USER'S MANUAL

Nema DC API Library

An Overview

A-SOCAPG-UMGA01EN v1.0

Legal Information and Disclaimers

AMBIQ MICRO INTENDS FOR THE CONTENT CONTAINED IN THE DOCUMENT TO BE ACCURATE AND RELIABLE. THIS CONTENT MAY, HOWEVER, CONTAIN TECHNICAL INACCURACIES, TYPOGRAPHICAL ERRORS OR OTHER MISTAKES. AMBIQ MICRO MAY MAKE CORRECTIONS OR OTHER CHANGES TO THIS CONTENT AT ANY TIME. AMBIQ MICRO AND ITS SUPPLIERS RESERVE THE RIGHT TO MAKE CORRECTIONS, MODIFICATIONS, ENHANCEMENTS, IMPROVEMENTS AND OTHER CHANGES TO ITS PRODUCTS, PROGRAMS AND SERVICES AT ANY TIME OR TO DISCONTINUE ANY PRODUCTS, PROGRAMS, OR SERVICES WITHOUT NOTICE.

THE CONTENT IN THIS DOCUMENT IS PROVIDED "AS IS". AMBIQ MICRO AND ITS RESPECTIVE SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THIS CONTENT FOR ANY PURPOSE AND DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS CONTENT, INCLUDING BUT NOT LIMITED TO, ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHT.

AMBIQ MICRO DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT OF AMBIQ MICRO COVERING OR RELATING TO THIS CONTENT OR ANY COMBINATION, MACHINE, OR PROCESS TO WHICH THIS CONTENT RELATE OR WITH WHICH THIS CONTENT MAY BE USED.

USE OF THE INFORMATION IN THIS DOCUMENT MAY REQUIRE A LICENSE FROM A THIRD PARTY UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF THAT THIRD PARTY, OR A LICENSE FROM AMBIQ MICRO UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF AMBIQ MICRO.

INFORMATION IN THIS DOCUMENT IS PROVIDED SOLELY TO ENABLE SYSTEM AND SOFTWARE IMPLEMENTERS TO USE AMBIQ MICRO PRODUCTS. THERE ARE NO EXPRESS OR IMPLIED COPYRIGHT LICENSES GRANTED HEREUNDER TO DESIGN OR FABRICATE ANY INTEGRATED CIRCUITS OR INTEGRATED CIRCUITS BASED ON THE INFORMATION IN THIS DOCUMENT. AMBIQ MICRO RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN. AMBIQ MICRO MAKES NO WARRANTY, REPRESENTATION OR GUARANTEE REGARDING THE SUITABILITY OF ITS PRODUCTS FOR ANY PARTICULAR PURPOSE, NOR DOES AMBIQ MICRO ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT, AND SPECIFICALLY DISCLAIMS ANY AND ALL LIABILITY, INCLUDING WITHOUT LIMITATION CONSEQUENTIAL OR INCIDENTAL DAMAGES. "TYPICAL" PARAMETERS WHICH MAY BE PROVIDED IN AMBIQ MICRO DATA SHEETS AND/OR SPECIFICATIONS CAN AND DO VARY IN DIFFERENT APPLICATIONS AND ACTUAL PERFORMANCE MAY VARY OVER TIME. ALL OPERATING PARAMETERS, INCLUDING "TYPICALS" MUST BE VALIDATED FOR EACH CUSTOMER APPLICATION BY CUSTOMER'S TECHNICAL EXPERTS. AMBIQ MICRO DOES NOT CONVEY ANY LICENSE UNDER NEITHER ITS PATENT RIGHTS NOR THE RIGHTS OF OTHERS. AMBIQ MICRO PRODUCTS ARE NOT DESIGNED, INTENDED, OR AUTHORIZED FOR USE AS COMPONENTS IN SYSTEMS INTENDED FOR SURGICAL IMPLANT INTO THE BODY, OR OTHER APPLICATIONS INTENDED TO SUPPORT OR SUSTAIN LIFE, OR FOR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE AMBIQ MICRO PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR. SHOULD BUYER PURCHASE OR USE AMBIQ MICRO PRODUCTS FOR ANY SUCH UNINTENDED OR UNAUTHORIZED APPLICATION, BUYER SHALL INDEMNIFY AND HOLD AMBIQ MICRO AND ITS OFFICERS, EMPLOYEES, SUBSIDIARIES, AFFILIATES, AND DISTRIBUTORS HARMLESS AGAINST ALL CLAIMS, COSTS, DAMAGES, AND EXPENSES, AND REASONABLE ATTORNEY FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PERSONAL INJURY OR DEATH ASSOCIATED WITH SUCH UNINTENDED OR UNAUTHORIZED USE, EVEN IF SUCH CLAIM ALLEGES THAT AMBIQ MICRO WAS NEGLIGENT REGARDING THE DESIGN OR MANUFACTURE OF THE PART.

Revision History

Rev	Date	Description
1.0	September 2025	Initial Release

Reference Documents

Document ID	Description
A-SOCAPG-UMGA02EN	Nema GFX API Library User's Manual
A-SOCAPG-UMGA03EN	Nema GFX Extensions Vector Graphics User's Manual
A-SOCAPG-ANGA01EN	Nema DC MiP Panels Configuration Application Note
A-SOCAPG-ANGA02EN	Nema GFX Extensions TSVG Supported Elements List Application Note
A-SOCAPG-SGGA01EN	Nema Pix-presso Starting Guide
A-SOCAPG-ANGA03EN	Nema GFX API Debugging Application Note
A-SOCAPG-UMGA04EN	Nema GUI-builder User's Manual
A-SOCAPG-UMGA05EN	Nema GFX Benchmark Suite User's Manual
A-SOCAPG-UMGA06EN	Nema Pico Graphics Processing Unit User's Manual
A-SOCAPG-UMGA07EN	Nema Pico Platform Drivers User's Manual

Table of Contents

List of Tables

List of Figures

SECTION

1

Preface

This manual intends to present and describe the basic functionalities and capabilities of the Nema DC API Library, an API for driving and configuring Nema DC.

Throughout the manual, the user will be provided with examples of proper library usage along with general information on its overall operation scheme.

SECTION

2

Nema DC-API Architecture

Nema DC API is a low level library that interfaces directly with the Nema DC display controller. It consists of a software framework designed to control and manage the different layers of the Nema DC with ease and efficiency.

The target of Nema DC API is to abstract the functionality of the Nema DC, providing the user with the ability to leverage the Nema DC functionality. Nema DC API allows great performance with minimum CPU/MCU usage and power consumption, since framebuffer composition is done at Hardware level. Besides this, its small footprint, efficient design, and lack of any external dependencies, makes it ideal for use in embedded applications.

Figure 2-1: Nema DC API Architecture

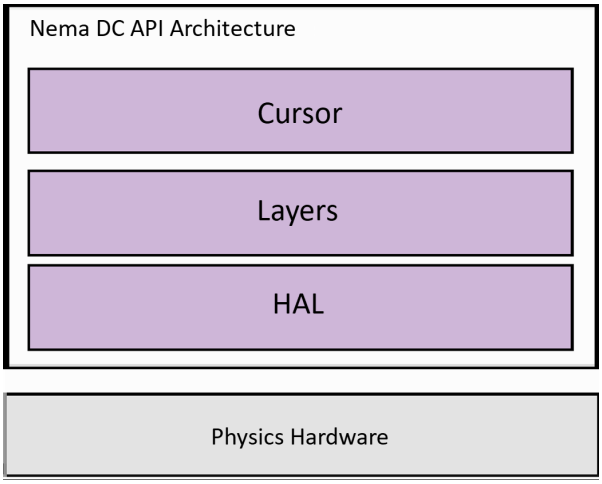


Figure 2-1 presents the software stack of Nema DC API that follows a modular architecture.

The lowest module consists of a thin **Hardware Abstraction Layer (HAL)**, which includes some hooks for basic interfacing with the hardware such as register accessing and vsync handling.

On top of the HAL lies the **Layer Manager**. Its main functionality is to provide function calls to set video layer information. Video layers can be scaled, clipped, positioned, and composed on the final display independently.

The next layer is the **Cursor Manager** which is responsible for driving the cursor pointer in case it is available.

2.1 Directory Structure

Nema DC API is located inside the **NemaGFX_SDK** directory. The HAL is platform specific and its implementation can be found inside the **NemaGFX_SDK/platforms/** directory. Inside this directory there are different implementations for each available platform 2 Nema DC API Architecture (e.g., baremetal generic, zc70x linux etc.). HAL is discussed in more details in *Section 4 Nema DC-API Portable Implementation on page 16*. The **NemaGFX_SDK/include/tsi/** directory contains the header files that define the interface of Nema DC API. Last but not least, the implementation of the functions defined inside the previously mentioned header files, that are platform-independent (Layer and Cursor Manager) can be found inside the **NemaGFX_SDK/NemaDC/** directory.

SECTION

3

Quick Start Guide

The Quick Start Guide for the Nema DC API provides simple and comprehensive guidelines on how to use the Library for developing purposes. The information included in this section can be found elsewhere in this document, but span across several paragraphs, which makes it daunting to start with a simple code example using the Nema DC API Library in a timely manner.

3.1 Initialization and Timing

The first thing to be done is the **ipdc** initialization using:

```
int nemadc_init(void)
```

this initializes the \ipdc and retrieves its configuration by getting access to its register file.

Once the Display Controller is initialized the background color can be set using:

```
void nemadc_set_bgcolor(uint32_t rgba)
```

Right after it, different timing parameters must be set. These parameters include setting display resolution and blanking the front and back porch. Table 1 includes some examples of typical values for different resolutions.

```
void nemadc_timing(int resx, int fpx, int blx, int bpx,
int resy, int fpy, int bly, int bpy)
```

Table 3-1: Timing Parameters for Different Resolutions

Freq	Res X	Res Y	Front Porch X	Front Porch Y	Blankin X	Blankin Y	Back Porch X	Back Porch Y
60Hz	800	600	40	1	40	4	88	23
60Hz	1280	800	64	1	136	3	200	24
60Hz	1920	1080	88	4	44	5	148	36

3.2 Video Layers

Layers are visual elements that are stacked to compose the final displayed image. Nema DC-100 supports one video layer, Nema DC-200 up to two layers and Nema DC-400 up to four layers. These layers can be scaled, cropped, positioned, blended and composed on the final display. They are completely independent to each other, therefore must be set separately. Note that in order to perform layer blending or scaling, Nema DC must be configured accordingly (blender and scaler submodules).

Layers can be enabled and disabled at runtime using the following commands:

```
  void nemadc_layer_enable(int layer_no)
void nemadc_layer_disable(int layer_no)
```

In order to set a layer, a **nemadc_layer_t** struct containing layer's info must be initialized. This struct type contains information regarding the layer, such as resolution, stride, format, etc.

Layers are set as follows:

```
void nemadc_set_layer(int layer_no, nemadc_layer_t *layer)
```

There is also the possibility of setting only the layer's physical address. This is useful in case the developer wants to swap buffers:

```
void nemadc_set_layer_addr(int layer_no, uintptr_t addr)
```

Besides the above, Nema DC implements a programmable global and per-layer Gamma Look Up Table (LUT) for gamma correction. Gamma LUT consists of a 3x256x8 memory array that holds the RGB values for each of the 256 colors in the palette. In order to program it, Nema DC API implements getter and setter calls for both the global and the per layer Gamma LUT.

```
  int  nemadc_get_palette(int index)   void nemadc_set_palette(int
index, int color)

  int  nemadc_get_layer_gamma_lut(int layer_no, int index)   void
nemadc_set_layer_gamma_lut(int layer_no, int index, int color)
```

3.3 VSync

While the Display Controller is busy scanning the current frame from memory, front frame should never be altered. Any modifications to the layer information or address should be done once the Display Controller is done with the frame refresh. To achieve this, Nema DC implements a call that waits for Vertical Synchronization (VSync):

```
void nemadc_wait_vsync(void)
```

3.4 Example

The following example depicts the way that Nema DC API operates. In this scenario, the Nema DC API is initialized and programmed to drive a 800 x 600 display. Two layers dc **layer_0** and **dc_layer1** are defined, as well as two buffer objects are created bo0 and bo1 inside the l**oad_objects(void)** function. The default layer resolution is 300 x 200. After the Nema DC is configured, the positions of the layers (startx, starty) are set to (40, 20) for dc **layer_0** and (140, 150) for dc **layer_1**. The two layers are afterwards scaled by a factor of 20% (their size is multiplied by 1.2). Scaling is performed by changing the layers's **sizex** and **sizey** parameters. Finally, after the previously described steps have been completed, the two layers are set, in order to be displayed on the screen. More specifically, calling the **nemadc_set_layer(0, &dc_layer0)** function, will display **dc_layer0** on the screen. Calling the **nemadc_set_layer(1, &dc_layer1**) will also display **dc_layer1** and place it in front of **dc_layer0**. The lower the **layer_no** argument of the **nemadc_set_layer(int layer_no, nemadc_layer_t *layer**) the backmost the layer will be placed.

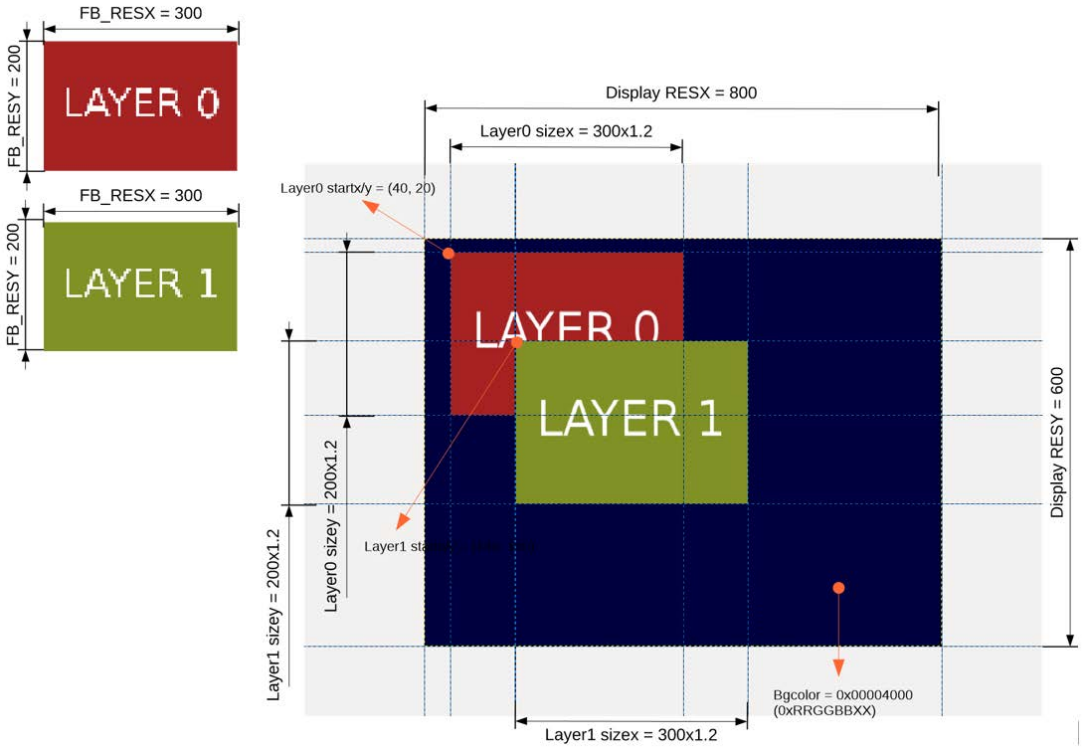
```
#include "nema_core.h"
#include <string.h>

#include "nema_dc.h"

//Textures to be displayed within the layers
#include "layer0.rgba.h"
#include "layer1.rgba.h"

// Display resolution
#define RESX 800
#define RESY 600

// Framebuffer resolution

#define FB_RESX 300   #define FB_RESY 200   static

size_t framebuffer_size = FB_RESX*FB_RESY*4;

  static nemadc_layer_t dc_layer0 = {(void *)0, 0, FB_RESX, FB_RESY,
FB_RESX*4, 0, 0, RESX, RESY, 0xff, NEMADC_BL_SRC, 0, NEMADC_RGBA8888,
0, 0, 0, 0, 0};
```

```
  static nemadc_layer_t dc_layer1 = {(void *)0, 0, FB_RESX, FB_RESY,
FB_RESX*4, 0, 0,  RESX, RESY, 0xff, NEMADC_BL_SRC, 0,
NEMADC_RGBA8888, 0, 0, 0, 0, 0};

  int load_objects(void)
  {
      // Initialize NemaGFX
      // Used only for graphics (contiguous) memory allocations
      if ( nema_init() != 0 ) {
          return -1;
}

      //-----------------------------------------------------
      // Load framebuffers to Contiguous Memory
      //-----------------------------------------------------

      // Allocate buffers

     nema_buffer_t bo0 = nema_buffer_create(framebuffer_size);
nema_buffer_t bo1 = nema_buffer_create(framebuffer_size);


 // memcpy framebuffers to allocated buffers
memcpy(bo0.base_virt, layer0_rgba, framebuffer_size);
memcpy(bo1.base_virt, layer1_rgba, framebuffer_size);

      dc_layer0.baseaddr_phys = bo0.base_phys;
dc_layer0.baseaddr_virt = bo0.base_virt;

      dc_layer1.baseaddr_phys = bo1.base_phys;
dc_layer1.baseaddr_virt = bo1.base_virt;

      return 0;
}

  int main(int argc, char *argv[]) {
      //Initialize NemaDC       if ( nemadc_init() != 0 )
          return -2;
}

      // Load framebuffers to Contiguous Memory space
      if ( load_objects() != 0 ) {
          return -3;
}

      //Format       |Pixclock|RESX|FP|SYNC|BP|RESY|FP|SYNC|BP
//800x600, 60Hz|40.000  |800 |40|128 |88|600 |1 |4   |23
nemadc_timing(800, 40, 128, 88, 600, 1, 4, 23);

     // set NemaDC's background color to dark blue
     // Background color is visible when the available layers are not
     // covering the entire display resolution
nemadc_set_bgcolor(0x00004000);

     // set layer's top left corner position (coordinates)
     // (0, 0) is on the top left corner of the display

      dc_layer0.startx = 40;
dc_layer0.starty = 20;
```

```
      dc_layer1.startx = 140;
dc_layer1.starty = 150;

     // ------------------------------------------------------------
     // LAYER SCALING
     // ignored if layer-scaler not enabled in hardware
     // ------------------------------------------------------------
     // - set layer's size on the display (width/height in pixels)
     // - layer's resx/resy correspond to the original framebuffer's
          resolution
     // - layer's sizex/sizey correspond to final scaled dimensions
          in pixels

dc_layer0.sizex = dc_layer0.resx*1.2f;
dc_layer0.sizey = dc_layer0.resy*1.2f;
dc_layer1.sizex = dc_layer1.resx*1.2f;
dc_layer1.sizey = dc_layer1.resy*1.2f;

      // program and enable layers

  nemadc_set_layer(0, &dc_layer0);   nemadc_set_layer(1, &dc_layer1);
     return 0;

}
```

The following figure illustrates the final displayed image along with the parameters that were set during the execution of the previous code sample.

Figure 3-1: Nema DC API Two Layers Example



The full code of this example can be found inside the **NemaGFX_SDK/examples/nema_dc/two_layers** directory.

SECTION

4

Nema DC-API Portable Implementation

Nema DC API library has been designed to be easily portable in a variety of different platforms. This includes systems with or without an operating system. In order to port Nema DC API successfully, one must take into account the target platform and adapt the HAL (Hardware Abstraction Layer).

The HAL is a thin layer of the library that:

- Communicates directly with the system hardware and the DC driver (when drivers are available)
- Performs the communication between the host and the DC (access to HW registers)
- Handles Vertical Synchronization (VSync)

4.1 Platform Specific HAL

Each target platform has a separate **nemadc_hal.c** file, located in **platforms/<device>/NemaDC** folder. In order to port Nema DC API to a new platform, a nemadc_hal.c must be implemented for the corresponding platform. It is advised to use the source files of an already ported platform as a template.

4.2 nemadc_hal.c

nemadc_hal.c contains all the platform specific functions responsible for hardware initialization, read/write operations, and vsync interrupt handling. It acts as a thin layer which incorporates all the platform dependent portions of a Nema DC API implementation.

4.2.1 System Initialization

The **nemadc_init()** function initializes the Nema DC API and calls the **nemadc_sys_init()** function which is responsible for the system initialization. The system initialization simply consists of mapping the Nema DC registers into memory.

4.2.2 Registers Read/Write

The host CPU writes to and reads from the Nema DC configuration registers. The functions **nemadc_reg_read()** and **nemadc_reg_write()** are used for the communication of the CPU with the Display Controller.

When the target platform does not support virtual memory (e.g., in BareMetal systems), the access to the Nema DC's registers is straightforward by using the register's physical memory address. The only prerequisite in this case, is the appropriate memory mapping of the display registers to the system's main memory.

The following examples illustrate the register read and write operations for BareMetal systems:

```
uint32_t  nemadc_reg_read(uint32_t reg) {
     uint32_t *ptr = (uint32_t *)(nemadc_regs + reg);
     return *ptr;
}

void nemadc_reg_write(uint32_t reg, uint32_t value) {
uint32_t *ptr = (uint32_t *)(nemadc_regs + reg);
    *ptr = value;

}
```

On platforms that support virtual memory (e.g., Linux, Android), the registers' physical addresses must be mapped to the virtual memory addresses before a register access is attempted. This can be performed in three ways.

The first one is through the device driver (if applicable) and perform the memory mapping using the mmap system call.

```
nemadc_regs_base_virt = mmap(NULL, 0x1000, PROT_READ |
PROT_WRITE, MAP_SHARED, nemad c_fd, 0);
```

The second way in Linux systems, is to use the **/dev/mem** instead of the **nema_dc** driver as shown in the following example.

```
 // Memory map registers
nema_devmem_mmap((void **)&nemadc_regs, NEMADC_BASEADDR, 0x2000,
"NemaDCRegisters");
```

The third way in Linux systems is to perform the read/write operations to the registers via respective IOCTL calls to the Nema DC driver.

```
  uint32_t  nemadc_reg_read(uint32_t reg) {     unsigned long
lcdarg[2];     lcdarg[0] = reg;

    ioctl(nemadc_fd, 0x45, lcdarg);
    return lcdarg[1]; }

void nema_reg_write(uint32_t reg, uint32_t value) {    unsigned long
lcdarg[2];

    lcdarg[0] = reg;     lcdarg[1] = value;
    ioctl(nemadc_fd, 0x44, lcdarg);
}
```

4.2.3 VSync Handling

The interrupt handler is triggered when an interrupt is issued by the Display Controller. Its purpose is to awaken suspended processes and clear the interrupt.

For platforms like Linux and Android, the interrupt handler is implemented within the Nema DC driver. For the rest of the platforms (BareMetal and RTOS based systems), an interrupt handler for clearing the interrupt has to be implemented in the **nemadc_hal.c** file.

A typical interrupt handler for BareMetal systems is the following:

```
   void nemadc_interrupt_handler (void)
   {
     //Clear the interrupt
     nemadc_reg_write(NEMADC_REG_INTERRUPT, 0);
   }
```

The HAL must also implement the **nemadc_wait_vsync** function that waits for a vsync signal. Its purpose is to suspend the process (put to sleep) until the DC finishes its current refresh cycle, continuing the execution when done, in order to avoid screen tearing or visual artifacts. Its implementation is platform (CPU) dependent. Each vsync must be requested by the user first. This is done by writing to the **NEMADC_REG_INTERRUPT** a non-zero value. If the kernel driver is available, then IOCTL calls are used:

```
  void nemadc_wait_vsync(void) {
    nemadc_reg_write(NEMADC_REG_INTERRUPT, 1); // Request VSync
    ioctl(nemadc_fd, _IOW('F', 0x20, __u32));  // Wait for VSync
  }
```

Otherwise it is manually defined according to the CPU platform. A typical implementation would be to active wait, using a while loop, until the **NEMADC_REG_INTERRUPT** is cleared:

```
  void nemadc_wait_vsync(void) {
    nemadc_reg_write(NEMADC_REG_INTERRUPT, 1);  // Request VSync
while(nemadc_reg_read(NEMADC_REG_INTERRUPT) != 0);  // Wait for VSync
  }
```

SECTION

5

Nema DC API Library Functions

This section provides an overview of the implemented functions of the Nema DC API Library.

5.1 Files

Here is a list of all files with brief descriptions:

5.1.1 nema_dc.h File

```
#include "nema_sys_defs.h"
#include "nema_dc_hal.h"
```

Table 5-1: nema_dc.h File Data Structure

Data	Structures
struct	nemadc_display_t More...
struct	nemadc_layer_t More...

5.1.1.1 Macros

```
#define COLORFORM_REG_VERSION
#define NEMADC_CFG_LAYER_EXISTS
#define NEMADC_CFG_LAYER_BLENDER
#define NEMADC_CFG_LAYER_SCALER
#define NEMADC_CFG_LAYER_GAMMA
#define NEMADC_COLMOD_LAYER_FLIPX
```

```
#define NEMADC_COLMOD_FLIPY
#define NEMADC_LAYER_DISABLE
#define NEMADC_LAYER_ENABLE
#define NEMADC_FORCE_A
#define NEMADC_SCALE_NN
#define NEMADC_MODULATE_A
#define NEMADC_LAYER_AHBLOCK
#define NEMADC_LAYER_GAMMALUT_EN
#define NEMADC_LAYER_REG_PROTECT
#define NEMADC_UNDERFLOW_MECH
#define NEMADC_BF_ZERO
#define NEMADC_BF_ONE
#define NEMADC_BF_SRCALPHA
#define NEMADC_BF_GLBALPHA
#define NEMADC_BF_SRCGBLALPHA
#define NEMADC_BF_INVSRCALPHA
#define NEMADC_BF_INVGBLALPHA
#define NEMADC_BF_INVSRCGBLALPHA
#define NEMADC_BF_DSTALPHA
#define NEMADC_BF_INVDSTALPHA
#define NEMADC_BL_SIMPLE
#define NEMADC_BL_CLEAR
#define NEMADC_BL_SRC
#define NEMADC_BL_SRC_OVER
#define NEMADC_BL_DST_OVER
#define NEMADC_BL_SRC_IN
       NEMADC_BL_DST_IN
       NEMADC_BL_SRC_OUT
          NEMADC_BL_DST_OUT
          NEMADC_BL_SRC_ATOP
#define NEMADC_BL_DST_ATOP
#define NEMADC_BL_ADD
#define NEMADC_BL_XOR
#define NEMADC_LUT8
#define NEMADC_RGBA8888
#define NEMADC_BGRA8888
#define NEMADC_ARGB8888
#define NEMADC_ABGR8888
#define NEMADC_RGB24
#define NEMADC_BGR24
#define NEMADC_RGBA4444
#define NEMADC_BGRA4444
#define NEMADC_ARGB4444
#define NEMADC_ABGR4444
```

```
#define NEMADC_RGBA5551
#define NEMADC_BGRA5551
#define NEMADC_ARGB1555
#define NEMADC_ABGR1555
#define NEMADC_RGB565
#define NEMADC_BGR565
#define NEMADC_AL88
#define NEMADC_AL44
#define NEMADC_RGB332
#define NEMADC_RGBA2222
#define NEMADC_BGRA2222
#define NEMADC_ARGB2222
#define NEMADC_ABGR2222
#define NEMADC_A8
#define NEMADC_L8
#define NEMADC_L1
#define NEMADC_L4
#define NEMADC_TSC12
#define NEMADC_TSC12A
#define NEMADC_YUYV
#define NEMADC_YUY2
#define NEMADC_V_YUV420
#define NEMADC_TLYUV420
#define NEMADC_TSC4
#define NEMADC_TSC6
#define NEMADC_TSC6A
#define NEMADC_TSC6AP
#define NEMADC_L4_LE
       NEMADC_DISABLE
       NEMADC_ENABLE
       NEMADC_CURSOR
       NEMADC_NEG_V
#define NEMADC_NEG_H
#define NEMADC_NEG_DE
#define NEMADC_DITHER
#define NEMADC_DITHER16
#define NEMADC_DITHER15
#define NEMADC_SINGLEV
#define NEMADC_INVPIXCLK
#define NEMADC_PALETTE
#define NEMADC_GAMMA
#define NEMADC_BLANK
#define NEMADC_INTERLACE
#define NEMADC_ONE_FRAME
```

```
#define NEMADC_P_RGB3_18B
#define NEMADC_P_RGB3_18B1
#define NEMADC_P_RGB3_16B
#define NEMADC_P_RGB3_16B1
#define NEMADC_P_RGB3_16B2
#define NEMADC_CLKOUTDIV
#define NEMADC_LVDSPADS
        NEMADC_YUVOUT
        NEMADC_MIPI_OFF
        NEMADC_OUTP_OFF
        NEMADC_LVDS_OFF
#define NEMADC_SCANDOUBLE
#define NEMADC_TESTMODE
#define NEMADC_P_RGB3
#define NEMADC_S_RGBX4
#define NEMADC_S_RGB3
#define NEMADC_S_12BIT
#define NEMADC_LVDS_ISP68
#define NEMADC_MIP_IF
#define NEMADC_LVDS_ISP8
#define NEMADC_T_16BIT
#define NEMADC_BT656
#define NEMADC_JDIMIP
#define NEMADC_IPI
#define NEMADC_CFG_PALETTE
#define NEMADC_CFG_FIXED_CURSOR
#define NEMADC_CFG_PROGR_CURSOR
#define NEMADC_CFG_DITHERING
#define NEMADC_CFG_FORMAT
#define NEMADC_CFG_HiQ_YUV
        NEMADC_CFG_DBIB
        NEMADC_CFG_YUVOUT
        NEMADC_CFG_L0_ENABLED
        NEMADC_CFG_L0_BLENDER
#define NEMADC_CFG_L0_SCALER
#define NEMADC_CFG_L0_GAMMA
#define NEMADC_CFG_L1_ENABLED
#define NEMADC_CFG_L1_BLENDER
#define NEMADC_CFG_L1_SCALER
#define NEMADC_CFG_L1_GAMMA
#define NEMADC_CFG_L2_ENABLED
#define NEMADC_CFG_L2_BLENDER
#define NEMADC_CFG_L2_SCALER
#define NEMADC_CFG_L2_GAMMA
```

```
#define NEMADC_CFG_L3_ENABLED
#define NEMADC_CFG_L3_BLENDER
#define NEMADC_CFG_L3_SCALER
#define NEMADC_CFG_L3_GAMMA
#define NEMADC_CFG_SPI
#define NEMADC_CFG_IPI
#define NEMADC_CFG_L0_YUVMEM
#define NEMADC_CFG_L1_YUVMEM
#define NEMADC_CFG_L2_YUVMEM
#define NEMADC_CFG_L3_YUVMEM
#define NEMADC_CFG_L0_TSCMEM
#define NEMADC_CFG_L1_TSCMEM
#define NEMADC_CFG_L2_TSCMEM
#define NEMADC_CFG_L3_TSCMEM
#define NEMADC_EN_PIXCLK
#define NEMADC_EN_CFCLK
#define NEMADC_EN_L0BUS
#define NEMADC_EN_L0PIX
#define NEMADC_EN_L1BUS
#define NEMADC_EN_L1PIX
#define NEMADC_EN_L2BUS
#define NEMADC_EN_L2PIX
#define NEMADC_EN_L3BUS
#define NEMADC_EN_L3PIX
#define DC_STATUS_rsrvd_0
#define DC_STATUS_rsrvd_1
#define DC_STATUS_rsrvd_2
#define DC_STATUS_rsrvd_3
#define DC_STATUS_rsrvd_4
#define DC_STATUS_rsrvd_5
#define DC_STATUS_rsrvd_6
#define DC_STATUS_rsrvd_7
#define DC_STATUS_rsrvd_8
#define DC_STATUS_rsrvd_9
#define DC_STATUS_rsrvd_10
#define DC_STATUS_rsrvd_11
#define DC_STATUS_rsrvd_12
#define DC_STATUS_gpi_stuck
#define DC_STATUS_crc_ready
#define DC_STATUS_dbi_rd_wr_on
#define DC_STATUS_dbi_cmd_ready
#define DC_STATUS_dbi_cs
#define DC_STATUS_frame_end
```

```
#define DC_STATUS_dbi_pending_trans
#define DC_STATUS_dbi_pending_cmd
#define DC_STATUS_dbi_pending_data
#define DC_STATUS_mmu_error
#define DC_STATUS_te
#define DC_STATUS_sticky
#define DC_STATUS_underflow
#define DC_STATUS_LASTROW
#define DC_STATUS_DPI_Csync
#define DC_STATUS_vsync_te
#define DC_STATUS_hsync
#define DC_STATUS_framegen_busy
#define DC_STATUS_ACTIVE
#define DC_STATUS_dbi_busy
#define NemaDC_clkctrl_cg_l3_bus_clk
#define NemaDC_clkctrl_cg_l3_pix_clk
#define NemaDC_clkctrl_cg_l2_bus_clk
#define NemaDC_clkctrl_cg_l2_pix_clk
#define NemaDC_clkctrl_cg_l1_bus_clk
#define NemaDC_clkctrl_cg_l1_pix_clk
#define NemaDC_clkctrl_cg_l0_bus_clk
#define NemaDC_clkctrl_cg_l0_pix_clk
#define NemaDC_clkctrl_cg_regfil_clk
#define NemaDC_clkctrl_cg_bypass_clk
#define NemaDC_clkctrl_cg_rsrvd_21
#define NemaDC_clkctrl_cg_rsrvd_20
#define NemaDC_clkctrl_cg_rsrvd_19
#define NemaDC_clkctrl_cg_rsrvd_18
#define NemaDC_clkctrl_cg_rsrvd_17
#define NemaDC_clkctrl_cg_rsrvd_16
#define NemaDC_clkctrl_cg_rsrvd_15
#define NemaDC_clkctrl_cg_rsrvd_14
#define NemaDC_clkctrl_cg_rsrvd_13
#define NemaDC_clkctrl_cg_rsrvd_12
#define NemaDC_clkctrl_cg_rsrvd_11
#define NemaDC_clkctrl_cg_rsrvd_10
#define NemaDC_clkctrl_cg_rsrvd_9
#define NemaDC_clkctrl_cg_rsrvd_8
#define NemaDC_clkctrl_cg_rsrvd_7
#define NemaDC_clkctrl_cg_rsrvd_6
#define NemaDC_clkctrl_cg_rsrvd_5
#define NemaDC_clkctrl_cg_rsrvd_4
#define NemaDC_clkctrl_cg_rsrvd_3
```

```
#define NemaDC_clkctrl_cg_clk_swap
#define NemaDC_clkctrl_cg_clk_inv
#define NemaDC_clkctrl_cg_clk_en
```

5.1.1.2 Functions

int nemadc_init(void)

Initialize NemaDC library.

uint32_t nemadc_get_config(void)

Read Configuration Register.

uint32_t nemadc_get_crc(void)

Read CRC Checksum Register.

void nemadc_set_bgcolor(uint32_t rgba)

Set NemaDC Background Color.

void nemadc_timing(int resx, int fpx, int blx, int bpx, int resy, int fpy, int bly, int bpy)

Set Display timing parameters.

int nemadc_stride_size(uint32_t format, int width)

Return stride size in bytes.

void nemadc_clkdiv(int div, int div2, int dma_prefetch, int phase)

Set the built-in Clock Dividers and DMA Line Prefetch. (See Configuration Register 0x4)

void nemadc_clkctrl(uint32_t ctrl)

Control the clock gaters.

void nemadc_set_mode(int mode)

Set operation mode.

uint32_t nemadc_get_status(void)

Get status from Status Register.

void nemadc_request_vsync_non_blocking(void)

Request a VSync Interrupt without blocking.

void nemadc_set_layer(int layer_no, nemadc_layer_t *layer)

Set the Layer Mode. This function can enable a layer and set attributes to it.

void nemadc_set_layer_addr(int layer_no, uintptr_t addr)

Set the physical address of a layer.

void nemadc_set_layer_gamma_lut(int layer, int index, int colour)

Set an entry in the lut8 Palette Gamma table for a layer.

int nemadc_get_layer_gamma_lut(int layer, int index)

Get an entry in the lut8 Palette Gamma table for a layer.

void nemadc_set_palette(uint32_t index, uint32_t colour)

Sets an entry in the lut8 Palatte Gamma table.

int nemadc_get_palette(uint32_t index)

Reads an entry from the lut8 Palatte Gamma table.

void nemadc_layer_disable(int layer_no)

Disable layer.

void nemadc_layer_enable(int layer_no)

Enable layer.

void nemadc_cursor_enable(int enable)

Enable or Disable fixed cursor.

void nemadc_cursor_xy(int x, int y)

Set the location of the cursor.

void nemadc_set_cursor_img(unsigned char *img)

Set programmable cursor image (32x32 pixels)

void nemadc_set_cursor_lut(uint32_t index, uint32_t color)

Set a color for the Cursor LUT.

unsigned char nemadc_check_config(uint32_t flag)

Check whether NemaDC supports a specific characteristic.

uint32_t nemadc_get_col_mode(void)

Read Color Mode Register.

int nemadc_get_layer_count(void)

Get the number of layers available.

uint32_t nemadc_get_ipversion(void)

Read IP Version register.

5.1.1.3 Detailed Description

Macro Definition Documentation

#define NEMADC_ABGR8888

ABGR8888

#define NEMADC_ARGB4444

ARGB4444

#define NEMADC_ARGB8888

ARGB8888

#define NEMADC_BF_DSTALPHA

Alpha Destination

#define NEMADC_BF_GLBALPHA

Alpha Global

#define NEMADC_BF_INVDSTALPHA

Inverted Destination

#define NEMADC_BF_INVGBLALPHA

Inverted Global

#define NEMADC_BF_INVSRCALPHA

Inverted Source

#define NEMADC_BF_INVSRCGBLALPHA

Inverted Source And Global

#define NEMADC_BF_ONE

```
White
#define NEMADC_BF_SRCALPHA
Alpha Source
#define NEMADC_BF_SRCGBLALPHA
Alpha Source And Alpha Global
#define NEMADC_BF_ZERO
Black
#define NEMADC_BGRA8888
BGRA8888
#define NEMADC_BLANK
BLANK
#define NEMADC_BL_ADD
Sa + Da
#define NEMADC_BL_CLEAR
0
#define NEMADC_BL_DST_ATOP
Sa * (1 - Da) + Da * Sa
#define NEMADC_BL_DST_IN
Da * Sa
#define NEMADC_BL_DST_OUT
Da * (1 - Sa)
#define NEMADC_BL_DST_OVER
Sa * (1 - Da) + Da
#define NEMADC_BL_SIMPLE
Sa * Sa + Da * (1 - Sa)
#define NEMADC_BL_SRC
Sa
#define NEMADC_BL_SRC_ATOP
Sa * Da + Da * (1 - Sa)
#define NEMADC_BL_SRC_IN
Sa * Da
#define NEMADC_BL_SRC_OUT
Sa * (1 - Da)
#define NEMADC_BL_SRC_OVER
Sa + Da * (1 - Sa)
#define NEMADC_BL_XOR
Sa * (1 - Da) + Da * (1 - Sa)
#define NEMADC_BT656
BT656
#define NEMADC_CFG_DBIB
DBI Type-B interface enabled
#define NEMADC_CFG_DITHERING
Dithering enabled
```

#define NEMADC_CFG_FIXED_CURSOR
Fixed Cursor enabled
#define NEMADC_CFG_FORMAT
Formatting enabled
#define NEMADC_CFG_HiQ_YUV
High Quality YUV converted enabled
#define NEMADC_CFG_L0_BLENDER
Layer 0 has blender
#define NEMADC_CFG_L0_ENABLED
Layer 0 enabled
#define NEMADC_CFG_L0_GAMMA
Layer 0 has gamma LUT
#define NEMADC_CFG_L0_SCALER
Layer 0 has scaler
#define NEMADC_CFG_L1_BLENDER
Layer 1 has blender
#define NEMADC_CFG_L1_ENABLED
Layer 1 enabled
#define NEMADC_CFG_L1_GAMMA
Layer 1 has gamma LUT
#define NEMADC_CFG_L1_SCALER
Layer 1 has scaler
#define NEMADC_CFG_L2_BLENDER
Layer 2 has blender
#define NEMADC_CFG_L2_ENABLED
Layer 2 enabled
#define NEMADC_CFG_L2_GAMMA
Layer 2 has gamma LUT
#define NEMADC_CFG_L2_SCALER
Layer 2 has scaler
#define NEMADC_CFG_L3_BLENDER
Layer 3 has blender
#define NEMADC_CFG_L3_ENABLED
Layer 3 enabled
#define NEMADC_CFG_L3_GAMMA
Layer 3 has gamma LUT
#define NEMADC_CFG_L3_SCALER
Layer 3 has scaler
#define NEMADC_CFG_PALETTE
Global Gamma enabled
#define NEMADC_CFG_PROGR_CURSOR
Programmable Cursor enabled
#define NEMADC_CFG_YUVOUT

RGB to YUV converted

#define NEMADC_CLKOUTDIV

CLKOUTDIV

#define NEMADC_CURSOR

CURSOR

#define NEMADC_DISABLE

DISABLE

#define NEMADC_DITHER

DITHER 18-bit

#define NEMADC_DITHER15

DITHER 15-bit

#define NEMADC_DITHER16

DITHER 16-bit

#define NEMADC_ENABLE

ENABLE

#define NEMADC_FORCE_A

Force Alpha

#define NEMADC_GAMMA

GAMMA

#define NEMADC_INTERLACE

INTERLACE

#define NEMADC_INVPIXCLK

INVPIXCLK

#define NEMADC_IPI

IPI

#define NEMADC_JDIMIP

JDIMIP

#define NEMADC_L8

L8

#define NEMADC_LAYER_AHBLOCK

Activate HLOCK signal on AHB DMAs

#define NEMADC_LAYER_DISABLE

Disable Layer

#define NEMADC_LAYER_ENABLE

Enable Layer

#define NEMADC_LAYER_GAMMALUT_EN

Enable Gamma Look Up Table

#define NEMADC_LAYER_REG_PROTECT

Enable Register Protection

#define NEMADC_LUT8

LUT8

#define NEMADC_LVDSPADS

LVDSPADS

#define NEMADC_LVDS_ISP68
LVDS_ISP68
#define NEMADC_LVDS_ISP8
LVDS_ISP8
#define NEMADC_LVDS_OFF
LVDS_OFF
#define NEMADC_MIPI_OFF
MIPI_OFF
#define NEMADC_MIP_IF
Parallel Memory in Pixel Interface
#define NEMADC_MODULATE_A
Modulate Alpha
#define NEMADC_NEG_DE
NEG_DE
#define NEMADC_NEG_H
NEG_H
#define NEMADC_NEG_V
NEG_V
#define NEMADC_ONE_FRAME
ONE_FRAME
#define NEMADC_OUTP_OFF
OUTP_OFF
#define NEMADC_PALETTE
PALETTE
#define NEMADC_P_RGB3
P_RGB3
#define NEMADC_P_RGB3_16B
P_RGB3
#define NEMADC_P_RGB3_16B1
P_RGB3
#define NEMADC_P_RGB3_16B2
P_RGB3
#define NEMADC_P_RGB3_18B
P_RGB3
#define NEMADC_P_RGB3_18B1
P_RGB3
#define NEMADC_RGB24
RGB24
#define NEMADC_RGB332
RGB332
#define NEMADC_RGB565
RGB565
#define NEMADC_RGBA4444

RGBA4444
#define NEMADC_RGBA5551
RGBA5551
#define NEMADC_RGBA8888
RGBA8888
#define NEMADC_SCALE_NN
Activate Bilinear Filter
#define NEMADC_SCANDOUBLE
SCANDOUBLE
#define NEMADC_SINGLEV
SINGLEV
#define NEMADC_S_12BIT
S_12BIT
#define NEMADC_S_RGB3
S_RGB3
#define NEMADC_S_RGBX4
S_RGBX4
#define NEMADC_TESTMODE
TESTMODE
#define NEMADC_TSC4
TSC4
#define NEMADC_TSC6
TSC6
#define NEMADC_TSC6A
TSC6A
#define NEMADC_T_16BIT
T_16BIT
#define NEMADC_UNDERFLOW_MECH
Underflow mechanism, when set to 1 it disables this mechanism
#define NEMADC_YUVOUT
YUVOUT

5.1.1.4 Function Documentation

unsigned char nemadc_check_config (uint32_t flag)

Check whether NemaDC supports a specific characteristic.

Table 5-2: Flag Parameter

Parameter	Description
flag	Flag to query

`void nemadc_clkctrl ( uint32_t ctrl )`

Control the clock gaters.

Table 5-3: Clock Control Parameter

Parameter	Description
ctrl	clock control

`void nemadc_clkdiv ( int div, int div2, int dma_prefetch, int phase )`

Set the built-in Clock Dividers and DMA Line Prefetch. (See Configuration Register 0x4).

Table 5-4: Built-in Clock Dividers and DMA Line Prefetch Parameters

Parameter	Description
div	Set Divider 1
div2	Set Divider 2
dma_prefetch	Set number of lines for the dma to prefetch
phase	Clock phase shift

`void nemadc_cursor_enable ( int enable )`

Enable or Disable fixed cursor.

Table 5-5: Fixed Cursor Parameter

Parameter	Description
enable	1 for enable or 0 for disable cursor

`void nemadc_cursor_xy ( int x, int y )`

Set the location of the cursor.

Table 5-6: Location of the Cursor Parameter

Parameter	Description
x	Cursor X coordinate
x	Cursor Y coordinate

`uint32_t nemadc_get_col_mode ( void )`

Read Color Mode Register.

Return
Color mode register.

`uint32_t nemadc_get_config ( void )`

Read Configuration Register.

Return
Configuration Register Value.

`uint32_t nemadc_get_crc ( void )`

Read CRC Checksum Register.

Return
CRC checksum value of last frame. For testing purposes.

`uint32_t nemadc_get_ipversion ( void )`

Read IP Version register.

Return
NemaDC Version. IP Version register was included after 22.03.01 release. If used with previous version of the IP, function returns 0x210000.

`int nemadc_get_layer_count ( void )`

Get the number of layers available.

Return
Number of layers.

`int nemadc_get_layer_gamma_lut ( int layer, int index )`

Get an entry in the lut8 Palette Gamma table for a layer.

Table 5-7: lut8 Palette Gamma Table for a Layer Parameter

Parameter	Description
layer	Layer number
index	Color Index

Return
Palette index.

```
int nemadc_get_palette (  uint32_t index )
```

Reads an entry from the lut8 Palatte Gamma table.

Table 5-8: lut8 Palette Gamma Table Parameter

Parameter	Description
index	Color Index

Return
Return Color for given palette index.

```
int nemadc_init (  void )
```

Initialize NemaDC library.

Return
-1 on error

```
void nemadc_layer_disable (  int layer_no )
```

Disable layer.

Table 5-9: Disable Layer Parameter

Parameter	Description
layer_no	Layer Number

```
void nemadc_layer_enable (  int layer_no )
```

Enable layer.

Table 5-10: Enable Layer Parameter

Parameter	Description
layer_no	Layer Number

```
void nemadc_set_cursor_lut (  uint32_t index,  uint32_t color )
```

Set a color for the Cursor LUT.

Table 5-11: Color for the Cursor LUT Parameter

Parameter	Description
index	Color index
color	32-bit RGBA value

`void nemadc_set_layer ( int layer_no, nemadc_layer_t * layer )`

Set the Layer Mode. This function can enable a layer and set attributes to it.

Table 5-12: Layer Mode Parameter

Parameter	Description
layer_no	The layer number
layer	Layer Attributes struct

`void nemadc_set_layer_gamma_lut (int layer, int index,int colour)`

Set an entry in the lut8 Palette Gamma table for a layer.

Table 5-13: Entry in the lut8 Palette Gamma Table for a Layer Parameter

Parameter	Description
layer	Layer number
index	Color Index
Color	32-bit RGBA color value or gamma index

`void nemadc_set_mode ( int mode )`

Set operation mode.

Table 5-14: Operation Mode Parameter

Parameter	Description
mode	Mode of operation (see Register 0)
uint32_t	color 32-bit RGBA color value or Gamma index

`void nemadc_set_palette ( uint32_t index, uint32_t colour )`

Sets an entry in the lut8 Palette Gamma table.

Table 5-15: Entry in lut8 Palette Gamma Parameter

Parameter	Description
uint32_t	index Color Index

```
int nemadc_stride_size (  uint32_t format,  int width )
```

Return stride size in bytes.

Table 5-16: Stride Size Parameter

Parameter	Description
format	Texture color format
width	Texture width

Return
Stride in bytes

```
void nemadc_timing (  int resx,  int fpx,  int blx,  int bpx,  int
resy,  int fpy, int bly,  int bpy )
```

Set Display timing parameters.

Table 5-17: Display Timing Parameter

Parameter	Description
resx	Resolution X
fpx	Front Porch X
blx	Blanking X
bpx	Back Porch X
resv	Resolution Y
fpy	Front Porch Y
bly	Blanking Y
bpy	Back Porch Y

5.1.2 nema_dc_dsi.h File

```
#include "nema_sys_defs.h"
```

5.1.2.1 Macros

```
#define NEMADC_IPI_VIDEO_MODE
#define NEMADC_IPI_DATA_MODE
#define NEMADC_IPI_COLOR_RGB565
#define NEMADC_IPI_COLOR_RGB666
#define NEMADC_IPI_COLOR_RGB666_LOS
#define NEMADC_IPI_COLOR_RGB888
#define NEMADC_IPI_COLOR_YCbYCr422
#define NEMADC_IPI_COLOR_YCbYCr422_LOS
#define NEMADC_IPI_COLOR_YCbYCr420
#define NEMADC_IPI_COLOR_COMPRESSED
#define NEMADC_IPI_COLORM
#define NEMADC_IPI_SHUTDOWN
#define NEMADC_IPI_AUTO_TEAR_EFF
#define NEMADC_IPI_HIBERNATE
#define NEMADC_IPI_ENABLE
#define NemaDC_dt_vsync_start
#define NemaDC_dt_vsync_end
#define NemaDC_dt_hsync_start
#define NemaDC_dt_hsync_end
#define NemaDC_dt_cmpr_mode
#define NemaDC_dt_end_of_trans
#define NemaDC_dt_pic_param
#define NemaDC_dt_cmpr_pix_stream
#define NemaDC_dt_color_mode_off
#define NemaDC_dt_color_mode_on
#define NemaDC_dt_shut_down_peripheral
#define NemaDC_dt_turn_on_peripheral
#define NemaDC_dt_generic_short_write_param_no
#define NemaDC_dt_generic_short_write_param_n1
#define NemaDC_dt_generic_short_write_param_n2
#define NemaDC_dt_generic_read_param_no
#define NemaDC_dt_generic_read_param_n1
#define NemaDC_dt_execute_queue
#define NemaDC_dt_generic_read_param_n2
#define NemaDC_dt_DCS_short_write_param_no
#define NemaDC_dt_DCS_short_write_param_n1
#define NemaDC_dt_DCS_read_param_no
```

```
#define NemaDC_dt_set_max_return_packet_size
#define NemaDC_dt_blanking_packet
#define NemaDC_dt_generic_long_write
#define NemaDC_dt_DCS_long_write
#define NemaDC_dt_packed_pixel_stream_rgb565
#define NemaDC_dt_packed_pixel_stream_rgb666
#define NemaDC_dt_loosely_packed_pixel_stream_rgb666
#define NemaDC_dt_loosely_packed_pixel_stream_rgb888
#define NemaDC_dcs_datacmd
#define NemaDC_ge_data
#define NemaDC_ge_cmd
#define NemaDC_ge_datacmd
#define ENABLE_DSI
#define ENABLE_LOWPOWER
#define ENABLE_HIGHSPEED
#define GENERIC_CMD_ENABLE
#define MIPI_CMD_ENABLE
#define DSI_VC_0
#define DSI_VC_1
#define DSI_VC_2
#define DSI_VC_3
```

5.1.2.2 Functions

void nemadc_dsi_start_frame_transfer(void)
*Send **scanline** command and start memory write.*

void

nemadc_dsi_start_frame_transfer_generic(void)
Send scanline command and start memory write (generic)

void nemadc_dsi_ct(uint32_t data_type, uint32_t cmd_type, uint32_t type) DC
DBI interface to DSI.

5.1.2.3 Function Documentation

void nemadc_dsi_ct (uint32_t data_type, uint32_t cmd_type, uint32_t type)

DC DBI interface to DSI.

Table 5-18: DC DBI Interface to DSI Parameter

Parameter	Description
data_type	Data (pixel) type
cmd_type	Command type
type	DSI command type

5.1.3 nema_dc_hal.h File

```
#include "nema_sys_defs.h"
```

5.1.3.1 Functions

```
int32_t nemadc_sys_init(void)
Initialize system. Implementor defined. Called in

nemadc_init() void nemadc_wait_vsync(void)
Wait for VSYNC.

uint32_t nemadc_reg_read(uint32_t reg)
Read Hardware register.

void nemadc_reg_write(uint32_t reg, uint32_t value)
Write Hardware Register.
```

5.1.3.2 Function Documentation

```
uint32_t nemadc_reg_read (  uint32_t reg )
```

Read Hardware register.

Table 5-19: Read Hardware Register Parameter

Parameter	Description
reg	Register to read

Return
Value read from the register. See also **nema_reg_write**.

```
void nemadc_reg_write (  uint32_t reg,  uint32_t value )
```

Write Hardware Register.

Table 5-20: Write Hardware Register Parameter

Parameter	Description
reg	Register to write
value	Value to be written

See also **nema_reg_read()**.

int32_t nemadc_sys_init (void)
Initialize system. Implementor defined. Called in **nemadc_init()**.

Return
0 if no errors occurred.

See also **nema_init()**.

5.1.4 nema_dc_mipi.h File

```
#include "nema_sys_defs.h"
```

5.1.4.1 Macros

```
#define MIPI_enter_idle_mode
#define MIPI_enter_invert_mode
#define MIPI_enter_normal_mode
#define MIPI_enter_partial_mode
#define MIPI_enter_sleep_mode
#define MIPI_exit_idle_mode
#define MIPI_exit_invert_mode
#define MIPI_exit_sleep_mode
#define MIPI_get_3D_control
#define MIPI_get_address_mode
#define MIPI_get_blue_channel
#define MIPI_get_diagnostic_result
#define MIPI_get_display_mode
#define MIPI_get_green_channel
#define MIPI_get_pixel_format
#define MIPI_get_power_mode
#define MIPI_get_red_channel
#define MIPI_get_scanline
#define MIPI_get_signal_mode
#define MIPI_nop
#define MIPI_read_DDB_continue
#define MIPI_read_DDB_start
#define MIPI_read_memory_continue
#define MIPI_read_memory_start
#define MIPI_set_3D_control
#define MIPI_set_address_mode
#define MIPI_set_column_address
#define MIPI_set_display_off
#define MIPI_set_display_on
```

```
#define MIPI_set_gamma_curve
#define MIPI_set_page_address
#define MIPI_set_partial_columns
#define MIPI_set_partial_rows
#define MIPI_set_pixel_format
#define MIPI_set_scroll_area
#define MIPI_set_scroll_start
#define MIPI_set_tear_off
#define MIPI_set_tear_on
#define MIPI_set_tear_scanline
#define MIPI_set_vsync_timing
#define MIPI_soft_reset
#define MIPI_write_LUT
#define MIPI_write_memory_continue
#define MIPI_write_memory_start
#define MIPI_snapshot
#define MIPI_DBIB_STORE_BASE_ADDR
#define MIPI_DBIB_CMD
#define MIPI_CMD16
#define MIPI_CMD24
#define MIPI_MASK_QSPI
#define MIPICFG_DBI_EN
#define MIPICFG_FRC_CSX_0
#define MIPICFG_FRC_CSX_1
#define MIPICFG_SPI_CSX_V
#define MIPICFG_DIS_TE
#define MIPICFG_SPIDC_DQSPI
#define MIPICFG_RSTN_DBI_SPI
#define MIPICFG_RESX
#define MIPICFG_DMA
#define MIPICFG_SPI3
#define MIPICFG_SPI4
#define MIPICFG_GPI
#define MIPICFG_EN_STALL
#define MIPICFG_SPI_CPHA
#define MIPICFG_SPI_CPOL
#define MIPICFG_SPI_JDI
#define MIPICFG_EN_DVALID
#define MIPICFG_SPI_HOLD
#define MIPICFG_INV_ADDR
#define MIPICFG_SCAN_ADDR
#define MIPICFG_PIXCLK_OUT_EN
#define MIPICFG_EXT_CTRL
```

```
#define MIPICFG_BLANKING_EN
#define MIPICFG_DSPI_SPIX
#define MIPICFG_QSPI
#define MIPICFG_QSPI_DDR
#define MIPICFG_DSPI
#define MIPICFG_NULL
#define MIPI_DCS_RGB111
#define MIPI_DCS_RGB332
#define MIPI_DCS_RGB444
#define MIPI_DCS_BW
#define MIPI_DCS_RGB565
#define MIPI_DCS_RGB666
#define MIPI_DCS_RGB888
#define MIPICFG_PF_SPI
#define MIPICFG_PF_DSPI
#define MIPICFG_PF_QSPI
#define MIPICFG_PF_DBI8
#define MIPICFG_PF_DBI9
#define MIPICFG_PF_DBI16
#define MIPICFG_PF_GPI
#define MIPICFG_PF_OPT0
#define MIPICFG_PF_OPT1
#define MIPICFG_PF_OPT2
#define MIPICFG_PF_OPT3
#define MIPICFG_PF_OPT4
#define MIPICFG_1RGB111_OPT0
#define MIPICFG_1RGB111_OPT1
#define MIPICFG_1RGB111_OPT2
#define MIPICFG_1RGB111_OPT3
#define MIPICFG_1RGB111_OPT4
#define MIPICFG_1RGB332_OPT0
#define MIPICFG_1RGB444_OPT0
#define MIPICFG_1RGB565_OPT0
#define MIPICFG_1RGB565_OPT1
#define MIPICFG_1RGB666_OPT0
#define MIPICFG_1RGB888_OPT0
#define MIPICFG_2RGB444_OPT0
#define MIPICFG_2RGB444_OPT1
#define MIPICFG_2RGB565_OPT0
#define MIPICFG_2RGB565_OPT1
#define MIPICFG_2RGB666_OPT0
#define MIPICFG_2RGB666_OPT1
#define MIPICFG_2RGB888_OPT0
```

```
#define MIPICFG_2RGB888_OPT1
#define MIPICFG_BW
#define MIPICFG_4RGB111_OPT0
#define MIPICFG_4RGB332_OPT0
#define MIPICFG_4RGB444_OPT0
#define MIPICFG_4RGB565_OPT0
#define MIPICFG_4RGB565_OPT1
#define MIPICFG_4RGB666_OPT0
#define MIPICFG_4RGB888_OPT0
#define MIPICFG_8RGB332_OPT0
#define MIPICFG_8RGB444_OPT0
#define MIPICFG_8RGB565_OPT0
#define MIPICFG_8RGB565_OPT1
#define MIPICFG_8RGB666_OPT0
#define MIPICFG_8RGB888_OPT0
#define MIPICFG_16RGB332_OPT0
#define MIPICFG_16RGB444_OPT0
#define MIPICFG_16RGB565_OPT0
#define MIPICFG_16RGB565_OPT1
#define MIPICFG_16RGB666_OPT0
#define MIPICFG_16RGB666_OPT1
#define MIPICFG_16RGB888_OPT0
#define MIPICFG_16RGB888_OPT1
#define MIPICFG_9RGB666_OPT0
#define MIPICFG_32RGB332_OPT0
#define MIPICFG_32RGB444_OPT0
#define MIPICFG_32RGB565_OPT0
#define MIPICFG_32RGB666_OPT0
#define MIPICFG_32RGB666_OPT1
#define MIPICFG_32RGB888_OPT0
#define nemadc_MIPI_set_mode
```

5.1.4.2 Functions

void nemadc_MIPI_out(int cmd)
Send command or data to MIPI Interface.

void nemadc_MIPI_CFG_out(int cfg)
Configure NemaDC's serial interace.

int nemadc_MIPI_in(void)
Read data from MIPI interface.

unsigned nemadc_MIPI_read(int cmd, int n_params)

Read MIPI DBI Type-B parameters.

void nemadc_MIPI_cmd(int cmd)
Send DCS command to display over the physical interface.

void nemadc_MIPI_cmd_params(int cmd, int n_params,...)
Similar to nemadc_MIPI_cmd, with command parameters.

int nemadc_MIPI_updateregion(int start_x, int start_y, int end_x, int end_y)
Does Partial Update in MIPI.

void nemadc_MIPI_enable(void)
Convenience function. Sends exit_sleep and display_on commands.

void nemadc_MIPI_disable(void)
Convenience function. Sends display_off and enter_sleep_mode commands.

void nemadc_MIPI_set_pixel_format(int pixel_format)
Set the display pixel format. Sends set_pixel_format command to the display.

void nemadc_MIPI_set_position(int minx, int miny, int maxx, int maxy)
Set the frame position. Sends set_column_address and set_page_address commands.

void nemadc_MIPI_set_partial_mode(int minx, int miny, int maxx, int maxy)
Set the display partial area and enter Partial Display Mode.

void nemadc_MIPI_start_frame_transfer(void)
Convenience function. Send a write_memory_start command in order to start transfering the frame to the display.

5.1.4.3 Macro Definition Documentation

#define MIPICFG_BLANKING_EN
Enables horizontal blanking

#define MIPICFG_DBI_EN
Enables MIPI DBI/SPI interface

#define MIPICFG_DIS_TE
Disables Input Tearing Signal

#define MIPICFG_DMA
(unused) Enables pixel data from DMA

#define MIPICFG_DSPI
Enables DSPI

#define MIPICFG_DSPI_SPIX
Enables DSPI sub-pixel transaction

#define MIPICFG_EN_DVALID
Enables read using external data valid signal

#define MIPICFG_EN_STALL
Enables back-pressure from dbi_stall_i signal

#define MIPICFG_EXT_CTRL
Enables external control signals

#define MIPICFG_FRC_CSX_0
Enables CSX force value

#define MIPICFG_FRC_CSX_1
Force CSX to 1

#define MIPICFG_GPI
Enables Generic Packet Interface

#define MIPICFG_INV_ADDR
Inverts scanline address

#define MIPICFG_PIXCLK_OUT_EN
Redirects pixel generation clock to the output

#define MIPICFG_QSPI
Enables QSPI

#define MIPICFG_QSPI_DDR
Enables QSPI DDR

#define MIPICFG_RESX
Controls MIPI DBI Type-B RESX output signal

#define MIPICFG_RSTN_DBI_SPI
DBI/SPI interfaces clear

#define MIPICFG_SCAN_ADDR
Scan address used as header of each line

#define MIPICFG_SPI3
Enables SPI 3-wire interface

#define MIPICFG_SPI4
Enables SPI 4-wire interface

#define MIPICFG_SPIDC_DQSPI
Enables the usage of SPI_DC wire as SPI_SD1

#define MIPICFG_SPI_CPHA
Sets SPI Clock Phase

#define MIPICFG_SPI_CPOL
Sets SPI Clock Polarity
#define MIPICFG_SPI_CSX_V
CSX active high/low

#define MIPICFG_SPI_HOLD
Binds scanline address with pixel data

#define MIPICFG_SPI_JDI
reserved

5.1.4.4 Function Documentation

void nemadc_MIPI_CFG_out (int cfg)

Configure NemaDC's serial interface.

Table 5-21: Configure NemaDC's Serial Interface Parameter

Parameter	Description
cfg	configuration mode

void nemadc_MIPI_cmd (int cmd)

Send DCS command to display over the physical interface.

Table 5-22: Send DCS to Display Over the Physical Interface Parameter

Parameter	Description
cmd	MIPI DCS command
n_params	Number of cmd parameters

void nemadc_MIPI_out (int cmd)

Send command or data to MIPI Interface.

Table 5-23: Send Command or Data to MIPI Interface Parameter

Parameter	Description
cmd	command or data to be sent

`unsigned nemadc_MIPI_read ( int cmd, int n_params )`

Read MIPI DBI Type-B parameters.

Table 5-24: Read MIPI DBI Type-B Parameter

Parameter	Description
cmd	MIPI DCS command
n_params	Number of parameters to read (max: 3)

`void nemadc_MIPI_set_partial_mode (int minx, int miny, int maxx, int maxy)`

Set the display partial area and enter Partial Display Mode.

Table 5-25: Display Partial Area and Enter Partial Display Mode Parameter

Parameter	Description
minx	partial areas' minimum x
miny	partial areas' minimum y
maxx	partial areas' maximum x
maxy	partial areas' maximum y

`void nemadc_MIPI_set_pixel_format ( int pixel_format )`

Set the display pixel format. Sends **set_pixel_format** command to the display.

Table 5-26: Set the Display Pixel Format Parameter

Parameter	Description
pixel_format	pixel format

`void nemadc_MIPI_set_position (int minx, int miny, int maxx, int maxy)`

Set the frame position. Sends **set_column_address** and **set_page_address** commands.

Table 5-27: Set the Frame Position Parameter

Parameter	Description
minx	frames' minimum x
miny	frames' minimum y
maxx	frames' maximum x
maxy	frames' maximum y

```
int nemadc_MIPI_updateregion (  int start_x,  int start_y,  int
end_x,  int end_y )
```

Does Partial Update in MIPI.

Table 5-28: Partial Update in MIPI Parameter

Parameter	Description
start_x	start x coordinate
start_y	start y coordinate
end_x	end ex coordinate
end_y	end y coordinate

5.2 Directories

Here is a list of all directories with brief descriptions:

5.2.1 File List

NemaDC
nema_dc_dsi.h
nema_dc_mipi.h

5.2.2 File List

NemaDC
nema_dc.h
nema_dc_hal.h

5.3 Data Structures

Here is a list of all data structures with brief descriptions:

5.3.1 nemadc_display_t Data Structure

```
#include <nema_dc.h>
```

Table 5-29: nemadc_display_t Data Structure

Data	Fields
uint32_t resx	Resolution X
uint32_t resy	Resolution Y
uint32_t fpx	Front Porch X
uint32_t fpy	Front Porch Y
uint32_t bpx	Back Porch X
uint32_t bpy	Back Porch Y
uint32_t blx	Blanking X
uint32_t bly	Blanking Y

5.3.2 nemadc_layer_t Data Structure

```
#include <nema_dc.h>
```

Table 5-30: nemadc_layer_t Data Structure

Data	Fields
void * baseaddr_virt	Virtual Address
uintptr_t baseaddr_phys	Physical Address
uint32_t resx	Resolution X
uint32_t resy	Resolution Y
int32_t stride	Stride
int32_t startx	Start X
int32_t starty	Start Y
uint32_t sizex	Size X
uint32_t sizey	Size Y
uint8_t alpha	Alpha uint8_t
blendmode	Blending
Mode uint8_t	buscfg ??
uint32_t format	Format
uint32_t mode	Mode
uint32_t u_base	U Base

Table 5-30: nemadc_layer_t Data Structure *(Continued)*

Data	Fields
uint32_t v_base	Y Base
uint32_t u_stride	U Strideuin
t32_t v_stride	V Strideuin
t8_t flipx_en	Flipx_en
uint8_t flipy_en	Flipy_en
uint32_t extra_bits	Extra configuration bits of Layer

6500 River Place Boulevard, Building 7, Suite 200, Austin, TX 78730
www.ambiq.com
sales@ambiq.com
+1 512. 879.2850

A-SOCAPG-UMGA01EN v1.0
September 2025