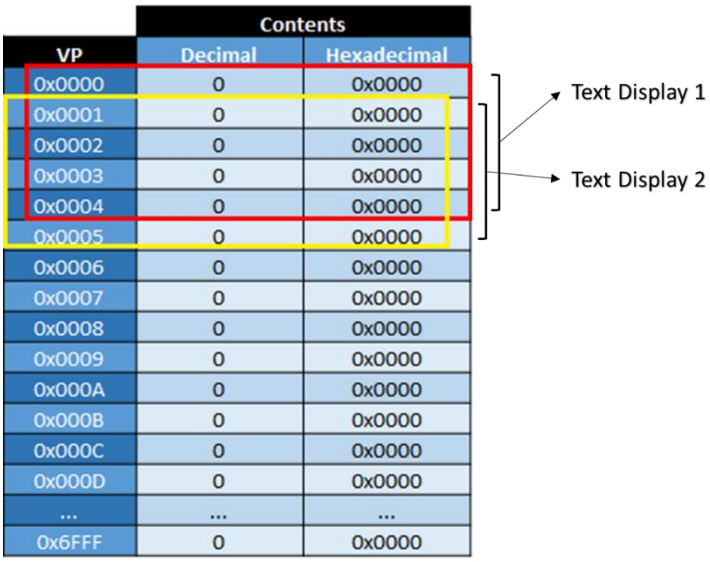
VP	Contents	
	Decimal	Hexadecimal
0x0000	0	0x0000
0x0001	0	0x0000
0x0002	0	0x0000
0x0003	0	0x0000
0x0004	0	0x0000
0x0005	0	0x0000
0x0006	0	0x0000
0x0007	0	0x0000
0x0008	0	0x0000
0x0009	0	0x0000
0x000A	0	0x0000
0x000B	0	0x0000
0x000C	0	0x0000
0x000D	0	0x0000
...	...	...
0x6FFF	0	0x0000

VP	Contents	
	Decimal	Hexadecimal
0x0000	2	0x0002
0x0001	0	0x0000
0x0002	0	0x0000
0x0003	0	0x0000
0x0004	0	0x0000
0x0005	0	0x0000
0x0006	0	0x0000
0x0007	0	0x0000
0x0008	0	0x0000
0x0009	0	0x0000
0x000A	0	0x0000
0x000B	0	0x0000
0x000C	0	0x0000
0x000D	0	0x0000
...	...	...
0x6FFF	0	0x0000

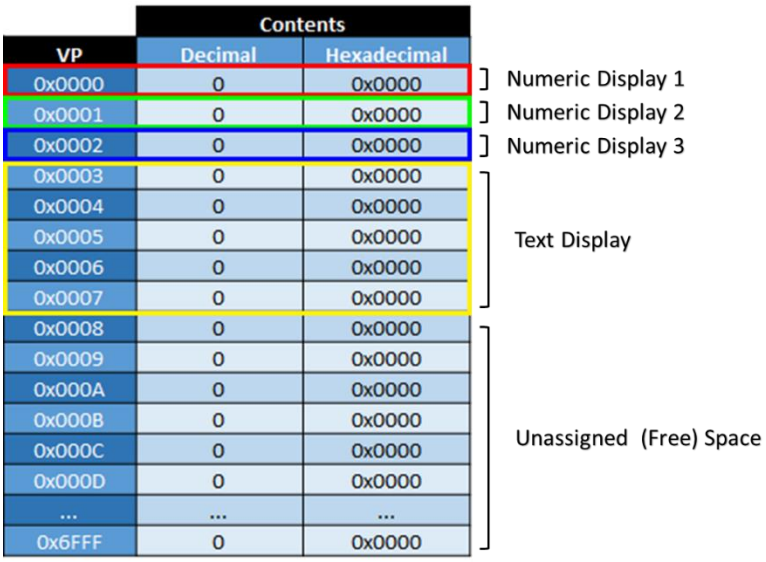
A **Parameter Pointer (PP)** is a VP used to store the parameters of a Display Variable, so that they can be modified by the user at run-time.

	Info	If the PP of a Display Variable is set to the **default value** (-1 or 0xFFFF), the parameters of this Display Variable are fixed, and can only be changed by modifying the project. If you set the PP to a valid RAM address, the parameters are modifiable, and are initialized along the rest of the RAM.

Since there are no restrictions on address assignment, conflicts may show up on a project. For example, usually, you should not assign two Text Displays with Text Lengths greater than 2 to subsequent VPs, because their data will overwrite each other:



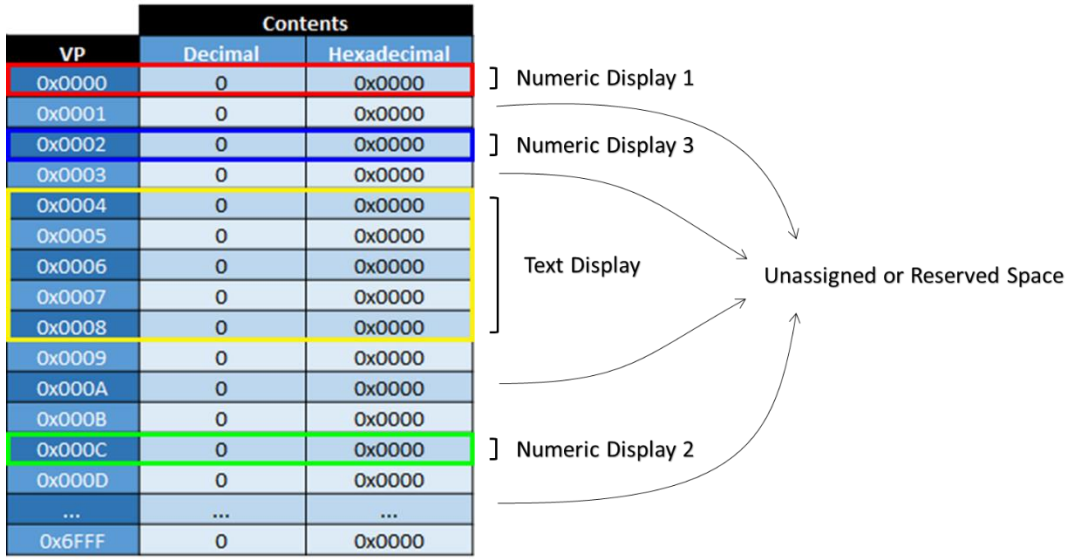
To avoid this problem, always keep in mind how many VPs an Interface Object takes, space them properly:



	Info	While unforeseen address overlapping is a project error, it's sometimes desirable to have overlapping VPs. Multiple Display Variables may be assigned to the same VP, for example, to display the same value in different locations or formats.

To improve scalability and avoid unwanted data overlapping, we recommend planning your project's address allocation beforehand, so that you can reserve addresses between Objects, and give them plenty of space to expand:

VP	Contents	
	Decimal	Hexadecimal
0x0000	0	0x0000
0x0001	0	0x0000
0x0002	0	0x0000
0x0003	0	0x0000
0x0004	0	0x0000
0x0005	0	0x0000
0x0006	0	0x0000
0x0007	0	0x0000
0x0008	0	0x0000
0x0009	0	0x0000
0x000A	0	0x0000
0x000B	0	0x0000
0x000C	0	0x0000
0x000D	0	0x0000
...	...	...
0x6FFF	0	0x0000



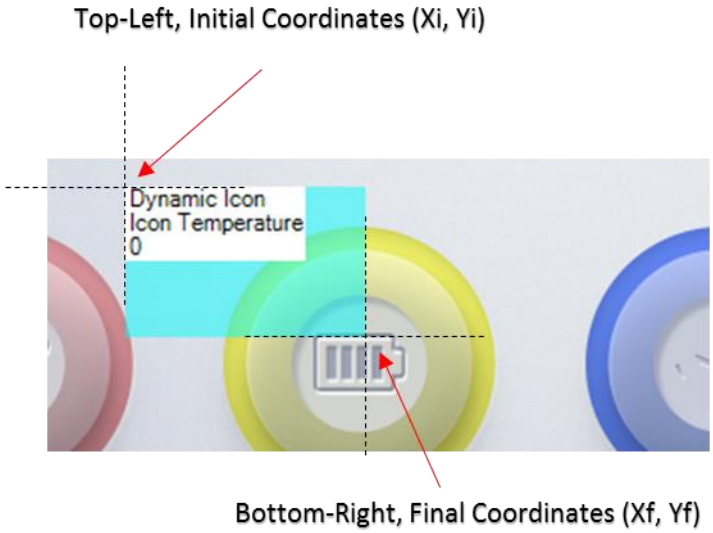
Each project requires a different strategy for addressing rules, but here are some examples:

- All Objects will have even VPs.
- Text Displays will always have VPs 0x3#00 (0x3100, 0x3200, etc.);
- Other Objects will have VPs 0x0##0 (0x0100, 0x0110, 0x0200, etc.), and have PPs 0x5##0 (0x5100, 0x5110, 0x5200, etc.);

By using addressing rules, you can greatly reduce mapping issues and reworks when scaling up or down your projects.

Info	Always verify how many VPs and PPs (if used) an Interface Object requires.

When an Object has an "Area" property, this area is a rectangular region, defined by its top-left coordinate (Initial Coordinates, denoted Xi and Yi) and its bottom-right coordinate (Final Coordinates, denoted Xf and Yf).



8.2 Controls

Controls provide direct user input interaction. They can be considered as buttons. All Controls can be activated by a physical touch from the user, and most of them can be activated via Serial Communication. They are usually employed to modify the contents of the RAM space, although they may be used purely as Serial Communication triggers.

	Info	Controls may be user-activated or software-activated (via Serial Communication). Refer to section “6.2 - Software Control Activation” for details on how to activate a Software Control.

	Caution	Controls can’t have overlapping areas.

All Controls have at least 5 parameters, detailed in the following table:

Definition	Length (bytes)	Description
PicId	2	Picture ID
Area	8	Area of the Control: (Xi, Yi, Xf, Yf). Special Conditions: • Xi = 0xFFFF: This Control is a **Software Control**, i.e., a Control that can be activated via Serial Communication. In this case, Yi.H defines the **Software Control Code**, which is the code that, when written to Register 0x4F, activates this Control. • Xi = 0x5***: This Control plays Audio Segments when activated. In this case, the high nibbles of Xf and Yf define the AudioId (1 byte), and the high nibble of Yi defines the Audio Length. Buzzer feedback is disabled for this Control.
JumpId	2	PicId of the screen to jump to. 0xFF**: Disable jump.
FxId	2	PicId of the screen where the "pressed" effects for this control are. 0xFF**: Disable effect.
Control_Code	2	Operation Code for the Control. High byte defines the operation mode: 0xFF** = Invalid. 0xFE** = **Normal Control**. Auto-Send Data enabled for this Control. 0xFD** = **Normal Control**. Auto-Send Data disabled for this Control. Other Values = **Basic Touch Control**. High byte and Low byte are ASCII Code characters. Low byte defines the Control Type. If High byte is neither 0xFF, 0xFE or 0xFD, low byte is ASCII Code.
Control_Parameters	16 or 32	When Control_Code = 0xFE** or 0xFD**, additional parameters.

8.2.1 Basic Touch

Used to implement basic Screen navigation and to create the keys on keyboards.

Address	Definition	Length (byte)	Description
0x00	PicId	2	ID of the Screen.
0x02	Area	8	Area of the Control: (Xi, Yi, Xf, Yf).
0x0A	JumpId	2	PicId of the screen to jump to. 0xFF**: Disable jump.
0x0C	FxId	2	PicId of the screen where the "pressed" effects for this control are. 0xFF**: Disable effect.
0x0E	Control_Code	2	Two ASCII Code characters, for when this Basic Touch is used as a keyboard key. 0xFF, 0xFE and 0xFD characters are not allowed in the **high byte**.

8.2.2 Set Value

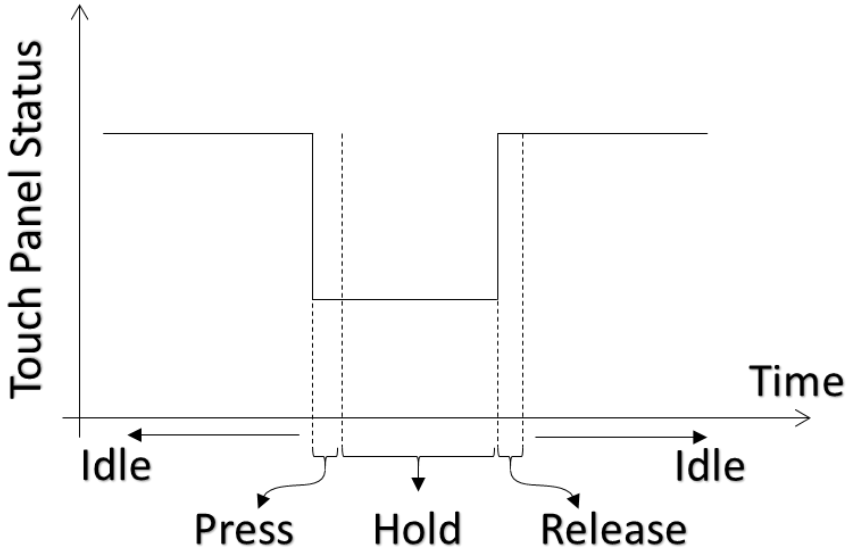
Used to implement a button that writes a value to a VP when pressed, and to signal events to an external controller.

Address	Definition	Length (byte)	Description
0x00	PicId	2	ID of the Screen.
0x02	Area	8	Area of the Control: (Xi, Yi, Xf, Yf).
0x0A	JumpId	2	PicId of the screen to jump to. 0xFF**: Disable jump.
0x0C	FxId	2	PicId of the screen where the "pressed" effects for this control are. 0xFF**: Disable effect.
0x0E	Control_Code	2	0xFE05 or 0xFD05
0x10	0xFE	1	0xFE
0x11	VP	2	Variable Pointer.
0x13	VP_Mode	1	Value Memory Size. 0x00: 16-bit Integer. 0x01: 8-bit Unsigned Integer in High Byte of the VP. 0x02: 8-bit Unsigned Integer in Low Byte of the VP. 0x10-0x1F: 1-bit value. 0x10 corresponds to VP.0, 0x1F corresponds to VP.F. In 8-bit mode, uses the Low Byte in return value. In 1-bit mode, uses the LSB.
0x14	Return_Value	2	Return key code.
0x16	0x00	10	0x00.

8.2.3 Touch Status

Used to implement push-buttons. Can be configured to execute different operations during its three states:

- Pressed
- Held down
- Released



Address	Definition	Length (byte)	Description
0x00	PicId	2	ID of the Screen.
0x02	Area	8	Area of the Control: (Xi, Yi, Xf, Yf).
0x0A	JumpId	2	PicId of the screen to jump to. 0xFF**: Disable jump.
0x0C	FxId	2	PicId of the screen where the "pressed" effects for this control are. 0xFF**: Disable effect.
0x0E	Control_Code	2	0xFE08 or 0xFD08
0x10	0xFE	1	0xFE
0x11	OP_Mode_Press	1	Operation mode for when this control is pressed. 0x00: Does nothing. 0x01: Copies *[Source Address (Press)] to RAM at [Target Address (Press)]. 0x02: Sends *[Source Address (Press)] to the Serial Port. 0x03: Copies *[Source Address (Press)] to Control Register Memory at [Target Address (Press)].
0x12	Source_Address_Press	2	Source Address for copying (Press).
0x14	Target_Address_Press	2	Target Address for copying (Press).
0x16	0x00	1	0x00
0x17	Length_Press	1	Number of bytes to copy (Press). **When in Operation Mode 0x01, this number must be even.**
0x18	0xFE	1	0xFE
0x19	OP_Mode_Hold	1	Operation mode for when this control is held down. 0x00: Does nothing. 0x01: Copies *[Source Address (Hold)] to RAM at [Target Address (Hold)]. 0x02: Sends *[Source Address (Hold)] to the Serial Port. 0x03: Copies *[Source Address (Hold)] to Control Register Memory at [Target Address (Hold)].

0x1A	Source_Address_Hold	2	Source Address for copying (Hold).
0x1C	Target_Address_Hold	2	Target Address for copying (Hold).
0x1E	0x00	1	0x00
0x1F	Length_Hold	1	Number of bytes to copy (Hold). **When in Operation Mode 0x01, this number must be even.**
0x20	0xFE	1	0xFE
0x21	OP_Mode_Release	1	Operation mode for when this control is released. 0x00: Does nothing. 0x01: Copies *[Source Address (Release)] to RAM at [Target Address (Release)]. 0x02: Sends *[Source Address (Release)] to the Serial Port. 0x03: Copies *[Source Address (Release)] to Control Register Memory at [Target Address (Release)].
0x22	Source_Address_Release	2	Source Address for copying (Release).
0x24	Target_Address_Release	2	Target Address for copying (Release).
0x26	0x00	1	0x00
0x27	Length_Release	1	Number of bytes to copy (Release). **When in Operation Mode 0x01, this number must be even.**
0x28	0x00	8	0x00

8.2.4 Numeric Input

Opens a keyboard for numeric values input. It uses fixed-point integer values.

Address	Definition	Length (byte)	Description
0x00	PicId	2	ID of the Screen.
0x02	Area	8	Area of the Control: (Xi, Yi, Xf, Yf).
0x0A	JumpId	2	PicId of the screen to jump to. 0xFF**: Disable jump.
0x0C	FxId	2	PicId of the screen where the "pressed" effects for this control are. 0xFF**: Disable effect.
0x0E	Control_Code	2	0xFE00 or 0xFD00
0x10	0xFE	1	0xFE
0x11	VP	2	Variable Pointer.
0x13	VP_Mode	1	Value Memory Size. 0x00: 16-bit Integer. 0x01: 32-bit Integer. 0x02: 8-bit Unsigned Integer in High Byte of the VP. 0x03: 8-bit Unsigned Integer in Low Byte of the VP. 0x04: 64-bit Integer. In 8-bit mode, uses the Low Byte in return value. In 1-bit mode, uses the LSB.
0x14	Integer_Digits	1	Number of digits to the left of the decimal separator.
0x15	Decimal_Digits	1	Number of digits to the right of the decimal separator.
0x16	Cursor_Origin	4	Coordinates of the top-left corner of the cursor's origin.
0x1A	Font_Color	2	Color of the font.
0x1C	LibId	1	Index in the FLASH memory of the Font to use.
0x1D	Font_Width	1	Font width, in pixels. Range: [0x04,0xFF]
0x1E	Cursor_Color	1	Color of the input cursor. 0x00: Black Other Values: White.
0x1F	Show_Characters	1	Sets whether the characters are to be displayed normally. 0x00: Characters displayed as asterisks (*). Other Values: Normal display.
0x20	0xFE	1	0xFE
0x21	External_Keyboard	1	Indicates if the keyboard image is in the same Screen as this Control. 0x00: Keyboard on current Screen. Other Values: Keyboard on another Screen.
0x22	Source_PicId	2	PicId of the Screen used as image source for this control. It's the PicId where the keyboard image is. Not used if "External_Keyboard" = 0x00.
0x24	Source_Area	8	Area of the keyboard: (Xi, Yi, Xf, Yf). Not used if "External_Keyboard" = 0x00.
0x2C	Target_Origin	4	Top-left coordinates of the pasting area of the keyboard. Not used if "External_Keyboard" = 0x00.
0x30	0xFE	1	0xFE
0x31	Limit_Value_Range	1	Limit the control's accepted input values according to the range set. 0xFF: Limit accepted values to the range set. Other Values: Any value is accepted.
0x32	Min_Value	4	Minimum value accepted by the control.
0x36	Max_Value	4	Maximum value accepted by the control.
0x3A	0x00	6	0x00.

The keys on the keyboard must be designed with Basic Touch Controls. Each Basic Touch's "**TP_Code**" must be assigned to a code:

Key	Definition	Description
0x00F0	Cancel	Cancels the input, doesn't change any data.
0x00F1	Return	Completes the input, writing the value to the VP.
0x00F2	Backspace	Deletes the right-most character.
0x0030 - 0x0039	Digits	ASCII Code of a digit (0 - 9).
0x002D	+/-	Inverts the sign of the value (positive or negative).
0x002E	.	Inserts a decimal separator (dot).



Caution The sum of **Integer** and **Decimal** digits should not exceed 20.

8.2.5 Text Input

Opens a keyboard for alphanumeric (text) values input. It uses terminator characters (0xFF) to signal end of text.

Address	Definition	Length (byte)	Description
0x00	PicId	2	ID of the Screen.
0x02	Area	8	Area of the Control: (Xi, Yi, Xf, Yf).
0x0A	JumpId	2	PicId of the screen to jump to. 0xFF**: Disable jump.
0x0C	FxId	2	PicId of the screen where the "pressed" effects for this control are. 0xFF**: Disable effect.
0x0E	Control_Code	2	0xFE06 or 0xFD06
0x10	0xFE	1	0xFE
0x11	VP	2	Variable Pointer.
0x13	Text_Max_Length	1	Maximum text length, in words (two characters for each word). Range: [0x01,0x7B].
0x14	Scan_Mode	1	Input mode. 0x00: re-input, 0x01: modify existing text.
0x15	LibId	1	Index in the FLASH memory of the Font to use.
0x16	Font_Width	1	Font width, in pixels. Range: [0x04,0xFF]
0x17	Font_Height	1	Font height, in pixels. When using "**LibId**" = 0x00, must be twice the Width.
0x18	Cursor_Color	1	Color of the input cursor. 0x00: Black Other Values: White.
0x19	Color	2	Text color.
0x1B	Text_Area_Initial	4	Top-left coordinates of the area where the text will be displayed: (Xi, Yi).
0x1F	Update_Input_Status	1	Enables or disables input status updating. When enabled, the address [VP-1] is used to indicate the input status of this control: [VP-1].High = Has the value 0x5A when the keyboard is closed. Other values indicate the keyboard is open. [VP-1].Low = Last successful input length, in bytes. 0x55: Enable Input Status Update. 0x00: Disable Input Status Update.
0x20	0xFE	1	0xFE
0x21	Text_Area_Final	4	Bottom-right coordinates of the area where the text will be displayed: (Xf, Yf).
0x25	External_Keyboard	1	Indicates if the keyboard image is in the same Screen as this Control. 0x00: Keyboard on current Screen. Other Values: Keyboard on another Screen.
0x26	Source_PicId	2	PicId of the Screen used as image source for this control. It's the PicId where the keyboard image is. Not used if "External_Keyboard" = 0x00.
0x28	Source_Area	8	Area of the keyboard: (Xi, Yi, Xf, Yf). Not used if "External_Keyboard" = 0x00.
0x30	0xFE	1	0xFE
0x31	Target_Origin	4	Top-left coordinates of the pasting area of the keyboard. Not used if "External_Keyboard" = 0x00.
0x35	Hide_Characters	1	Sets whether the characters are to be masked by asterisks. 0x00: Normal display. 0x01: Characters displayed as asterisks (*).
0x36	0x00	10	0x00.

When you close the keyboard (if not cancelled), it automatically appends terminator bytes (0xFF) to the end of the text, if any unused addresses remain. One terminator byte is used if there's an odd number of characters, and two terminator bytes (one VP) are used if there's an even number of characters (smaller than the maximum length).

The keys on the keyboard must be designed with Basic Touch Controls. Each Basic Touch's "**TP_Code**" must be assigned to a code. For digit keys, use ASCII codes:

Code	Lower	Upper	Code	Lower	Lower	Code	Lower	Lower	Code	Lower	Lower
0x7E60	`	~	0x5171	q	Q	0x4161	a	A	0x5A7A	z	Z
0x2131	1	!	0x5777	w	W	0x5373	s	S	0x5878	x	X
0x4032	2	@	0x4565	e	E	0x4464	d	D	0x4363	c	C
0x2333	3	#	0x5272	r	R	0x4666	f	F	0x5676	v	V
0x2434	4	$	0x5474	t	T	0x4767	g	G	0x4262	b	B
0x2535	5	%	0x5979	y	Y	0x4868	h	H	0x4E6E	n	N
0x5E36	6	^	0x5575	u	U	0x4A6A	j	J	0x4D6D	m	M
0x2637	7	&	0x4969	i	I	0x4B6B	k	K	0x3C2C	,	<
0x2A38	8	*	0x4F6F	o	O	0x4C6C	l	L	0x3E2E	.	>
0x2839	9	(	0x5070	p	P	0x3A3B	;	:	0x3F2F	/	?
0x2930	0	)	0x7B5B	[	{	0x2227	'	"	0x2020	Space	Space
0x5F2D	-	_	0x7D5D	]	}	0x0D0D	New Line	New Line			
0x2B3D	=	+	0x7C5C	\	|						

For control keys, use these codes:

Code	Definition	Description
0x00F0	Cancel	Cancels the input, doesn't change any data.
0x00F1	Return	Completes the input, writing the value to the VP.
0x00F2	Backspace	Deletes the character to the left of the cursor.
0x00F3	Delete	Deletes the character to the right of the cursor.
0x00F4	CapsLock	Caps lock. Button Effect must be enabled for this function.
0x00F7	Left	Moves the cursor left.
0x00F8	Right	Moves the cursor right.



Info

Keep "**Text_Max_Length**" and "**Update_Input_Status**" in mind when calculating the total VP range you should reserve to a Text Input.

8.2.6 Incremental Input

Used to implement a button that increments the content of a VP.

Address	Definition	Length (byte)	Description
0x00	PicId	2	ID of the Screen.
0x02	Area	8	Area of the Control: (Xi, Yi, Xf, Yf).
0x0A	JumpId	2	PicId of the screen to jump to. 0xFF**: Disable jump.
0x0C	FxId	2	PicId of the screen where the "pressed" effects for this control are. 0xFF**: Disable effect.
0x0E	Control_Code	2	0xFE02 or 0xFD02
0x10	0xFE	1	0xFE
0x11	VP	2	Variable Pointer.
0x13	VP_Mode	1	Value Memory Size. 0x00: 16-bit Integer. 0x01: 8-bit Unsigned Integer in High Byte of the VP. 0x02: 8-bit Unsigned Integer in Low Byte of the VP. 0x10-0x1F: 1-bit value. 0x10 corresponds to VP.0, 0x1F corresponds to VP.F. In 8-bit mode, uses the Low Byte in return value. In 1-bit mode, uses the LSB.
0x14	Increment_Sign	1	Sign of the increment. 0x00: --, others: ++.
0x15	Loop	1	0x00: Disabled. The value stops changing when max. or min. values are reached. Other Values: Enabled. The value loops around the range when it reaches max. or min. values.
0x16	Increment_Value	2	Step size: 0x0000-0x7FFF. **Must be 0 or 1 when "VP_Mode" is in 1-bit mode.**
0x18	Min_Value	2	Minimum value accepted by the control.
0x1A	Max_Value	2	Maximum value accepted by the control.
0x1C	Continuous_Increment	1	0x00: Continuous. The value is changed while the user holds it. 0x01: One-Step. The value is changed once per touch.
0x1D	0x00	3	0x00.

8.2.7 Slider Input

Used to implement a sliding button that dynamically changes the content of a VP.

Info Slider Input doesn't support the Software Control feature.

Address	Definition	Length (byte)	Description
0x00	PicId	2	ID of the Screen.
0x02	Area	8	Area of the Control: (Xi, Yi, Xf, Yf).
0x0A	JumpId	2	PicId of the screen to jump to. 0xFF**: Disable jump.
0x0C	FxId	2	PicId of the screen where the "pressed" effects for this control are. 0xFF**: Disable effect.
0x0E	Control_Code	2	0xFE03
0x10	0xFE	1	0xFE
0x11	VP	2	Variable Pointer.
0x13	VP_Mode_Orientation	1	High nibble defines the VP_Mode: 0x0*: 16-bit Integer. 0x1*: 8-bit Unsigned Integer in High Byte of the VP. 0x2*: 8-bit Unsigned Integer in Low Byte of the VP. Low nibble defines the slider orientation: 0x*0: Horizontal. 0x*1: Vertical.
0x14	Sliding_Area	8	Sliding Area. Should be equal to "**Area**".
0x1C	Min_Value	2	Minimum value accepted by the control.
0x1E	Max_Value	2	Maximum value accepted by the control.

Info A Slider Input activates after 0.5 seconds of being held pressed.

8.2.8 RTC Input

Opens a keyboard to modify the current date and time.

Address	Definition	Length (byte)	Description
0x00	PicId	2	ID of the Screen.
0x02	Area	8	Area of the Control: (Xi, Yi, Xf, Yf).
0x0A	JumpId	2	PicId of the screen to jump to. 0xFF**: Disable jump.
0x0C	FxId	2	PicId of the screen where the "pressed" effects for this control are. 0xFF**: Disable effect.
0x0E	Control_Code	2	0xFE04 or 0xFD04
0x10	0xFE	1	0xFE
0x11	0x00	3	0x00.
0x14	(x, y)	4	Position of cursor, right alignment.
0x18	Color	2	Font color.
0x1A	LibId	1	Index in the FLASH memory of the Font to use.
0x1B	Font_Width	1	Font width, in pixels. Range: [0x04,0xFF]
0x1C	Cursor_Color	1	Color of the input cursor. 0x00: Black Other Values: White.
0x1D	External_Keyboard	1	Indicates if the keyboard image is in the same Screen as this Control. 0x00: Keyboard on current Screen. Other Values: Keyboard on another Screen.
0x1E	Source_PicId	2	PicId of the Screen used as image source for this control. It's the PicId where the keyboard image is. Not used if "External_Keyboard" = 0x00.
0x20	0xFE	1	0xFE
0x21	Source_Area	8	Area of the keyboard: (Xi, Yi, Xf, Yf). Not used if "External_Keyboard" = 0x00.
0x29	Target_Origin	4	Top-left coordinates of the pasting area of the keyboard: (Xi, Yi). Not used if "External_Keyboard" = 0x00.
0x2D	0x00	3	0x00.



Info It's not possible to adjust date and time separately using RTC Input.

8.2.9 Popup

Opens a keyboard in a popup window-style.

Address	Definition	Length (byte)	Description
0x00	PicId	2	ID of the Screen.
0x02	Area	8	Area of the Control: (Xi, Yi, Xf, Yf).
0x0A	JumpId	2	PicId of the screen to jump to. 0xFF**: Disable jump.
0x0C	FxId	2	PicId of the screen where the "pressed" effects for this control are. 0xFF**: Disable effect.
0x0E	Control_Code	2	0xFE01 or 0xFD01
0x10	0xFE	1	0xFE
0x11	VP	2	Variable Pointer.
0x13	VP_Mode	1	Value Memory Size. 0x00: 16-bit Integer. 0x01: 8-bit Unsigned Integer in High Byte of the VP. 0x02: 8-bit Unsigned Integer in Low Byte of the VP. 0x10-0x1F: 1-bit value. 0x10 corresponds to VP.0, 0x1F corresponds to VP.F. In 8-bit mode, uses the Low Byte in return value. In 1-bit mode, uses the LSB.
0x14	Source_PicId	2	PicId of the Screen used as image source for this control. It's the PicId where the popup image is.
0x16	Source_Area	8	Area of the keyboard: (Xi, Yi, Xf, Yf).
0x1E	Target_Origin_X	2	Left coordinate of the pasting area of the keyboard.
0x20	0xFE	1	0xFE
0x21	Target_Origin_Y	2	Top coordinate of the pasting area of the keyboard.
0x23	0x00	13	0x00.

When creating the Basic Touch buttons for a Popup, valid ASCII Codes range from **0x00 to 0xFE**, while **0xFF** cancels the Popup (close it without changing the value in the VP).

8.3 Display Variables

Display Variables provide visual interaction to the users. They function as numeric, textual and graphic indicators. Display Variables are always associated to a memory address, and show the contents of such address in some human-readable form.



Caution There is a maximum number of Display Variables per Screen. This number (64 or 128) is set on the Configuration 1 (R2 Register.

While **Controls** can be placed in any PicId, Display Variables have a maximum allowed PicId, which depends on the number of Display Variables per Page configuration:

Display Variables per Page	PicId range that accepts Display Variables
64	[0,1023]
128	[0,511]



Caution Make sure to reserve enough VPs for each Display Variable. Consider their data sizes and their PP lengths (when PPs are used).

8.3.1 Dynamic Icon

Used to show an Icon from an Icon Library. The current Icon is determined by the value of the VP.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A00	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x0008	2	
0x06	0x00	VP	2	Variable Pointer.
0x08	0x01	Coordinates	4	Top-left coordinates of the Icons to display: (Xi, Yi).
0x0C	0x03	Min_Value	2	Minimum value.
0x0E	0x04	Max_Value	2	Maximum value.
0x10	0x05	Min_Icon	2	Icon associated to the Min_Value.
0x12	0x06	Max_Icon	2	Icon associated to the Max_Value.
0x14	0x07.H	LibId	1	Index in the FLASH memory of the Icon Library to use.
0x15	0x07.L	Transparency	1	0x00: Transparent background. Other Values: Opaque background.
PP Length:	**8 VPs**			



Info Values greater than "**Max_Value**" or smaller than "**Min_Value**" will show no Icons.

8.3.2 Animated Icon

Used to show a loop animation of Icons from an Icon Library. The animation state is determined by the value of the VP.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A01	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x000A	2	
0x06	0x00	VP	2	Variable Pointer. High word: Unsigned Integer. Low word: Reserved, stores animation status.
0x08	0x01	Coordinates	4	Top-left coordinates of the Icons to display: (Xi, Yi).
0x0C	0x03	Restart_Animation	2	Indicates whether the animation should always start from the first frame when restarting. 0x0000: Animation continues from the last shown frame when reset. 0x0001: Animation starts from the first frame ("**Icon_Start**") when reset.
0x0E	0x04	Value_Stop	2	Value that stops the animation.
0x10	0x05	Value_Start	2	Value that starts the animation.
0x12	0x06	Icon_Stop	2	Icon displayed when the animation is stopped.
0x14	0x07	Icon_Start	2	Icon displayed at the first frame of the animation.
0x16	0x08	Icon_End	2	Icon displayed at the last frame of the animation.
0x18	0x09.H	LibId	1	Index in the FLASH memory of the Icon Library to use.
0x19	0x09.L	Transparency	1	0x00: Transparent background. Other Values: Opaque background.
PP Length:	**10 VPs**			



Caution Animated Icon uses **2 VPs**. Make sure to reserve VPs accordingly.

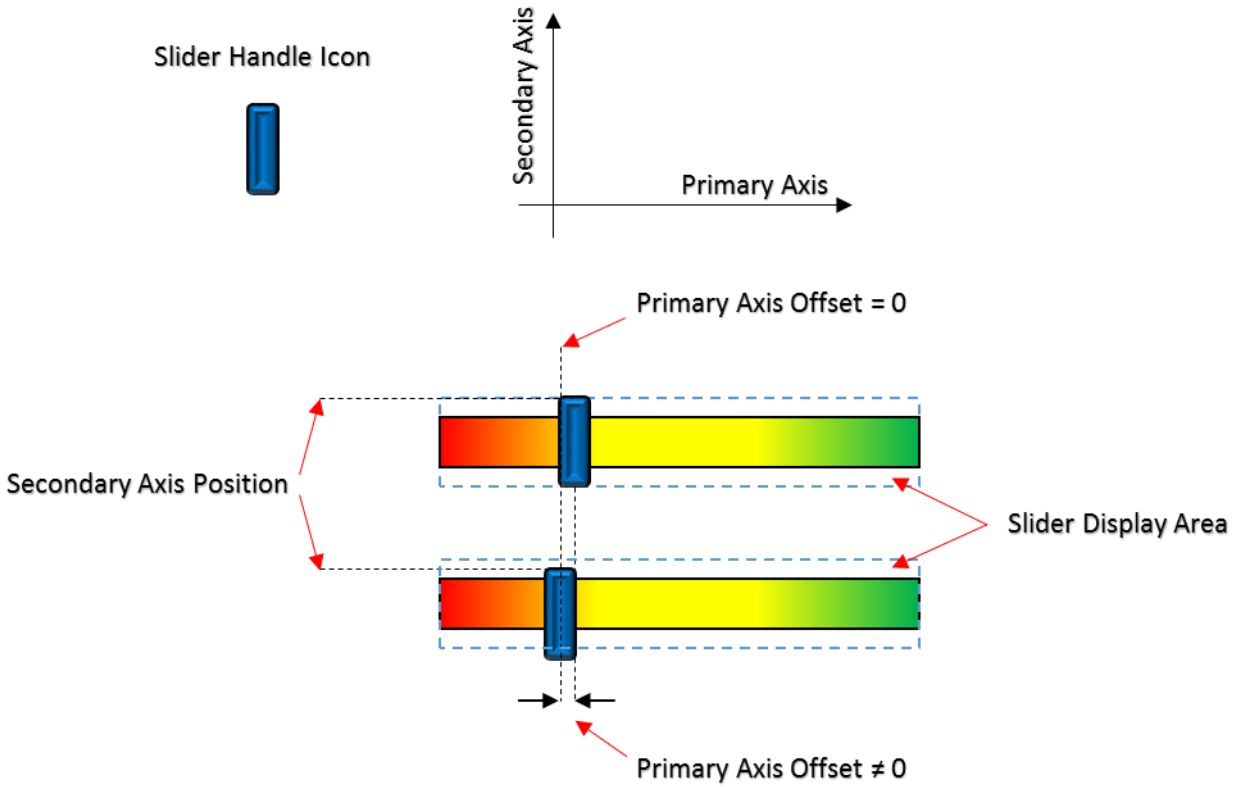


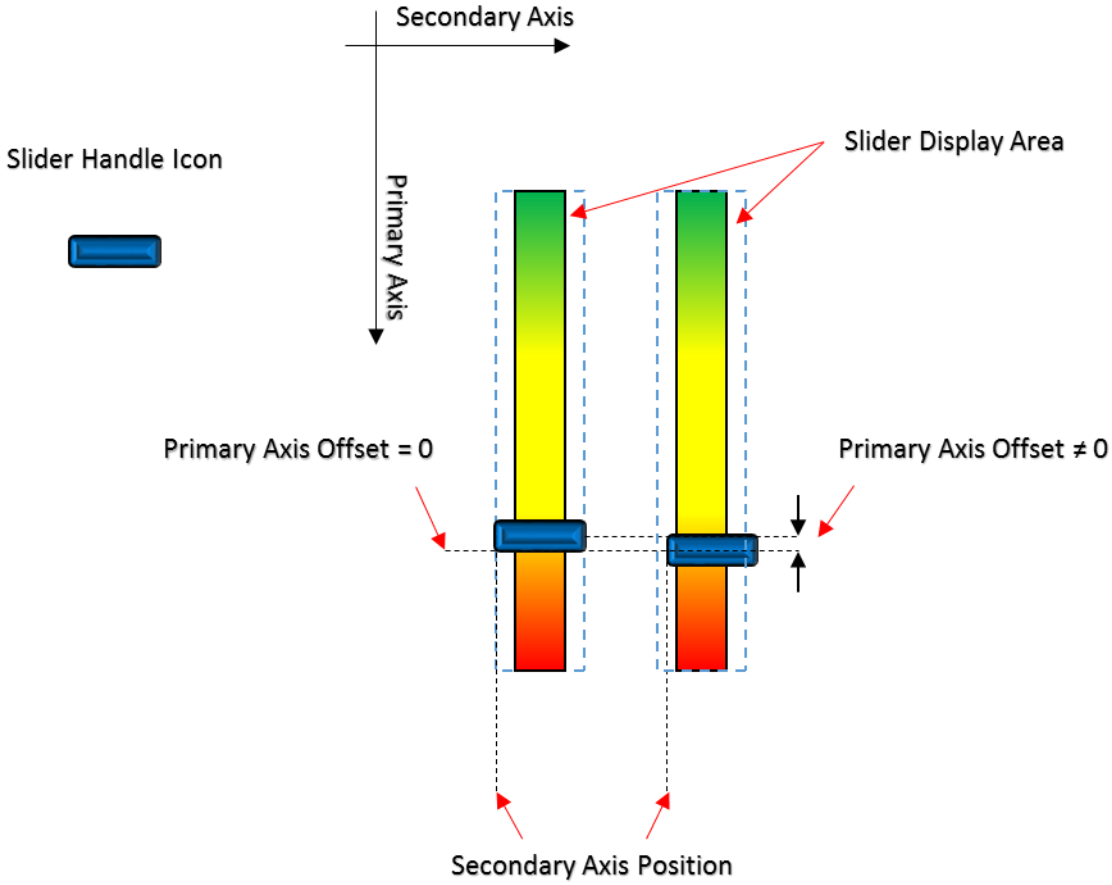
Info Values different from both "**Value_Start**" and "**Value_Stop**" will show no Icons.

8.3.3 Slider Display

Used to show an Icon that moves along a given axis (horizontal or vertical) based on the value of the VP. Typically used in linear graphs, or in conjunction with Slider Inputs.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A02	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x0009	2	
0x06	0x00	VP	2	Variable Pointer.
0x08	0x01	Min_Value	2	Minimum value accepted by the control.
0x0A	0x02	Max_Value	2	Maximum value accepted by the control.
0x0C	0x03	Coordinate_Initial	2	Minimum position of the slider, when its value equals "Min_Value". X coordinate for horizontal sliders, Y coordinate for vertical sliders.
0x0E	0x04	Coordinate_Final	2	Maximum position of the slider, when its value equals "Max_Value". X coordinate for horizontal sliders, Y coordinate for vertical sliders.
0x10	0x05	IconId	2	Index of the Icon in the Icon Library that will be used as the Slider handle.
0x12	0x06	Icon_Secondary_Position	2	Position of slider icon in the secondary axis. Y coordinate for horizontal sliders, X coordinate for vertical sliders.
0x14	0x07.H	Icon_Primary_Offset	1	Icon position offset adjustment on the primary axis, in pixels. Offset in the X axis for horizontal sliders, and in the Y axis for vertical sliders.
0x15	0x07.L	Orientation	1	Slider orientation. 0x00: Horizontal Other Values: Vertical.
0x16	0x08.H	LibId	1	Index in the FLASH memory of the Icon Library to use.
0x17	0x08.L	Transparency	1	0x00: Transparent background. Other Values: Opaque background.
0x18	0x09.H	VP_Mode	1	Value Memory Size. 0x00: 16-bit Integer. 0x01: 8-bit Unsigned Integer in High Byte of the VP. 0x02: 8-bit Unsigned Integer in Low Byte of the VP.
PP Length:	**10 VPs**			

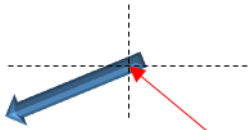

Slider Handle Icon
Secondary Axis
Primary Axis
Primary Axis Offset = 0
Secondary Axis Position
Slider Display Area
Primary Axis Offset ≠ 0


Secondary Axis
Slider Handle Icon
Primary Axis
Slider Display Area
Primary Axis Offset = 0
Primary Axis Offset ≠ 0
Secondary Axis Position

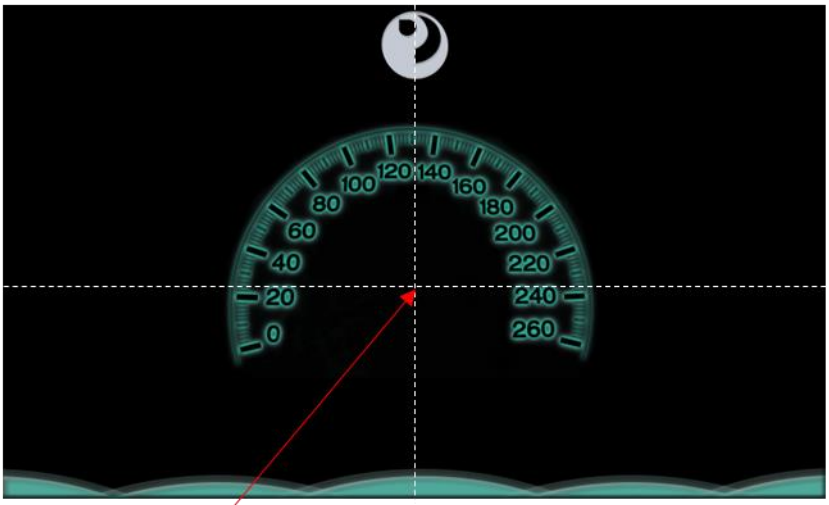
8.3.4 Rotating Icon

Used to show an Icon that pivots around a given rotation center, based on the value of the VP. Typically used in radial graphs, like speedometers and dials.

File Address	**PP Address**	Definition	Length (Bytes)	**Description**
0x00		0x5A05	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x000C	2	
0x06	0x00	VP	2	Variable Pointer.
0x08	0x01	IconId	1	Index of the Icon in the Icon Library that will be rotated (usually a dial needle).
0x0A	0x02	Icon_Center_X	2	Center of rotation on the Icon. X coordinate.
0x0C	0x03	Icon_Center_Y	2	Center of rotation on the Icon. Y coordinate.
0x0E	0x04	Screen_Center_X	2	Center of rotation on the Screen. The center of rotation of the Icon is placed in this point, and pivots around it. X coordinate.
0x10	0x05	Screen_Center_Y	2	Center of rotation on the Screen. The center of rotation of the Icon is placed in this point, and pivots around it. Y coordinate.
0x12	0x06	Min_Value	2	Minimum value.
0x14	0x07	Max_Value	2	Maximum value.
0x16	0x08	Angle_Min	2	Minimum angle, associated to “**Min_Value**”. Given in 0.5° steps. Range: [0,720] [0x000,0x2D0], which is equivalent to 0° to 360°.
0x18	0x09	Angle_Max	2	Maximum angle, associated to “**Max_Value**”. Given in 0.5° steps. Range: [0,720] [0x000,0x2D0], which is equivalent to 0° to 360°.
0x1A	0x0A.H	VP_Mode	1	Value Memory Size. 0x00: 16-bit Integer. 0x01: 8-bit Unsigned Integer in High Byte of the VP. 0x02: 8-bit Unsigned Integer in Low Byte of the VP.
0x1B	0x0A.L	LibId	1	Index in the FLASH memory of the Icon Library to use.
0x1C	0x0B	Transparency	1	0x00: Transparent background. Other Values: Opaque background.
PP Length:	**12 VPs**			



Icon Rotation Center



Screen Rotation Center



Clockwise Rotation

8.3.5 Bitwise Icon

Used to show fixed and/or animated Icons, according to a bit flag value on the VP. The value of each bit represents the state of a single Icon, and many Icons can be shown, in different states, based on the VPC.

Typically used to display several alarms at once, or to implement bar graphs.

<table>
<tr><th>File Address</th><th>PP Address</th><th>Definition</th><th>Length (Bytes)</th><th>Description</th></tr>
<tr><td>0x00</td><td></td><td>0x5A06</td><td>2</td><td></td></tr>
<tr><td>0x02</td><td></td><td>PP</td><td>2</td><td>Parameter Pointer.
0xFFFF: Disables PP (no run-time modification).
0x0000 - 0x6FFF: Enables PP (run-time modification possible).</td></tr>
<tr><td>0x04</td><td></td><td>0x000C</td><td>2</td><td></td></tr>
<tr><td>0x06</td><td>0x00</td><td>VP</td><td>2</td><td>Variable Pointer.</td></tr>
<tr><td>0x08</td><td>0x01</td><td>VP_Aux</td><td>2</td><td>Auxiliary Variable Pointer. 2 words. To match UnicView AD, should be allocated right after VP.
High word: Reserved, stores animation status for bit 0.
Low word: Reserved, stores animation status for bit 1.</td></tr>
<tr><td>0x0A</td><td>0x02</td><td>Active_Bits</td><td>2</td><td>Indicates which bits are displayed.
0b1: Active bit.
0b0: Inactive bit.</td></tr>
<tr><td>0x0C</td><td>0x03.H</td><td>Display_Mode</td><td>1</td><td>The following table describes what icons are shown when each bit value is either 0 or 1.
<table>
<tr><th rowspan="2">Mode</th><th colspan="2">Bit Value</th></tr>
<tr><th>0</th><th>1</th></tr>
<tr><td>0x00</td><td>Icon Start/Stop 0</td><td>Icon Start/Stop 1</td></tr>
<tr><td>0x01</td><td>Icon Start/Stop 0</td><td>None</td></tr>
<tr><td>0x02</td><td>Icon Start/Stop 0</td><td>Animation: Icon Start/Stop 1 -> Icon End 1</td></tr>
<tr><td>0x03</td><td>None</td><td>Icon Start/Stop 1</td></tr>
<tr><td>0x04</td><td>None</td><td>Animation: Icon Start/Stop 1 -> Icon End 1</td></tr>
<tr><td>0x05</td><td>Animation:
Icon Start/Stop 0 ->
Icon End 0.</td><td>Icon Start/Stop 1</td></tr>
<tr><td>0x06</td><td>Animation:
Icon Start/Stop 0 ->
Icon End 0</td><td>None</td></tr>
<tr><td>0x07</td><td>Animation:
Icon Start/Stop 0 ->
Icon End 0</td><td>Animation: Icon Start/Stop 1 -> Icon End 1</td></tr>
</table></td></tr>
<tr><td>0x0D</td><td>0x03.L</td><td>Alignment_Mode</td><td>1</td><td>Bit icons arranged mode.
0x00: Horizontal, no space reserved for inactive bits.
0x01: Vertical, no space reserved for inactive bits.
0x02: Horizontal, space reserved for inactive bits.
0x03: Vertical, space reserved for inactive bits.</td></tr>
<tr><td>0x0E</td><td>0X04.H</td><td>Transparency</td><td>1</td><td>0x00: Transparent background.
Other Values: Opaque background.</td></tr>
<tr><td>0x0F</td><td>0x04.L</td><td>LibId</td><td>1</td><td>Index in the FLASH memory of the Icon Library to use.</td></tr>
<tr><td>0x10</td><td>0x05</td><td>Icon_Start_Stop_0</td><td>2</td><td>Modes 0, 1, 2; Bit value = 0; Icon shown.
Modes 5, 6, 7; Bit value = 0: First icon in animation mode.</td></tr>
<tr><td>0x12</td><td>0x06</td><td>Icon_End_0</td><td>2</td><td>Modes 5, 6, 7; Bit value = 0: Last icon in animation mode.</td></tr>
<tr><td>0x14</td><td>0x07</td><td>Icon_Start_Stop_1</td><td>2</td><td>Modes 0, 3, 5; Bit value = 1; Icon shown.
Modes 2, 4, 7; Bit value = 1: First icon in animation mode.</td></tr>
<tr><td>0x16</td><td>0x08</td><td>Icon_End_1</td><td>2</td><td>Modes 2, 4, 7; Bit value = 1: Last icon in animation mode.</td></tr>
<tr><td>0x18</td><td>0x09</td><td>Coordinates</td><td>4</td><td>Top-left coordinates of the Icons to display: (Xi, Yi).</td></tr>
</table>

0x1C	0x0B	Spacing	2	The size reserved for each Icon, in pixels.
0x1E				0x00.
PP Length:	**12 VPs**			



Caution Animated Icon uses **3 VPs**. Make sure to reserve VPs accordingly.

8.3.6 Numeric Art

Works like a Numeric Display, using Icons instead of Fonts. Typically used when you need to display numeric information that needs an anti-aliased look. It uses fixed-point integer values.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A03	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x0007	2	
0x06	0x00	VP	2	Variable Pointer.
0x08	0x01	Coordinates	4	Top-left coordinates of the Icons to display: (Xi, Yi).
0x0C	0x03	Icon_0	2	Icon corresponding to digit 0. The Icon Library must follow this indexing order: [0123456789-.].
0x0E	0x04.H	LibId	1	Index in the FLASH memory of the Icon Library to use.
0x0F	0x04.L	Transparency	1	0x00: Transparent background. Other Values: Opaque background.
0x10	0x05.H	Integer_Digits	1	Number of digits to the left of the decimal separator.
0x11	0x05.L	Decimal_Digits	1	Number of digits to the right of the decimal separator.
0x12	0x06.H	VP_Mode	1	Value Memory Size. 0x00: 16-bit Integer. 0x01: 32-bit Integer. 0x02: 8-bit Unsigned Integer in High Byte of the VP. 0x03: 8-bit Unsigned Integer in Low Byte of the VP. 0x04: 64-bit Integer. 0x05: 16-bit Unsigned Integer. 0x06: 32-bit Unsigned Integer. In 8-bit mode, uses the Low Byte in return value. In 1-bit mode, uses the LSB.
0x13	0x06.L	Alignment	1	0x00: Left- alignment 0x01: Right- alignment.
PP Length:	**7 VPs**			



Caution The sum of **Integer** and **Decimal** digits should not exceed **20**.

8.3.7 Numeric Display

Used to display numeric information. It uses fixed-point integer values.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A10	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x000D	2	
0x06	0x00	VP	2	Variable Pointer.
0x08	0x01	Coordinates	4	Top-left coordinates of the text to display: (Xi, Yi).
0x0C	0x03	Color	2	Text color.
0x0E	0x04.H	LibId	1	Index in the FLASH memory of the Font to use.
0x0F	0x04.L	Font_Width	1	Font width, in pixels. Range: [0x04,0xFF]
0x10	0x05.H	Alignment	1	0x00: Right-alignment. 0x01: Left-alignment. 0x02: Center- alignment.
0x11	0x05.L	Integer_Digits	1	Number of digits to the left of the decimal separator.
0x12	0x06.H	Decimal_Digits	1	Number of digits to the right of the decimal separator.
0x13	0x06.L	VP _Mode	1	Value Memory Size. 0x00: 16-bit Integer. 0x01: 32-bit Integer. 0x02: 8-bit Unsigned Integer in High Byte of the VP. 0x03: 8-bit Unsigned Integer in Low Byte of the VP. 0x04: 64-bit Integer. 0x05: 16-bit Unsigned Integer. 0x06: 32-bit Unsigned Integer. In 8-bit mode, uses the Low Byte in return value. In 1-bit mode, uses the LSB.
0x14	0x07.H	Unit_Length	1	Length of text to append after the digits, in characters. Range: [0,11]
0x15	0x07.L	Unit_Text	0 → 11	Text to append after the digits. ASCII code.
PP Length:	**8 to 19 VPs**			



Caution The sum of **Integer** and **Decimal** digits should not exceed 20.

8.3.8 Text Display

Used to display textual information.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A11	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x000D	2	
0x06	0x00	VP	2	Variable Pointer.
0x08	0x01	Coordinates	4	Top-left coordinates of the text to display: (Xi, Yi).
0x0C	0x03	Color	2	Text color.
0x0E	0x04	Text_Area	8	Area where the text will be displayed: (Xi, Yi, Xf, Yf).
0x16	0x08	Max_Text_Length	2	Maximum text length, in bytes (characters). Range: [1,123].
0x18	0x09.H	LibId0	1	Index in the FLASH memory of the Font to use, for encoding modes 0x01 - 0x04.
0x19	0x09.L	LibId1	1	Index in the FLASH memory of the Font to use, for encoding modes 0x00 and 0x05, and other non-ASCII fonts for encoding modes 0x01 - 0x04.
0x1A	0x0A.H	Font_Width	1	Font width, in pixels. Range: [0x04,0xFF]
0x1B	0x0A.L	Font_Height	1	Font height, in pixels. Must be even. In encoding modes 0x01 - 0x04, must be twice the Width.
0x1C	0x0B.H	Spacing_Encoding	1	Character spacing mode is defined by bit 7: 0b0: Automatic spacing. 0b1: Fixed spacing. Font encoding is defined by bits 6 to 0: 0x00: 8 bit. 0x01: GB2312. 0x02: GBK. 0x03: BIG5 0x04: SJIS 0x05: UNICODE.
0x1D	0x0B.L	Character_Spacing	1	Spacing between characters, in pixels.
0x1E	0x0C.H	Line_Spacing	1	Spacing between lines, in pixels.
0x1F	0x0C.L			0x00 fixed
PP Length:	**13 VPs**			

To indicate that a string is finished, a terminator character (**0xFF** or **0x00**) must be appended to the data.

0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007	0x0008	0x0009	0x000A	
P	r	o	c	u	l	u	s		T	e	c	h	n	o	l	o	g	i	e	s	
0x50	0x72	0x6f	0x63	0x75	0x6c	0x75	0x73	0x20	0x54	0x65	0x63	0x68	0x6e	0x6f	0x6c	0x6f	0x67	0x69	0x65	0x73	0xFF

Proculus Technologies — Displayed Text

Terminator Character

0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007	0x0008	0x0009	0x000A	
P	r	o	c	u	l	u	s		T	e	c	h		o	l	o	g	i	e	s	
0x50	0x72	0x6f	0x63	0x75	0x6c	0x75	0x73	0x20	0x54	0x65	0x63	0x68	0xFF	0x6f	0x6c	0x6f	0x67	0x69	0x65	0x73	0xFF

Proculus Tech — Displayed Text

Terminator Character

8.3.9 Image Animation

Used to create an animation of Screens. Can be implemented via Serial Communication as a series of Screen jumps.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A04	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x0004	2	
0x06	0x00	0x00	2	0x00
0x08	0x01	First_Frame	2	PicId of the first animation frame.
0x0A	0x02	Last_Frame	2	PicId of the last animation frame.
0x0C	0x03.H	Frame_Time	1	Time spent in each animation Screen. Range: [0x00,0xFF], in 8 ms steps.
PP Length:	**4 VPs**			

8.3.10 Hex Display

Used to display numeric information in hexadecimal format, with optional digit separators.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A13	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x000D	2	
0x06	0x00	VP	2	Starting Variable Pointer of data string, data is encoded with BCD format. The data will be displayed in HEX format when half-byte data is greater than 0x9, e.g.: 0x32: display 32, 0xBF: display BF.
0x08	0x01	Coordinates	4	Top-left coordinates of the text to display: (Xi, Yi).
0x0C	0x03	Color	2	Text color.
0x0E	0x04.H	Byte_Num	1	Byte numbers to be displayed, 0x01 - 0x0F.
0x0F	0x04.L	LibId	1	Index in the FLASH memory of the Font to use.
0x10	0x05.H	Font_Width	1	Font width, in pixels. Range: [0x04,0xFF]
0x11	0x05.L	Separator_Format	0 → 15	Sequence of characters (ASCII) representing the separators for this Hex Display. The current value (contained on VP) will be shown in hexadecimal, and after each byte, a separator character is inserted. Special characters: 0x00 (blank), 0x0D (new line). Example: Current value in the 2 addresses starting at VP: 0x1234AABB. Separator Format: 0x00200D (blank, space, new line). What is shown on the LCM: 1234 AA BB
PP Length:	**6 to 21 VPs**			

8.3.11 RTC Display

Used to display current date and/or time, in digital format. Uses the internal RTC.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A12	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x000D	2	
0x06	0x00	0x00	2	0x00
0x08	0x01	Coordinates	4	Top-left coordinates of the text to display: (Xi, Yi).
0x0C	0x03	Color	2	Text color.
0x0E	0x04.H	LibId	1	Index in the FLASH memory of the Font to use.
0x0F	0x04.L	Font_Width	1	Font width, in pixels. Range: [0x04,0xFF]
0x10	0x05	String_Code	0 → 16	Display format string. Use ASCII characters and the Field Codes on the following table. E.g.: Current time = 2012-05-02 12:00:00 Wednesday, Y-M-D H: Q: S 0x00, will display “2012-05-02 12:00:00”. M-D W H: Q 0x00, will display “05-02 WED 12:00”.
PP Length:	**6 to 22 VPs**			

RTC Field Codes:

Description	Field Code	Range/Format Displayed
Year	Y	2000-2099
Month	M	01-12
Day	D	01-31
Hour	H	00-23
Minute	Q	00-59
Second	S	00-59
Date	W	SUN MON TUE WED THU FRI SAT
Coding end	0x00	

8.3.12 Analog Clock

Used to display current time, in analog format (a radial clock). Uses the internal RTC.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A12	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x000D	2	
0x06	0x00	0x0001	2	0x0001
0x08	0x01	Screen_Center	4	Center of rotation on the Screen. The center of rotation of the Icon is placed in this point, and pivots around it. (X, Y).
0x0C	0x03	Icon_Hour	2	Index of the Icon showing the Hour hand. 0xFFFF: none.
0x0E	0x04	Icon_Hour_Central	4	Center of rotation on the Hour Icon. (X, Y).
0x12	0x06	Icon_Minute	2	Index of the Icon showing the Minute hand. 0xFFFF: none.
0x14	0x07	Icon_Minute_Central	4	Center of rotation on the Minute Icon. (X, Y).
0x18	0x09	Icon_Second	2	Index of the Icon showing the Second hand. 0xFFFF: none.
0x1A	0x0A	Icon_Second_Central	4	Center of rotation on the Second Icon. (X, Y).
0x1E	0x0C.H	LibId	1	Index in the FLASH memory of the Icon Library to use.
0x1F		0x00	1	0x00
PP Length:	**13 VPs**			

8.3.13 Table Display

Used to display tabular text.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A22	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x000C	2	
0x06	0x00	VP	2	Variable Pointer.
0x08	0x01.H	Column_Count	1	Number of columns. Range: [0x01,0xFF].
0x09	0x01.L	Row_Count	1	Number of rows. Range: [0x01,0xFF].
0x0A	0x02.H	First_Column	1	Column index (0 based) of the first column (left-most) to be shown. Range: [0x00,0xFF].
0x0B	0x02.L	First_Row	1	Row index (0 based) of the first row(top-most) to be shown. Range: [0x00,0xFF].
0x0C	0x03.H	Cell_Length	1	Cell Length, in words (number of characters = 2*[Cell Length]). 0x01 - 0x7F: Data length for all cell (2 to 254 characters). 0x00: Data starting from VP defines the length of each column.
0x0D	0x03.L	Format	1	Character spacing mode is defined by bit 7: 0b0: Automatic spacing. 0b1: Fixed spacing. Table content type is defined by bit 6: 0b0: Text. 0b1: First two VPs in each column data indicate the format. See the next table for details. Table border is defined by bit 5: 0b0: Show border. 0b1: Hide border. Bit 4: 0x00 Font encoding is defined by bits 3 to 0: 0x00: 8 bit. 0x01: GB2312. 0x02: GBK. 0x03: BIG5 0x04: SJIS 0x05: UNICODE.
0x0E	0x04	Table_Area	8	Area where the table will be displayed: (Xi, Yi, Xf, Yf).
0x16	0x08	Border_Color	2	Border color.
0x18	0x09	Color	2	Text color.
0x1A	0x0A.H	LibId0	1	Index in the FLASH memory of the Font to use, for encoding modes 0x01 - 0x04.
0x1B	0x0A.L	LibId1	1	Index in the FLASH memory of the Font to use, for encoding modes 0x00 and 0x05, and non-ASCII fonts for encoding modes 0x01 - 0x04.
0x1C	0x0B.H	Font_Width	1	Font width, in pixels. Range: [0x04,0xFF]
0x1D	0x0B.L	Font_Height	1	Font height, in pixels. Must be even. In encoding modes 0x01 - 0x04, must be twice the Width.
0x1E	0x0C.H	Show_Column_Header	1	0x00: Show column header when “First_Row” != 0; 0x01: Show column header only when “First_Row” == 0;
0x1F	0x0C.L	Show_Row_Header	1	0x00: Show row header when “First_Column” != 0; 0x01: Show row header only when “First_Column” == 0;
PP Length:	**13 VPs**			

The following table summarizes the content type custom formats. The indicated values must be written into the first 2 VPs of each cell data.

Address	Value	Format Description
First VP, High Byte	0x00	16-bit Integer.
	0x01	32-bit Integer.
	0x02	8-bit Unsigned Integer in High Byte of the VP.
	0x03	8-bit Unsigned Integer in Low Byte of the VP.
	0x04	64-bit Integer.
	0x05	16-bit Unsigned Integer.
	0x06	32-bit Unsigned Integer.
	0x10	Time Format, hh:mm:ss, BCD.
	0x11	Time Format, hh-mm-ss, BCD.
	0x12	Time Format, YYYY-MM-DD hh:mm:ss, BCD
	0xFF	Text
First VP, Low Byte	[0x00,0xFF]	In modes 0x00 - 0x06: High nibble is the number of integer digits. Low nibble is the number of decimal digits. In modes 0x10 - 0x11: BCD length, in bytes. Other modes: Not used.
Second VP	[0x0000,0xFFFF]	Text Color



Info

You can use double terminator characters (**0xFFFF**) to indicate the end of text in a cell.

If "**Cell_Length**" is 0, the cell length is specified for each column. Starting from the High Byte of the VP, each byte indicates the Length of each column. In this case, the actual data starts from VP + ("Column Count"/2) (rounded up).

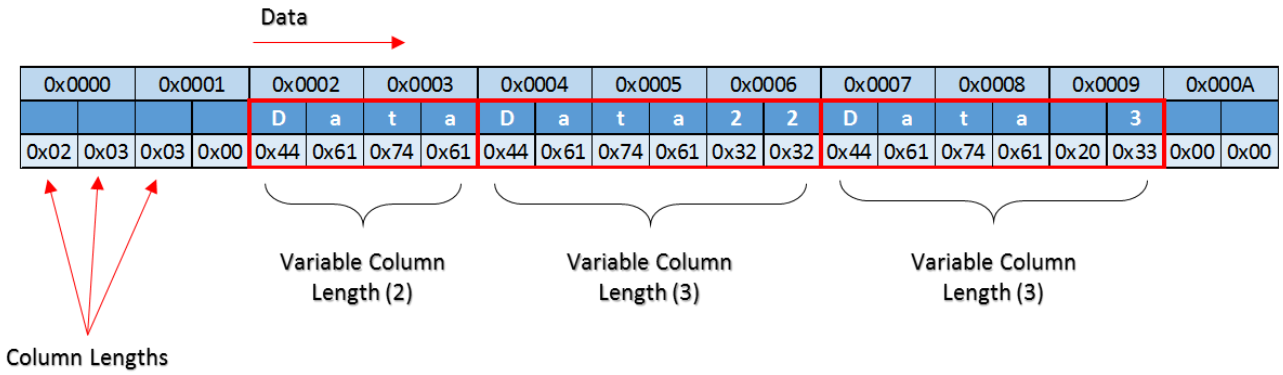
In the following examples, consider a table with VP = 0x0000, and 3 columns of data.

- **Fixed cell length:**

0x0000		0x0001		0x0002		0x0003		0x0004		0x0005		0x0006		0x0007		0x0008		0x0009		0x000A	
D	a	t	a			D	a	t	a	2	2	D	a	t	a		3				
0x44	0x61	0x74	0x61	0x20	0x20	0x44	0x61	0x74	0x61	0x32	0x32	0x44	0x61	0x74	0x61	0x20	0x33	0x00	0x00	0x00	0x00

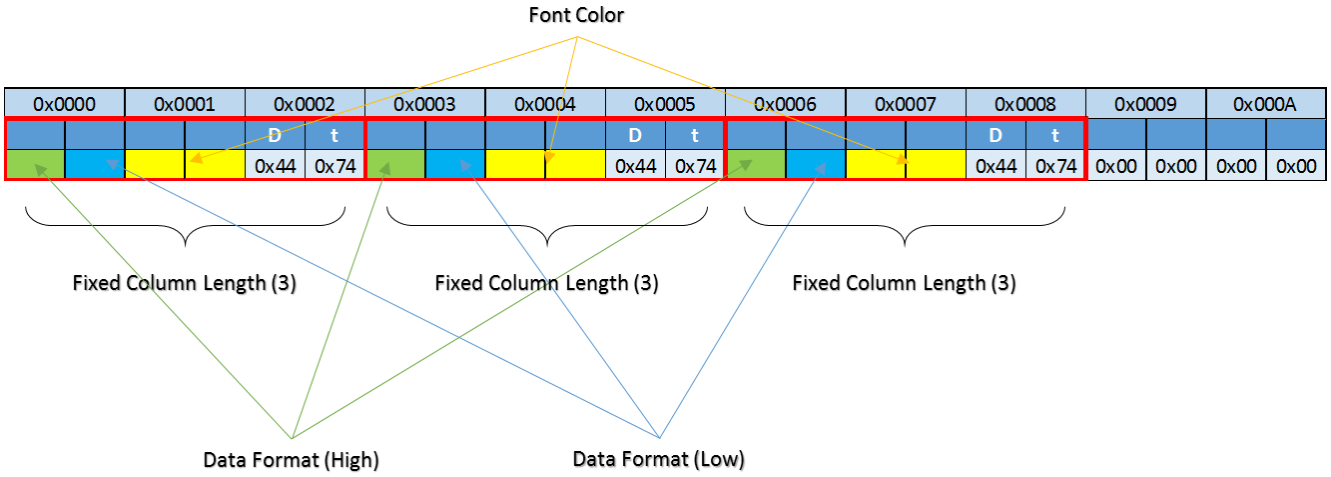
Fixed Column Length (3) | Fixed Column Length (3) | Fixed Column Length (3)

- **Variable cell length ("Cell Length" == 0x00):**

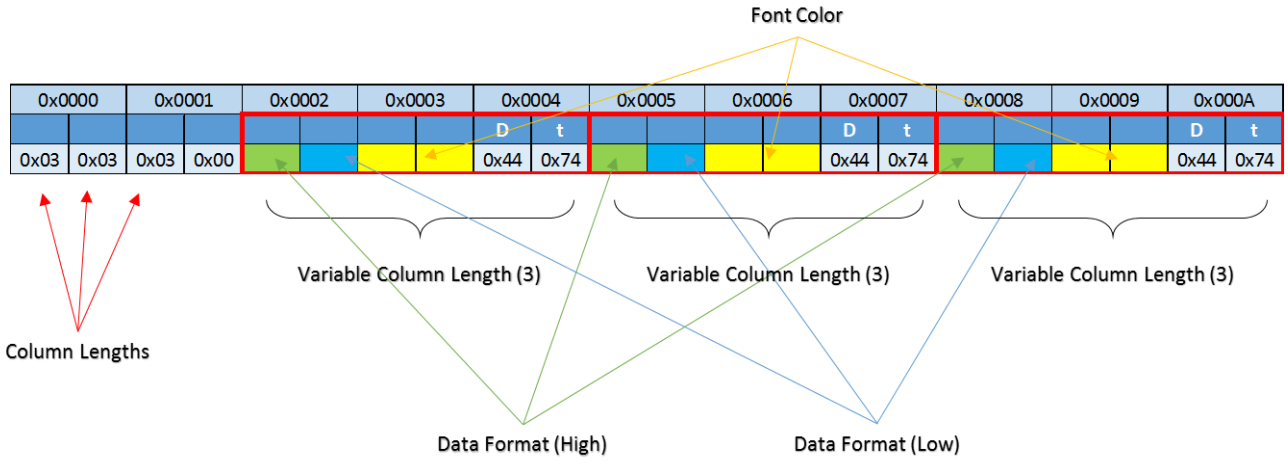


When using data type custom formats, you must reserve at least 3 VPs for each cell, because custom formats require the first 2 VPs of each cell.

- **Fixed cell length:**



- **Variable cell length ("Cell Length" == 0x00):**



8.3.14 Trend Curve Display

Used to plot line graphs.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A20	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x000A	2	
0x06	0x00	0x0000	2	0x0000
0x08	0x01	Curve_Area	8	Area where the trend curve will be plotted: (Xi, Yi, Xf, Yf).
0x10	0x05	Vertical_Origin	2	Origin point of the vertical axis.
0x12	0x06	Vertical_Origin_Value	2	Value associated to the vertical axis origin.
0x14	0x07	Color	2	Color of the plot.
0x16	0x08	Vertical_Zoom	2	Vertical Zoom. Range: [0x0000,0x7FFF].
0x18	0x09.H	Channel	1	Channel of the curve. Range: [0x00, 0x07].
0x19	0x09.L	Horizontal_Increment	1	Horizontal Increment. Range: [0x01,0xFF].
PP Length:	**10 VPs**			

"**Vertical_Zoom**" is calculated by this formula:

$$Vertical_Zoom = \frac{(Y_f - Y_i) * 256}{(V_{max} - V_{min})}$$

Where:

- Y_f = Bottom Y coordinate of the "**Curve_Area**".
- Y_i = Top Y coordinate of the "**Curve_Area**".
- V_{max} = Maximum value on the plot.
- V_{min} = Minimum value on the plot.



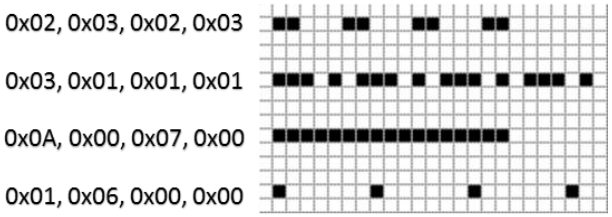
Info Refer to section "4.4.5 - Trend Curve Buffer Space" for more information.

8.3.15 Graphic Primitives Display (GPD)

Used to access many graphic manipulation functions, like copy/pasting and shape drawing.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A21	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x0008	2	
0x06	0x00	VP	2	Variable Pointer.
0x08	0x01	Text_Area	8	Area where graphics are displayed: (Xi, Yi, Xf, Yf). Only valid for Commands 0x0001 - 0x0005, 0x0009 - 0x000B
0x10	0x05.H	Enable_Dash_Pattern	1	Enables or disables the dashed line pattern for Commands 0x0002, 0x0003, 0x0009, 0x000A. 0x5A: The drawn lines are dashed. Other Values: Full lines are drawn.
0x11	0x05.L	Dash_Pattern	4	Dash pattern format. The pattern has 4 segments (bytes): Byte 1: Length of the first opaque segment. Byte 2: Length of the first transparent segment. Byte 3: Length of the second opaque segment. Byte 4: Length of the second transparent segment.
0x15			13	0x00
PP Length:	**6 VPs**			

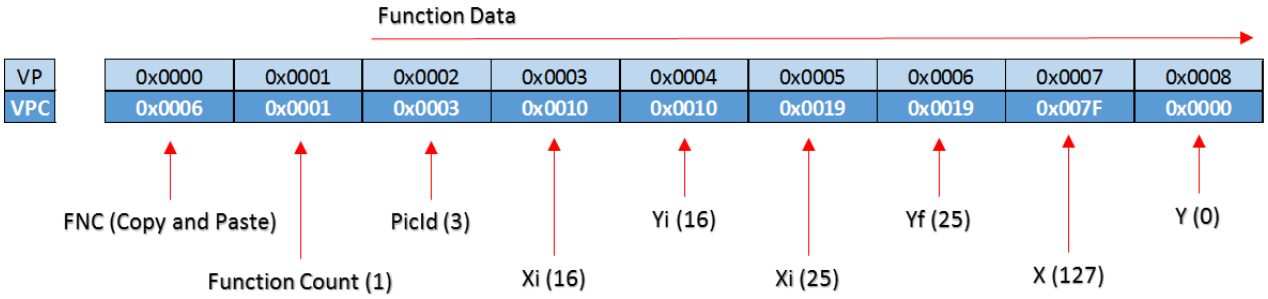
Examples of dash patterns:



The graphic primitives available (also called **Functions**) are used by writing their Function Code and Arguments in the VP of the Graphic Primitives Display. The first address (VP) holds the Function Code, which identifies the Function. The second address (VP + 1) indicates how many times this Function should be displayed. From the third address (VP + 2) and onwards, the information on how to draw these Functions is stored.

Address	Definition	Description
VP	Function Code (FNC)	Identifies the desired Function.
VP + 1	Function Count	Number of Functions displayed. Only works with the same FNC. For FNC 0x0002, it's the number of vertices - 1.
VP + 2	Function Data	Function arguments, like coordinates, color, etc.

For example, to perform a **Copy and Paste** Function:



All Functions have a **Control Byte**, which is used to **finish** the displaying of graphics, or to **skip** one specific Function. To finish or skip Functions, write the corresponding value to their Control Bytes:

- Finish Display (0xFF): No further Functions on the current GPD are displayed.
- Skip Function (0xFE): Skips only one Function on the current GPD.

The following table summarizes all available Functions.

FNC	Function	Data			
		Relative Address	Length (word)	Definition	Description
0x0001	Dot	0x00	2	(X,Y)	Dot coordinates. High byte of X coordinate is the Control Byte.
		0x02	1	Color	Dot color.
0x0002	Line	0x00	1	Color	Line color.
		0x01	2	(X,Y)0	Coordinates of Vertex 0. High byte of X coordinate is the Control Byte.
		0x03	2	(X,Y)1	Coordinates of Vertex 1. High byte of X coordinate is the Control Byte.
		0x01+2 *n	2	(X,Y)n	Coordinates of Vertex n. High byte of X coordinate is the Control Byte.
0x0003	Rectangle	0x00	2	(X,Y)i	Top-left coordinates. High byte of X coordinate is the Control Byte.
		0x02	2	(X,Y)f	Bottom-right coordinates.
		0x04	1	Color	Color of the rectangle.
0x0004	Rectangle Area	0x00	2	(X,Y)i	Top-left coordinates. High byte of X coordinate is the Control Byte.
		0x02	2	(X,Y)f	Bottom-right coordinates.
		0x04	1	Color	Fill color.
0x0005	Circle	0x00	2	(X,Y)	Coordinates of the center of the circle. High byte of X coordinate is the Control Byte.
		0x02	1	Radius	Radius of the circle.
		0x03	1	Color	Circle color.
0x0006	Copy and Paste	0x00	1	PicId	Index of the Screen to be copied. High byte of X coordinate is the Control Byte.
		0x01	2	(X,Y)i	Top-left coordinates of the copy area.
		0x03	2	(X,Y)f	Bottom-right coordinates of the copy area.
		0x05	2	(X,Y)	Paste position on current Screen, Top-left coordinates.
0x**07	Icon	0x00	2	(X,Y)	Top-left coordinates where the Icon will be displayed. High byte of X coordinate is the Control Byte.
		0x02	1	IconId	Index of the Icon in the Icon Library that will be displayed. High byte of FNC specifies the LibId. Transparent background.
0x0008	Color Fill	0x00	2	(X,Y)	First pixel to paint. All adjacent pixels with the same color will be also painted. High byte of X coordinate is the Control Byte.
		0x02	1	Color	Fill color.
0x0009	Spectrum Graph	0x00	1	Color	Connects (X, Yi), (X, Yf) with a line segment. High byte of X coordinate is the Control Byte.
		0x01	1	X	
		0x02	1	Yi	
		0x03	1	Yf	

0x000A	Segment	0x00	1	Color	Connects (Xi, Yi), (Xf, Yf) with line segment. High byte of Xi coordinate is the Control Byte.
		0x01	1	Xi	
		0x02	1	Yi	
		0x03	1	Xf	
		0x04	1	Yf	
0x000B	Arc	0x00	1	Color	Arc color.
		0x01	2	(X,Y)	Coordinates of the center of the arc. High byte of X coordinate is the Control Byte.
		0x03	1	Radius	Radius of the arc.
		0x04	1	Initial Angle	Initial angle of the arc. Range: [0,720], in 0.5° increments.
		0x05	1	Final Angle	Final angle of the arc. Range: [0,720], in 0.5° increments.
0x000C	Character	0x00	1	Color	Character color.
		0x01	2	(X,Y)0	Top-left coordinates of the character. High byte of X coordinate is the Control Byte.
		0x03.H	0.5	LibId	Index in the FLASH memory of the Font to use.
		0x03.L	0.5	Encoding	Font encoding: 0x00: 8 bit. 0x01: GB2312. 0x02: GBK. 0x03: BIG5 0x04: SJIS 0x05: UNICODE.
		0x04.H	0.5	Font Width	Font width, in pixels. Range: [0x04,0xFF]
		0x04.L	0.5	Font Height	Font height, in pixels.
		0x05	1	Text0	Character code. Only valid on high byte of 8-bit encoding. For 0x01-0x04 encoding and ASCII data, DefaultFont will be used.
0x000D	Color Inverting Rectangle	0x00	2	(X,Y)i	Top-left coordinates. High byte of X coordinate is the Control Byte.
		0x02	2	(X,Y)f	Bottom-right coordinates.
		0x04	1	Inverting Bits	Indicates the bits which will inverse the color. 0x0000: No color inversion. 0xFFFF: all color bits are inversed.
0x000E	Binary Color Graph	0x00	2	(X,Y)i	Top-left coordinates of the graph. High byte of X coordinate is the Control Byte.
		0x02	1	Width	Width of the graph, in pixels.
		0x03	1	Height	Height of the graph, in pixels.
		0x04	1	Color1	Color associated to value 0b1.
		0x05	1	Color0	Color associated to value 0b0. If Color0 == Color1, then Color0 will be transparent.
		0x06	?	Binary Data	Binary data to display. Data is read MSB to LSB, drawing the pixels left to right. A set bit (0b1) is painted with Color1, and a clear bit (0b0) is painted with Color0.
0x000F	Bitmap	0x00	2	(X,Y)i	Top-left coordinates of the bitmap. High byte of X coordinate is the Control Byte.
		0x02	1	Width	Bitmap width, in pixels. Bitmap size must not exceed 28665 pixels.
		0x03	1	Height	Bitmap height in pixels. Bitmap size must not exceed 28665 pixels.
		0x04	?	Color Data	Color data to display. Each pixel occupies 1 word (MSB aligned, RGB565 format).
0x0010	Zoom and Paste	0x00	2	(X,Y)	Paste position on current Screen, Top-left coordinates. High byte of X coordinate is the Control Byte.
		0x02	2	(X,Y)i	Top-left coordinates of the zoom area.
		0x04	2	(X,Y)f	Bottom-right coordinates of the zoom area.

8.3.16 QR Code Display

Used to display QR Codes generated from the value of the VP.

File Address	PP Address	Definition	Length (Bytes)	Description
0x00		0x5A25	2	
0x02		PP	2	Parameter Pointer. 0xFFFF: Disables PP (no run-time modification). 0x0000 - 0x6FFF: Enables PP (run-time modification possible).
0x04		0x0004	2	
0x06	0x00	VP	2	Variable Pointer.
0x08	0x01	Coordinates	4	Top-left coordinates of the Icons to display: (Xi, Yi).
0x0C	0x03	Pixel_Size	2	QR Pixel size, in pixels. It's the size of the smallest square on a QR Code. Range: [0x01,0x07].
0x0B-0x1F	0x05.L	Reserved	18	0x00
PP Length:	**6 VPs**			

The QR Code automatically reads and updates the data starting from its VP. To identify the end of the text data, append a double terminator (**0x0000** or **0xFFFF**). Data range: 0 to 458 bytes.

The size of the QR Code is set according to the data length:

- Up to 154 bytes: 45x45 QR pixels.
- Up to 458 bytes: 73x73 QR pixels.



Info QR Code needs the "12_QR_CODE.bin" file to work normally.

9 AD Assembly

AD Assembly is an upcoming feature, not currently supported. All notes about AD Assembly or OS are only for reference.

10 Modbus

There are special AD firmware versions that can use Modbus RTU as their communication protocol. When using Modbus Protocol, the standard AD Protocol is disabled.



Info When using Modbus Protocol, the standard AD Protocol is disabled. Project Download to LCM via Serial Communication is also disabled.

There are two separate firmware versions that support Modbus protocol:

- Modbus RTU Master
- Modbus RTU Slave

10.1 Master

In this firmware version, the LCM can operate as a Master device in a Modbus communication bus. The following table shows the supported Modbus Functions.

Function Code	Function Name	Data length (bytes)
0x01	Read Coils	<No. of Coils>/8 (rounded up)
0x02	Read Discrete Inputs	<No. of Inputs>/8 (rounded up)
0x03	Read Multiple Holding Registers	<No. of Registers>*2
0x04	Read Input Registers	<No. of Registers>*2
0x05	Write Single Coil	2
0x06	Write Single Holding Register	2
0x07	Read Exception Status	1
0x0F	Write Multiple Coils	<No. of Coils>
0x10	Write Multiple Holding Registers	<No. of Registers>*2
0x11	Report Slave ID	1

In this mode, there are some differences in how the mapping of RAM address should be done:

1. **Modbus Variable Range**

You must designate a certain VP range as Modbus Variable Range, which is an address interval that can be accessed by the Modbus Protocol.

VPs in this range can be read and written by the Modbus Functions. For example, you may designate the range [0x0100,0x4000] as Modbus Variable Range.

2. General Settings and Function Settings

The RAM addresses from 0x5000 to 0x6FFF are used as configuration data for the Modbus protocol.

VPs 0x5000 to 0x5007 store the **General Settings**:

Address	Definition	Value	Read Write
0x5000	Enable/Disable Modbus Protocol	0x5AA5 = Enables Modbus Protocol. Other Values = Disable Modbus Protocol.	W
0x5001.H	Save Modbus Settings	0x5A = Saves Modbus General and Function Settings in the RAM to the LCM's RAM Initialization memory.	W
0x5001.L	Load Modbus Settings	0x5A = Loads Modbus General and Function Settings from the LCM's RAM Initialization memory to the RAM.	W
0x5002	Serial Port Baud Rate	Baud Rate, in Fixed-Point format (xxx.x kbps). Maximum: 999.9 kbps. Example: 0x0480 = 115.2 kbps = 115200 bps	R/W
0x5003.H	Serial Port Settings	0x00 = 8N1 0x01 = 8E1 0x02 = 801 0x03 = 8N2	R/W
0x5003.L	Undefined	0x00	-
0x5004.H	Save Modbus Variable Range	0x5A = Saves the data in the Modbus Variable Range to the LCM's RAM Initialization memory.	W
0x5004.L	Load Modbus Variable Range	0x5A = Loads the data from the LCM's RAM Initialization memory into the Modbus Variable Range.	W
0x5005.H	Modbus Variable Range Definition Start	High byte of the fist VP on the Modbus Variable Range. The low byte is fixed to 0x00.	R/W
0x5005.L	Modbus Variable Range Definition End	High byte of the last VP on the Modbus Variable Range. The low byte is fixed to 0x00.	R/W
0x5006	Undefined	0x0000	-
0x5007	Undefined	0x0000	-

VPs 0x5008 to 0x6FFF store the **Function Settings**. Each Function occupies 8 VPs, and is totally independent from other Functions.

A Function describes a single Modbus Command. There are 1023 available Functions, and you can enable or disable them at any order.

Function 1	0x5008
	0x5009
	0x500A
	0x500B
	0x500C
	0x500D
	0x500E
	0x500F
Fucntion n	0x5008 + n*8
	0x5009 + n*8
	0x500A + n*8
	0x500B + n*8
	0x500C + n*8
	0x500D + n*8
	0x500E + n*8
	0x500F + n*8
Function 1022	0x6FF8
	0x6FF9
	0x6FFA
	0x6FFB
	0x6FFC
	0x6FFD
	0x6FFE
	0x6FFF

The following table shows the parameter mapping of the first Modbus Function.

Address	Definition	Value	Read Write
0x5008.H	Enable/Disable Function	0x5A = Enables this Function, Other Values = Disables this Function.	R/W
0x5008.L	Modbus Slave ID	Slave ID of the target device.	R/W
0x5009.H	Modbus Function Code	Function Code of this Function.	R/W
0x5009.L	Data Length	Length of the Modbus data.	R/W
0x500A	Response Timeout	The period, from the beginning of the transmission, within which the Slave Device must answer this Function, in ms. Maximum: 9999 ms.	R/W
0x500B - 0x500C	Sending Mode	Two words that control when this Function is sent. 0x0000:**** = Sends this Function continuously. The second byte doesn't matter. 0x0001:PicId = Sends this Function continuously, only when the current Screen matches the PicId defined in the second byte. 0x0002:VP = Writing 0x5A in the low byte of the VP defined in the second byte sends this Function once. The low byte of the VP is automatically cleared.	R/W
0x500D	Modbus Variable Address	Indicates the VP that contains data to be sent, or receives the response data from the Slave Device. If this value is 0xFF##, the received data will be written in the Trend Curve Buffer. "##" defines the channel.	R/W
0x500E	Slave Device Address	Indicates the target address on the Slave Device.	R/W
0x500F.H	Function Response Status	Indicates the status of the last time this Function was sent. 0x00 = Failure, 0xFF = Success.	R
0x500F.L	Undefined	0x00	-

10.2 Slave

In this firmware version, the LCM can operate as a Slave device in a Modbus communication bus. The following table shows the supported Modbus Functions.

Function Code	Function Name
0x03	Read VPs
0x04	Read Registers
0x0F	Write Registers
0x10	Write VPs
0x10	Write Trend Curve Buffer

In Modbus Slave mode, the Frame Header Configuration Register holds two settings:

- R3: Slave ID assigned to the LCM.
- RA: Serial Port packet settings: 0x00 = 8N1, 0x01 = 8E1, 0x02 = 8O1, 0x8N2.

Examples:

Function Code	Function Name	Commands	Description
0x03	Read VPs	Tx: 5A 03 00 00 00 02 C9 20 Rx: 5A 03 04 00 00 00 00 10 F6	Read 2 VPs, starting from 0x0000.
0x04	Read Registers	Tx: 5A 04 00 00 00 02 7C E0 Rx: 5A 04 04 71 40 00 00 0B A9	Read 4 Registers, starting from 0x00.
0x0F	Write Registers	Tx: 5A 0F 00 03 00 00 02 00 01 A8 20 Rx: 5A 0F 00 03 00 00 A8 E0	Write 2 Registers, starting from 0x03.
0x10	Write VPs	Tx: 5A 10 00 00 00 02 04 31 32 33 34 6D 9E Rx: 5A 10 00 00 00 02 4C E3	Write 2 VPs, starting from 0x0000.
0x10	Write Trend Curve Buffer	Tx: 5A 10 FF 01 00 02 04 3F FE 20 00 27 96 Rx: 5A 10 FF 01 00 02 2D 37	Write 2 values in into Channel 0 of Trend Curve Buffer. It's the same as writing VPs, but the **high byte** of the address (0xFF) indicates that this is a Trend Curve operation, and the **low byte** is the channels to write to.