

AN15129

Implement RGB LCD Interface by SmartDMA on MCX A577

Rev. 1.0 — 30 September 2026

Application note

Document information

Information	Content
Keywords	AN15129, MCX A577, SmartDMA, RGB LCD, parallel RGB interface, RGB565, LVGL, framebuffer, partial rendering, LCD timing, Arm Cortex-M33
Abstract	This application note describes how to use SmartDMA on the MCX A577 as a software LCD controller to generate 480 x 272 parallel RGB LCD timing in RGB565 format.



1 Overview

This application note describes how to use SmartDMA on the MCX A577 device to generate 480 x 272 parallel RGB LCD timing. A prebuilt SmartDMA firmware array is copied to SRAMX3 and started through its API table. SmartDMA then performs continuous full-frame scanning while the Arm Cortex-M33 builds the UI, schedules animations, and updates dirty areas.

The demonstration UI is a kitchen timer. Three decorative bubbles in the upper-left corner use the LVGL animation engine.

1.1 Key features

- Resolution: 480 x 272; pixel format: RGB565, 16 bits per pixel.
- SmartDMA drives 21 signals: D0-D15, PCLK, HSYNC, VSYNC, DE, and DISP.
- The full framebuffer is 261120 bytes; LVGL uses a 480 x 27 partial draw buffer.
- The SmartDMA firmware array is installed into SRAMX3 at runtime; Arm and SmartDMA share the framebuffer in SRAM.
- The current version is verified by building and downloading it with MCUXpresso IDE, monitoring the serial output, and testing the physical display.
- The three upper-left bubbles float asynchronously; their moving redraw areas are visible in the serial flush coordinates.

1.2 Scope

This document is intended for developers who must generate RGB LCD raster timing with MCX A577 SmartDMA without a dedicated LCD controller. If the resolution, panel, or sampling edge changes, recalculate the timing and verify the waveforms with a logic analyzer.

2 Fundamentals

This chapter introduces the concepts required to understand the SmartDMA-based RGB LCD solution, including:

- The parallel RGB LCD interface and its signal set
- The RGB565 pixel format and framebuffer size
- LVGL partial rendering into a shared full framebuffer
- SmartDMA operation as a software LCD controller
- Raster timing generation without a dedicated hardware LCD controller

2.1 Parallel RGB LCD interface

A parallel RGB panel typically has no display memory. The controller must continuously output pixels and synchronization signals line by line and frame by frame. If PCLK or the sync signals stop, the panel can display corrupted content or lose the image. [Table 1](#) lists the key inputs.

Table 1. RGB LCD interface signals

Signal	Function	Polarity/State in This Project
D[15:0]	One RGB565 pixel per PCLK	Updated during the active display area
PCLK	Pixel sampling clock	Data is stable before the falling edge; the panel samples correctly with the present wiring
HSYNC	Line synchronization	Active low

Table 1. RGB LCD interface signals...continued

Signal	Function	Polarity/State in This Project
VSYNC	Frame synchronization	Active low
DE	Data-enable window	Active high
DISP	Display enable	Held high during normal display

2.2 RGB565 pixel format

A standard 16-bit RGB565 pixel contains 5 red bits, 6 green bits, and 5 blue bits: bit[15:11] is R, bit[10:5] is G, and bit[4:0] is B. The size of one frame is:

480 × 272 × 2 bytes = 261120 bytes, approximately 255 kB

This project defines SmartDMA PIO0-PIO4 as LCD R3-R7, PIO5-PIO10 as G2-G7, and PIO11-PIO15 as B3-B7. To make these named signals carry the correct colors, the LVGL output swaps the red and blue fields before writing the frame buffer. When using another LCD module, connect by signal name. Do not reuse the connector pin numbers from the validation hardware.

```
lcdColor = ((color & 0xF800U) >> 11U) |
            ( color & 0x07E0U)          |
            ((color & 0x001FU) << 11U);
```

2.3 Framebuffer and partial rendering

For each frame, SmartDMA reads 480 x 272 pixels from the same full framebuffer. LVGL does not require another full-size draw buffer. It renders the dirty areas into a 27-line partial buffer and the flush callback copies the pixels into the full framebuffer. This approach reduces LVGL draw-buffer memory, but it does not reduce the SmartDMA full-frame scan bandwidth.

2.4 Role of SmartDMA

SmartDMA can be viewed as a software LCD controller. The application installs a prebuilt firmware image in SRAMX3, passes the stack and framebuffer addresses through a parameter block, and boots the RGB LCD API entry. After startup, SmartDMA drives the RGB data and control signals without Arm participation in per-pixel timing.

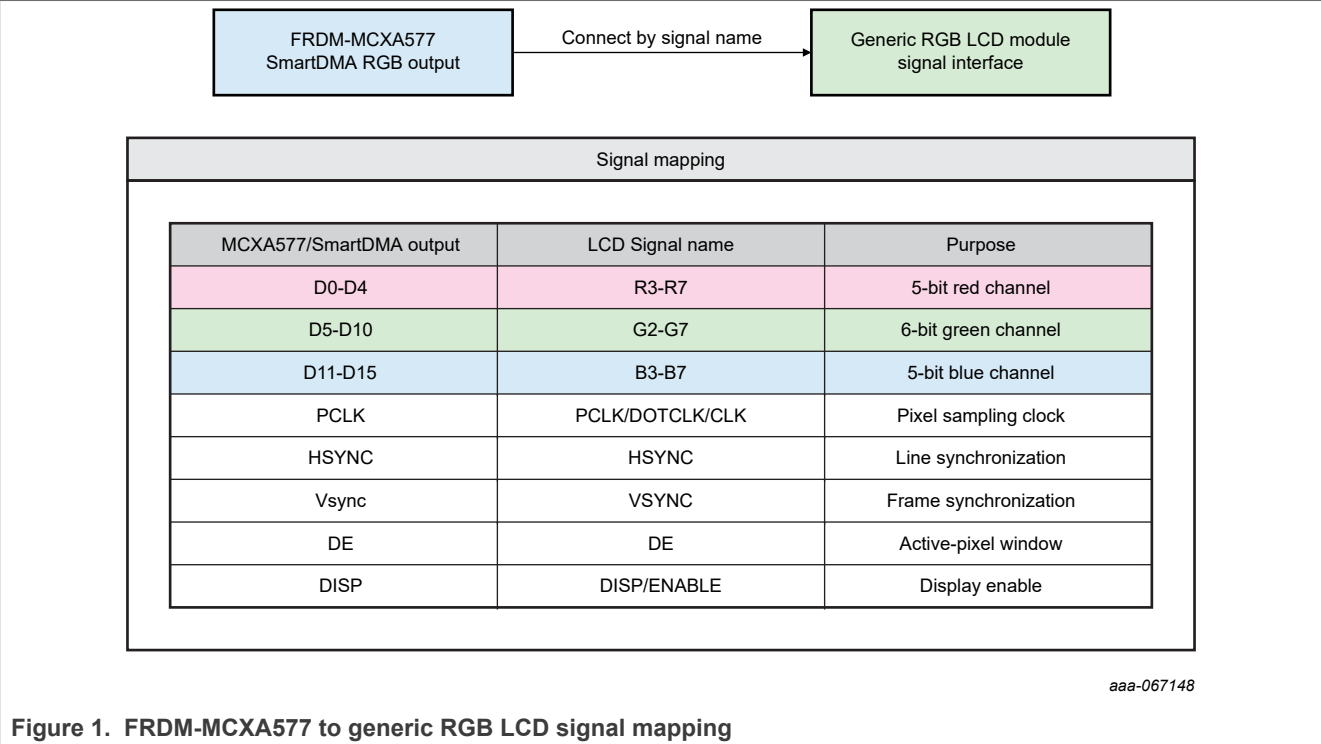
3 Hardware pinout and wiring

This application implements the following hardware:

- 1. FRDM-MCXA577 with type-c USB cable
- 2. K043FN02H-CT LCD module (not public)
- 3. Fly wires

Figure 1 maps the FRDM-MCXA577/SmartDMA outputs to a generic RGB LCD interface. Select any RGB LCD module that meets the electrical and timing requirements, then connect it by signal name.

Implement RGB LCD Interface by SmartDMA on MCX A577



3.1 RGB565 data lines

Table 2. RGB565 data-line mapping

SmartDMA data bit	MCX A577 header	MCU pin/mux	SmartDMA PIO	LCD signal
D0	J9-4	P0_4/ALT7	PIO0	R3
D1	J9-3	P0_5/ALT7	PIO1	R4
D2	J9-6	P3_2/ALT10	PIO2	R5
D3	J9-5	P0_7/ALT7	PIO3	R6
D4	J9-8	P0_14/ALT7	PIO4	R7
D5	J9-7	P0_15/ALT7	PIO5	G2
D6	J9-10	P0_16/ALT7	PIO6	G3
D7	J9-9	P0_17/ALT7	PIO7	G4
D8	J9-2	P0_18/ALT7	PIO8	G5
D9	J8-21	P2_0/ALT7	PIO24	G6
D10	J8-22	P2_1/ALT7	PIO25	G7
D11	J8-23	P2_2/ALT7	PIO26	B3
D12	J8-28	P4_7/ALT7	PIO27	B4
D13	J8-6	P2_4/ALT7	PIO28	B5
D14	J8-26	P2_5/ALT7	PIO29	B6
D15	J8-27	P2_6/ALT7	PIO30	B7

3.2 Timing and display-control lines

Table 3. Display-control signal mapping

Signal	MCX A577 header	MCU pin/mux	SmartDMA PIO	LCD signal
PCLK	J8-19	P0_22/ALT7	PIO12	PCLK/DOTCLK/CLK
HSYNC	J8-20	P0_23/ALT7	PIO13	HSYNC
VSYNC	J8-10	P2_19/ALT7	PIO21	VSYNC
DE	J9-12	P4_2/ALT7	PIO22	DE
DISP	J8-4	P3_20/ALT10	PIO20	DISP/ENABLE

3.3 Power, ground, and backlight

- Logic supply: verify that the I/O voltage of the selected LCD module is compatible with the MCX A577 3.3 V signals. Add level shifting when it is not compatible.
- Ground: the MCX A577 and LCD module must share a low-impedance common ground.
- Backlight: connect LED_A/LED_K or the backlight-control input according to the LCD module data sheet.
- RGB565 uses only R3-R7, G2-G7, and B3-B7. Handle unused lower color inputs as specified by the selected module data sheet.

4 Software

This chapter describes the software implementation of the demo, including:

- Source files and modules that make up the project
- Key application and driver functions
- End-to-end initialization and runtime flow, from board bring-up through SmartDMA startup and the LVGL scheduling loop
- Inventory of the system resources consumed by the solution, including the SmartDMA engine and PIOs, port modules, framebuffer and partial-buffer memory, SysTick, and the debug UART

4.1 Software modules

Table 4. Software modules

File/Module	Responsibility
source/main	System entry, board initialization, SmartDMA startup, periodic LVGL scheduling, and diagnostic output
source/app_smartdma.c/.h	Framebuffer, SmartDMA parameter block, firmware installation/start wrapper, and run/pause API
source/lvgl_support.c/.h	LVGL display registration, 480 x 27 partial buffer, and flush callback
generated/setup_scr_screen.c	Kitchen-timer UI construction and floating animation for the three upper-left bubbles
drivers/fsl_smartdma_fw.c/.h	Aligned 2656-byte firmware array, API index, SRAMX3 installation address, and parameter ABI
board/pin_mux.c	ALT7/ALT10 mux configuration for the 21 SmartDMA RGB LCD output signals
board/clock_config.*	System, bus, and SmartDMA clock configuration
drivers/fsl_smartdma.c/.h	SmartDMA reset/clock initialization, firmware copy, API-table boot, and external-flag control

4.2 Functions

Table 5. Key software functions

Function	File	Description
main()	main	Initializes hardware and display, starts LVGL, and calls lv_timer_handler() on a 5 ms service interval.
BOARD_InitSmartDMA_RGBLCDPins()	pin_mux.c	Enables clocks and releases resets for PORT0/2/3/4, then multiplexes 21 LCD signals to SmartDMA PIO.
RGB_LCD_FillColor()	app_smartdma.c	Fills the full framebuffer with a standard RGB565 color, swapping red and blue fields before writing to match board wiring.
SMARTDMA_RGBLCD_Start()	app_smartdma.c	Prepares parameters, initializes SmartDMA, installs the firmware array, boots API entry 0, and enables frame-boundary execution.
SMARTDMA_RGBLCD_SetRunning()	app_smartdma.c	Uses EXF to control whether the engine continues at the frame boundary; there is no stop-state handshake.
Firmware Image and API Table	drivers/fsl_smartdma_fw.c/.h	Provides the aligned firmware array, firmware size in bytes. Also, provides SRAMX3 installation address, API index, and parameter structure.
LVGL_SetupTick()	lvgl_support.c	Configures the 1 ms LVGL tick and produces the application-side 5 ms task flag.
lv_port_disp_init()	lvgl_support.c	Creates the 480 x 272 display and registers the partial draw buffer and flush callback.
LVGL_FlushDisplay()	lvgl_support.c	Processes dirty regions, swaps R/B per pixel while copying to the live framebuffer, then notifies LVGL that flushing is complete.
setup_ui() / setup_scr_screen()	generated/*.c	Creates the kitchen-timer screen, pot graphic, status cards, and bubble objects.
start_bubble_float()	setup_scr_screen.c	Configures position animation, forward/reverse durations, start delay, infinite repetition, and an ease-in-out path.
set_bubble_y()	setup_scr_screen.c	LVGL animation execution callback that writes the current value to the y-coordinate of the bubble object.

4.3 Initialization and runtime flow

1. BOARD_InitHardware() initializes clocks, the debug UART, and board resources.
2. BOARD_InitSmartDMA_RGBLCDPins() muxes D0-D15, PCLK, HSYNC, VSYNC, DE, and DISP to SmartDMA PIO.
3. RGB_LCD_FillColor(RGB565_BLACK) clears the framebuffer to black.
4. SMARTDMA_RGBLCD_Start() prepares the parameter block, resets and enables SmartDMA, copies the firmware array to SRAMX3. It then boots API entry kSMARTDMA_RGBLCD and sets the external flag to start continuous raster output.
5. lv_init(), LVGL_SetupTick(), and lv_port_disp_init() initialize LVGL and the display port.
6. setup_ui() creates the UI, while events_init() and custom_init() initialize events and user extensions.
7. The main loop waits for the 5 ms task flag and calls lv_timer_handler(); bubble animation triggers partial redraws.

8. `LVGL_FlushDisplay()` copies dirty areas to the full framebuffer while SmartDMA continuously scans it to the LCD in the background.

4.4 System resource inventory

Table 6. System resource inventory

Resource	Use/configuration	Description
SmartDMA	1 engine	Loaded from the firmware array into SRAMX3; Booted through API table entry 0
SmartDMA PIO	21 outputs	16-bit RGB data plus PCLK/HSYNC/VSYNC/DE/DISP
PORT modules	PORT0, PORT2, PORT3, and PORT4	ALT7/ALT10 with fast slew, open-drain disabled, and pulls disabled
Full framebuffer	261120 bytes / 255 kB	480 x 272 x 2, 4-byte aligned
LVGL partial buffer	25920 bytes / 25.31 kB	480 x 27 x 2, PARTIAL mode
SmartDMA stack/parameters	64 bytes + 8 bytes	Stack and framebuffer addresses are passed through the SmartDMA parameter interface
SmartDMA firmware image	2656 bytes total	12-word API table plus 2608-byte binary copied to 0x04018000-0x04018A5F
SysTick	1 ms tick/5 ms service interval	Drives the LVGL tick and main-loop scheduling
Debug UART	LPUART1, 12 MHz source, 115200 bit/s	Per-flush logging may affect animation timing

5 Key implementation details

This chapter documents the parts of the design that must be preserved when porting or rebuilding the demo:

- LCD timing parameters and theoretical waveforms
- SmartDMA firmware array package and its API table
- Firmware installation sequence in SRAMX3 and SmartDMA startup procedure
- LVGL flush callback with its R/B color swap
- Floating animation of the three upper-left bubbles
- SRAM/SRAMX3 memory layout
- Bandwidth and performance budget shared between SmartDMA scan-out and Arm LVGL writes

5.1 LCD timing parameters

Table 7. LCD timing parameters

Direction	Sync width	Back porch	Active area	Front porch	Total
Horizontal/PCLK	2	41	480	8	531 PCLK/line
Vertical/Lines	10	7	272	4	293 lines/frame

The installed firmware image generates 531 PCLK periods per line and 293 lines per frame for this panel configuration. Measure the PCLK on the target board and calculate the corresponding line and frame rates.

Use the logic-analyzer measurements as the final reference for the PCLK, line rate, and frame rate. The geometry below describes the validated firmware configuration but does not replace measurements on the selected panel.

5.1.1 Theoretical waveforms

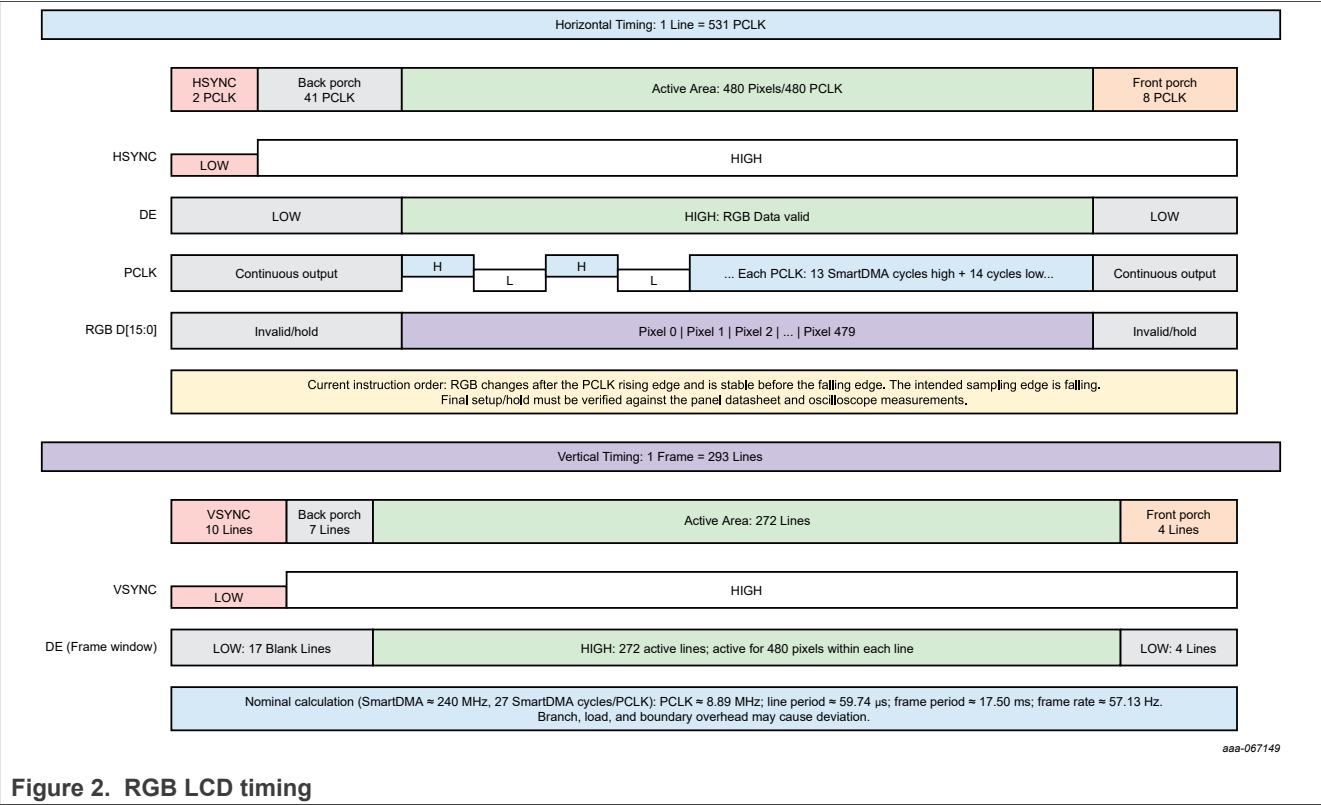


Figure 2. RGB LCD timing

5.2 Firmware array package

The `drivers/fsl_smartdma_fw.c` provides a 4-byte-aligned `uint32_t` firmware array. The 2656-byte image contains a 12-word API table followed by the 2608-byte RGB LCD firmware binary.

At runtime the complete image is copied to SRAMX3. The API index `kSMARTDMA_RGBLCD` selects table entry 0. Keep the array aligned, preserve the API-table order, and regenerate the binary through a verified build/export flow rather than editing firmware words manually.

5.3 Installing the firmware and starting SmartDMA

`SMARTDMA_RGBLCD_Start()` first assigns the dedicated SmartDMA stack and RGB565 framebuffer to `s_smartdmaParam`. `SMARTDMA_InitWithoutFirmware()` resets SmartDMA RAM and enables the SmartDMA clock. `SMARTDMA_InstallFirmware()` then copies the firmware array to SRAMX3, enables synchronized PIO input sampling, and records 0x04018000 as the API-table base.

`SMARTDMA_Boot()` provides the parameter-block address to SmartDMA and reads the firmware entry address from API table entry `kSMARTDMA_RGBLCD`. It then writes the protected control value that enables synchronization and starts execution. `SMARTDMA_SetExternalFlag(1U)` is called last because the raster firmware checks the external flag at each frame boundary.

5.4 LVGL flush and color swap

`lv_port_disp_init()` creates a 480 x 272 display and configures a 480 x 27 RGB565 draw buffer for `LV_DISPLAY_RENDER_MODE_PARTIAL`. `LVGL_FlushDisplay()` clips the area to the screen,

copies the dirty rectangle line by line while swapping the R/B fields, then executes `__DSB()` and `lv_display_flush_ready()`.

The current flush callback prints diagnostics for every transfer. This helps observe animated dirty areas but consumes CPU and UART time; disable or rate-limit it in a production build.

5.5 Floating animation of the three upper-left bubbles

`start_bubble_float()` creates a y-coordinate animation for each bubble. It moves from y to y + travel, reverses over the same duration, repeats indefinitely, and uses an ease-in-out path. Different travel distances, periods, and delays prevent mechanical synchronization.

5.6 Memory layout

Table 8. Memory layout

Object	Size	Region/Purpose
<code>g_rgb565FrameBuffer</code>	261120 bytes	SRAM; full 480 x 272 RGB565 framebuffer
LVGL partial draw buffer	25920 bytes	SRAM; 480 x 27 x 2 partial draw buffer
<code>s_smartdmaStack</code>	64 bytes	SRAM; dedicated SmartDMA stack, 4-byte aligned
<code>s_smartdmaParam</code>	8 bytes	SRAM; stack and framebuffer addresses
Installed SmartDMA firmware image	2656 bytes	SRAMX3; including the API table

The firmware array, API-table entry, installation address, and embedded absolute addresses form one validated image. If the image is rebuilt or relocated, regenerate and revalidate the complete array, then confirm that SRAM and SRAMX3 objects do not overlap.

5.7 Bandwidth and performance budget

This SmartDMA design does not use FlexIO or eDMA. However, it still reads pixels continuously from SRAM and shares the memory system with the Arm LVGL writes. Apply the measured PCLK and frame rate to the following formulas. The resulting values are theoretical bandwidth budgets, not measured MCX A577 bus throughput.

Table 9. Bandwidth and performance budget

Item	Calculation	Theoretical Result
Equivalent RGB pin data rate	PCLK x 2 bytes	Measured PCLK x 2 bytes/s
Active-pixel read rate	261120 bytes x measured frames/s	Calculate from measured frame rate
Active-area fraction of full raster	(480 x 272) / (531 x 293)	Approximately 83.92 %
Full framebuffer	480 x 272 x 2	261120 bytes / 255 kB
LVGL partial buffer	480 x 27 x 2	25920 bytes / 25.31 kB
Known display-buffer total	framebuffer + partial + stack + parameters	Approximately 280.38 kB

Measure CPU utilization, flush time, UART traffic, and tearing under three loads: a static screen, a small-bubble animation, and a full-screen animation. Compare results with per-flush PRINTF enabled and disabled. For a higher-resolution or higher-frame-rate panel, first estimate active-pixel bandwidth as width x height x 2 x frame rate. Afterward, reserve margin for Arm writes, bus arbitration, and boundary overhead.

6 Build, flash, and run

This chapter describes the development environment and tools required to reproduce the demo. It also provides step-by-step instructions for:

- Building the project
- Downloading and running the application on the FRDM-MCXA577 board with MCUXpresso IDE
- Verifying the expected serial output
- Verifying the expected onscreen behavior after startup

6.1 Development environment

- MCUXpresso IDE 25.6.136.
- Project: the code project (an-mcxa577-rgb565-lcd-interface-by-smartdma) can be found at: <https://github.com/nxp-appcodehub>.
- Board: FRDM-MCXA577; debugger connected over USB.

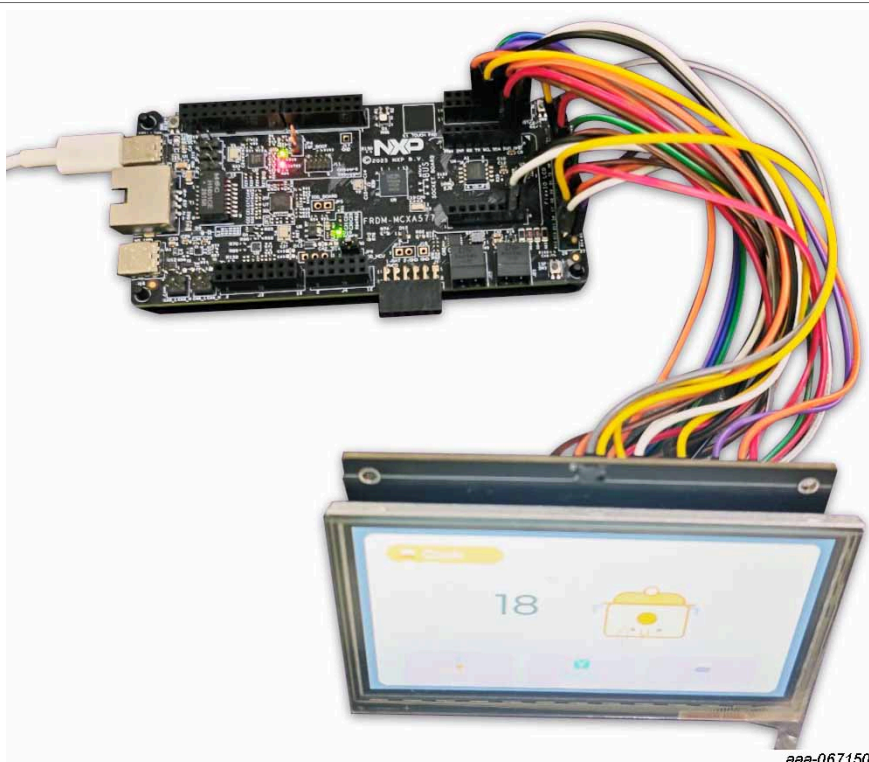
6.2 Build

1. Import `an-mcxa577-rgb565-lcd-interface-by-smartdma` into MCUXpresso IDE.
2. Select the appropriate debug configuration.
3. Click **Build Project** to build the application.

6.3 Download and run

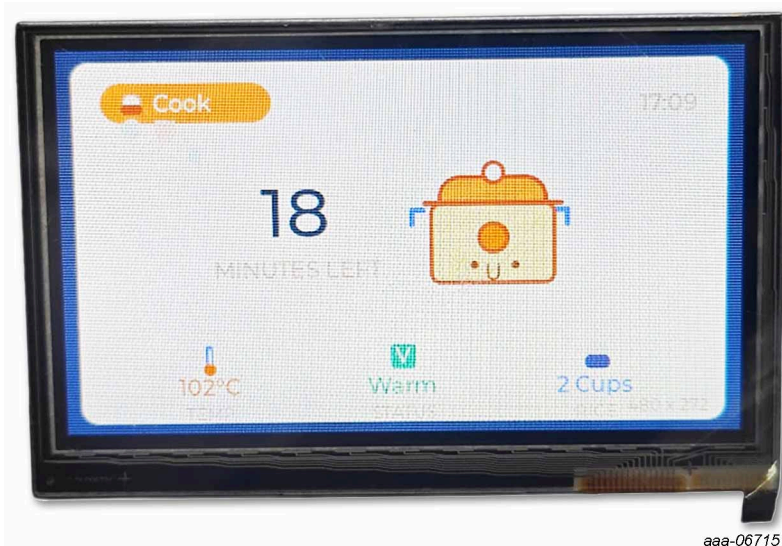
Connect the FRDM-MCXA577 board, select **an-mcxa577-rgb565-lcd-interface-by-smartdma** in MCUXpresso IDE, and click **Debug** to build, download, and run the application. Confirm that the LCD starts normally and displays the demo screen.

The hardware displays the kitchen-timer UI steadily. The screen has a dark-blue background and a light rounded main panel. An orange **Cook** label appears above a central countdown with a pot graphic. Temperature, status, and capacity cards are displayed along the bottom of the screen. The red, green, and blue channels are correct, which confirms the RGB/BGR field swap and all 16 data lines. The three upper-left bubbles continuously float up and down.



aaa-067150

Figure 3. The demo hardware



aaa-067151

Figure 4. The display output

The serial console output during startup and rendering is:

```
MCUX SDK version: ...  
LVGL 9.4 with SmartDMA RGB LCD: 480x272 RGB565.  
LVGL heap after UI: used=...  
LVGL flush N: (x1,y1)-(x2,y2) begin  
LVGL flush N: complete
```

7 Conclusion

The application note implements the complete path from firmware-array installation and SmartDMA RGB raster output to LVGL partial refresh and a dynamic UI. Without a dedicated LCD controller, FlexIO, or eDMA, it drives a 480 x 272 RGB565 panel steadily and demonstrates asynchronous floating animation for the three upper-left bubbles.

This project is a baseline for MCX A577 SmartDMA RGB LCD bring-up, firmware-array loading, LVGL integration, and future double-buffer optimization. Key porting tasks are to verify pinout and bus order, select a matching firmware/timing package, preserve the API and parameter ABI, and remeasure the output waveforms.

8 Acronyms

Table 10. Acronyms

Acronym	Definition
API	Application Programming Interface
BGR	Blue-Green-Red color order
DMA	Direct Memory Access
eDMA	Enhanced Direct Memory Access
FRDM	Freedom Development platform
HSYNC	Horizontal Synchronization
LCD	Liquid Crystal Display
LVGL	Light and Versatile Graphics Library
MCU	Microcontroller Unit
PCLK	Pixel Clock
PIO	Programmable Input/Output
RGB	Red-Green-Blue
RGB565	16-bit RGB pixel format (5 red, 6 green, 5 blue bits)
SmartDMA	Smart Direct Memory Access
SRAM	Static Random Access Memory
SRAMX3	SmartDMA executable SRAM region used for firmware storage
SysTick	System Tick Timer
UART	Universal Asynchronous Receiver/Transmitter
UI	User Interface
USB	Universal Serial Bus
VSYNC	Vertical Synchronization

9 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2026 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
- 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

10 Revision history

[Table 11](#) summarizes the revisions to this document.

Table 11. Revision history

Document ID	Release date	Description
AN15129 v.1.0	30 September 2026	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at <https://www.nxp.com/profile/terms>, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, μ Vision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Contents

1	Overview	2
1.1	Key features	2
1.2	Scope	2
2	Fundamentals	2
2.1	Parallel RGB LCD interface	2
2.2	RGB565 pixel format	3
2.3	Framebuffer and partial rendering	3
2.4	Role of SmartDMA	3
3	Hardware pinout and wiring	3
3.1	RGB565 data lines	4
3.2	Timing and display-control lines	5
3.3	Power, ground, and backlight	5
4	Software	5
4.1	Software modules	5
4.2	Functions	6
4.3	Initialization and runtime flow	6
4.4	System resource inventory	7
5	Key implementation details	7
5.1	LCD timing parameters	7
5.1.1	Theoretical waveforms	8
5.2	Firmware array package	8
5.3	Installing the firmware and starting SmartDMA	8
5.4	LVGL flush and color swap	8
5.5	Floating animation of the three upper-left bubbles	9
5.6	Memory layout	9
5.7	Bandwidth and performance budget	9
6	Build, flash, and run	10
6.1	Development environment	10
6.2	Build	10
6.3	Download and run	10
7	Conclusion	12
8	Acronyms	12
9	Note about the source code in the document	12
10	Revision history	13
	Legal information	14

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.