SMART ARM-based Microcontrollers

# AT07910: SAM4L Liquid Crystal Display (LCDCA) Driver

APPLICATION NOTE

## Introduction

This driver for Atmel® | SMART ARM®-based microcontrollers provides an interface for the configuration and management of the device's Liquid Crystal Display Controller functionality.

The LCD controller is intended for monochrome passive Liquid Crystal Displays (LCDs) with up to four common terminals and up to 40 segment terminals.

Devices from the following series can use this module:

- Atmel | SMART SAM4L

The outline of this documentation is as follows:

# Table of Contents

# 1. Software License

# 2. Prerequisites

There are no prerequisites for this module.

# 3. Module Overview

## 3.1. Display Physiology

An LCD is made up of several segments (or complete symbols), which can be visible or invisible. Any single segment has two electrodes, with liquid crystal between them. These electrodes are connected to the common terminal (COM) and the segment terminal (SEG) on the display. When a voltage above a threshold voltage is applied across the liquid crystal, the segment becomes visible. The voltage must alternate, to avoid an electrophoresis effect in the liquid crystal, which degrades the display.

## 3.2. Supported Display Types

The LCD Controller is intended for use with monochrome passive Liquid Crystal Displays (LCDs) with up to four common terminals and up to 40 segment terminals.

## 3.3. Functional Description

The LCDCA display memory stores the values of all segments to display and should be filled before the next frame starts.

The start of a new frame triggers the update of the shadow display memory. The content of the display memory is copied into the shadow display memory. A display memory refresh is possible without affecting data that is sent to the panel.

**Note:** The LCDCA display memory is not initialized at power-up.

When any bit in the display memory is written to a one, the corresponding LCD segment will be energized and opaque. If the same bit has zero written to it, then the corresponding LCD segment is de-energized and thus transparent.

# 4. Special Considerations

## 4.1. I/O Lines

The LCDCA pins (SEGx and COMy) are multiplexed with other peripherals. The user application must first configure the I/O Controller, to give control of the requisite pins to the LCDCA.

## 4.2. Power Management

This module can control the LCD display while CLK_LCDCA is disabled, but stops functioning when CLK_LCD (32kHz) is disabled.

The power consumption of LCDCA itself can be minimized by:

- Using the lowest acceptable frame rate (refer to the LCD glass technical characteristics)
- Using the low-power waveform (default mode)
- Using automated modes of operation
- Configuring the lowest possible contrast value

## 4.3. Clocks

The clock for this module (CLK_LCDCA) is generated by the Power Manager. It can be enabled or disabled, either manually (via the Power Manager) or automatically when the system enters a sleep mode that disables the clocks to the peripheral bus modules.

The 32kHz clock (CLK_LCD) must be enabled before use.

## 4.4. Interrupts

The LCDCA interrupt request line is connected to the Nested Vectored Interrupt Controller (NVIC). Use of the LCDCA interrupt requires the interrupt controller to be configured first.

## 4.5. Wake Up

The LCD Controller's wake-up signal is connected to the Power Manager Controller (PMC). Use of the wake-up mechanism requires the PMC to enable the corresponding asynchronous wake-up source first. Also, the LCDCA interrupt must be enabled first.

## 4.6. Debug Operation

When an external debugger forces the CPU into debug mode, the LCDCA continues normal operation.

# 5. Extra Information

For extra information, see Extra Information for Liquid Crystal Display Controller Driver. This includes:

- Acronyms
- Dependencies
- Errata
- Module History

# 6. Examples

For a list of examples related to this driver, see Examples for Liquid Crystal Display Controller.

# 7. API Overview

## 7.1. Variable and Type Definitions

### 7.1.1. Type lcdca_automated_char_config_t

```
typedef struct lcdca_automated_char_config lcdca_automated_char_config_t
```

Automated character display configuration structure.

### 7.1.2. Type lcdca_blink_config_t

```
typedef struct lcdca_blink_config lcdca_blink_config_t
```

Blink configuration structure

### 7.1.3. Type lcdca_callback_t

```
typedef void(* lcdca_callback_t )(void)
```

LCDCA interrupt callback type

### 7.1.4. Type lcdca_circular_shift_config_t

```
typedef struct lcdca_circular_shift_config lcdca_circular_shift_config_t
```

Circular shift configuration structure

### 7.1.5. Type lcdca_config_t

```
typedef struct lcdca_config lcdca_config_t
```

LCDCA controller configuration structure

## 7.2. Structure Definitions

### 7.2.1. Struct lcdca_automated_char_config

Automated character display configuration structure.

**Table 7-1 Members**

| Type | Name | Description |
|---|---|---|
| uint8_t | automated_mode | Automated display mode selection |
| uint8_t | automated_timer | Timer selected for automated mode |
| uint8_t | dign | Number of digits used in the display |
| uint8_t | dir_reverse | Digit display direction |
| uint8_t | lcd_tdg | Type of digit selected |
| uint8_t | steps | The number of steps to use in scrolling mode. STEPS = string length - DIGN + 1 |
| uint8_t | stseg | Start segment |

### 7.2.2. Struct lcdca_blink_config

Blink configuration structure

**Table 7-2 Members**

| Type | Name | Description |
|---|---|---|
| uint8_t | lcd_blink_mode | Blink Mode |
| uint8_t | lcd_blink_timer | LCDCA blink timer |

### 7.2.3. Struct lcdca_circular_shift_config

Circular shift configuration structure

**Table 7-3 Members**

| Type | Name | Description |
|---|---|---|
| uint8_t | data | Circular shift register value |
| uint8_t | lcd_csr_dir | Shift direction |
| uint8_t | lcd_csr_timer | LCDCA shift register timer |
| uint8_t | size | Size of the circular shift register |

### 7.2.4. Struct lcdca_config

LCDCA controller configuration structure

**Table 7-4 Members**

| Type | Name | Description |
|---|---|---|
| int8_t | contrast | Contrast setting |
| uint8_t | duty_type | Duty type selection |

| Type | Name | Description |
|---|---|---|
| uint8_t | fc0 | Frame Counter 0 |
| uint8_t | fc1 | Frame Counter 1 |
| uint8_t | fc2 | Frame Counter 2 |
| uint8_t | lcd_clkdiv | Prescale clock source divider |
| bool | lcd_pres | Clock source prescale bypass enable/disable |
| bool | lp_wave | Wave mode (false: low power waveform, true: standard waveform) |
| uint8_t | port_mask | Number of SEGx pin connections used |
| bool | x_bias | External bias (false: internal gen, true: external gen) |

## 7.3. Macro Definitions

### 7.3.1. LCDCA Address Limits

#### 7.3.1.1. Macro LCDCA_MAX_NR_OF_COM

```
#define LCDCA_MAX_NR_OF_COM
```

Maximum number of common lines

#### 7.3.1.2. Macro LCDCA_MAX_NBR_OF_SEG

```
#define LCDCA_MAX_NBR_OF_SEG
```

Maximum number of segment lines

### 7.3.2. LCDCA Display Digit

#### 7.3.2.1. Macro LCDCA_TDG_7SEG3COM

```
#define LCDCA_TDG_7SEG3COM
```

7-segment display with three common terminals

#### 7.3.2.2. Macro LCDCA_TDG_7SEG4COM

```
#define LCDCA_TDG_7SEG4COM
```

7-segment display with four common terminals

#### 7.3.2.3. Macro LCDCA_TDG_14SEG4COM

```
#define LCDCA_TDG_14SEG4COM
```

14-segment display with four common terminals

#### 7.3.2.4. Macro LCDCA_TDG_16SEG3COM

```
#define LCDCA_TDG_16SEG3COM
```

16-segment display with three common terminals

### 7.3.3. LCDCA Timer Resource

#### 7.3.3.1. Macro LCDCA_TIMER_FC0

```
#define LCDCA_TIMER_FC0
```

Timer FC0 resource

#### 7.3.3.2. Macro LCDCA_TIMER_FC1

```
#define LCDCA_TIMER_FC1
```

Timer FC1 resource

#### 7.3.3.3. Macro LCDCA_TIMER_FC2

```
#define LCDCA_TIMER_FC2
```

Timer FC2 resource

### 7.3.4. LCDCA CMCFG Digit Reverse Mode

#### 7.3.4.1. Macro LCDCA_CMCFG_DREV_LEFT

```
#define LCDCA_CMCFG_DREV_LEFT
```

Decrement the segment index

#### 7.3.4.2. Macro LCDCA_CMCFG_DREV_RIGHT

```
#define LCDCA_CMCFG_DREV_RIGHT
```

Increment the segment index

### 7.3.5. LCDCA Duty Selection.

#### 7.3.5.1. Macro LCDCA_DUTY_1_4

```
#define LCDCA_DUTY_1_4
```

Duty=1/4, Bias=1/3, COM0:3

#### 7.3.5.2. Macro LCDCA_DUTY_STATIC

```
#define LCDCA_DUTY_STATIC
```

Duty=Static, Bias=Static, COM0

#### 7.3.5.3. Macro LCDCA_DUTY_1_2

```
#define LCDCA_DUTY_1_2
```

Duty=1/2, Bias=1/3, COM0:1

#### 7.3.5.4. Macro LCDCA_DUTY_1_3

```
#define LCDCA_DUTY_1_3
```

Duty=1/3, Bias=1/3, COM0:2

### 7.3.6. LCDCA Blink Mode

#### 7.3.6.1. Macro LCDCA_BLINK_FULL

```
#define LCDCA_BLINK_FULL
```

All LCD segments will blink

#### 7.3.6.2. Macro LCDCA_BLINK_SELECTED

```
#define LCDCA_BLINK_SELECTED
```

Only the selected segment will blink

### 7.3.7. LCDCA Shift Register Direction

#### 7.3.7.1. Macro LCDCA_CSR_RIGHT

```
#define LCDCA_CSR_RIGHT
```

Right direction

#### 7.3.7.2. Macro LCDCA_CSR_LEFT

```
#define LCDCA_CSR_LEFT
```

Left direction

### 7.3.8. LCDCA Automated Mode.

#### 7.3.8.1. Macro LCDCA_AUTOMATED_MODE_SEQUENTIAL

```
#define LCDCA_AUTOMATED_MODE_SEQUENTIAL
```

Sequential character string display mode

#### 7.3.8.2. Macro LCDCA_AUTOMATED_MODE_SCROLLING

```
#define LCDCA_AUTOMATED_MODE_SCROLLING
```

Scrolling character string display mode

### 7.3.9. LCDCA Automated Direction.

#### 7.3.9.1. Macro LCDCA_AUTOMATED_DIR_REVERSE

```
#define LCDCA_AUTOMATED_DIR_REVERSE
```

Digit direction is reversed.

#### 7.3.9.2. Macro LCDCA_AUTOMATED_DIR_NOT_REVERSE

```
#define LCDCA_AUTOMATED_DIR_NOT_REVERSE
```

Digit direction is not reversed.

### 7.3.10. Macro LCDCA_AUTOMATED_CHAR_DMA_CH

```
#define LCDCA_AUTOMATED_CHAR_DMA_CH
```

LCDCA automated character DMA Channel (sequential or scrolling)

## 7.4. Function Definitions

### 7.4.1. Function lcdca_automated_char_reload()

LCDCA automated display reload.

```
void lcdca_automated_char_reload(
        const uint8_t * data,
        size_t width)
```

**Table 7-5 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | data | Data string buffer pointer |
| **[in]** | width | Data string length |

### 7.4.2. Function lcdca_automated_char_set_config()

Configure the LCDCA automated display.

```
void lcdca_automated_char_set_config(
        struct lcdca_automated_char_config * ac_cfg)
```

**Note:** The automated display is disabled.

**Table 7-6 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | ac_cfg | Pointer to an automated display configuration structure |

### 7.4.3. Function lcdca_automated_char_start()

Start the LCDCA automated display.

```
void lcdca_automated_char_start(
        const uint8_t * data,
        size_t width)
```

**Note:** This function also initializes and enables the PDCA channel associated with the LCDCA module.

**Table 7-7 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | data | Data string buffer pointer |
| **[in]** | width | Data string length |

### 7.4.4. Function lcdca_automated_char_stop()

Stop the LCDCA automated display mode.

```
void lcdca_automated_char_stop( void )
```

**Note:** This function also disables the PDCA channel associated with the LCDCA module.

### 7.4.5. Function lcdca_blink_disable()

Disable the LCDCA blink mode.

```
void lcdca_blink_disable( void )
```

### 7.4.6. Function lcdca_blink_enable()

Enable the LCDCA blink mode.

```
void lcdca_blink_enable( void )
```

### 7.4.7. Function lcdca_blink_set_config()

Set the LCDCA blink configuration.

```
void lcdca_blink_set_config(
        struct lcdca_blink_config * blink_cfg)
```

**Table 7-8 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | blink_cfg | Pointer to the LCD blink configuration structure |

### 7.4.8. Function lcdca_circular_shift_disable()

Disable the LCDCA circular shift mode.

```
void lcdca_circular_shift_disable( void )
```

### 7.4.9. Function lcdca_circular_shift_enable()

Enable the LCDCA circular shift mode.

```
void lcdca_circular_shift_enable( void )
```

### 7.4.10. Function lcdca_circular_shift_set_config()

Set the LCDCA circular shift configuration.

```
void lcdca_circular_shift_set_config(
        struct lcdca_circular_shift_config * cs_cfg)
```

**Note:** The circular shift register is disabled.

**Table 7-9 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | cs_cfg | Pointer to a circular shift configuration structure |

### 7.4.11. Function lcdca_clear_blink_all_pixel()

Stop all LCDCA pixels/segments from blinking.

```
void lcdca_clear_blink_all_pixel( void )
```

### 7.4.12. Function lcdca_clear_blink_pixel()

Stop a specified LCDCA pixel/segment from blinking.

```
void lcdca_clear_blink_pixel(
        uint8_t pix_com,
        uint8_t pix_seg)
```

**Table 7-10 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | pix_com | Pixel/segment COMx coordinate |
| **[in]** | pix_seg | Pixel/segment SEGy coordinate (range 0 to 1 inclusive) |

### 7.4.13. Function lcdca_clear_display_memory()

Clear all the LCDCA display memory.

```
void lcdca_clear_display_memory( void )
```

### 7.4.14. Function lcdca_clear_pixel()

Disable the specified pixel/segment in the LCDCA display memory.

```
void lcdca_clear_pixel(
        uint8_t pix_com,
        uint8_t pix_seg)
```

**Table 7-11 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | pix_com | Pixel/segment COMx coordinate |
| **[in]** | pix_seg | Pixel/segment SEGy coordinate |

### 7.4.15. Function lcdca_clear_status()

Clear the LCDCA beginning of frame interrupt status.

```
void lcdca_clear_status( void )
```

### 7.4.16. Function lcdca_clk_init()

LCDCA clock initialization.

```
void lcdca_clk_init( void )
```

### 7.4.17. Function lcdca_disable()

Disable the LCDCA module.

```
void lcdca_disable( void )
```

### 7.4.18. Function lcdca_disable_interrupt()

Disable the LCDCA beginning of frame interrupt.

```
void lcdca_disable_interrupt( void )
```

### 7.4.19. Function lcdca_disable_timer()

Disable the specified LCDCA timer.

```
void lcdca_disable_timer(
        uint8_t lcd_timer)
```

**Table 7-12 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | lcd_timer | Timer number to be disabled |

### 7.4.20. Function lcdca_disable_wakeup()

Disable the LCDCA wake-up.

```
void lcdca_disable_wakeup( void )
```

### 7.4.21. Function lcdca_enable()

Enable the LCDCA module.

```
void lcdca_enable( void )
```

### 7.4.22. Function lcdca_enable_interrupt()

Enable the LCDCA beginning of frame interrupt.

```
void lcdca_enable_interrupt( void )
```

### 7.4.23. Function lcdca_enable_timer()

Enable the specified LCDCA timer, and wait until it is running.

```
void lcdca_enable_timer(
        uint8_t lcd_timer)
```

**Note:** The function lcdca_enable() must be called prior to this function.

**Table 7-13 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | lcd_timer | Timer number to be enabled |

### 7.4.24. Function lcdca_enable_wakeup()

Enable the LCDCA wake-up.

```
void lcdca_enable_wakeup( void )
```

### 7.4.25. Function lcdca_get_pixel()

Get the specified pixel/segment state from the LCDCA display memory.

```
bool lcdca_get_pixel(
        uint8_t pix_com,
        uint8_t pix_seg)
```

**Table 7-14 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | pix_com | Pixel/segment COMx coordinate |
| **[in]** | pix_seg | Pixel/segment SEGy coordinate |

**Returns**
The pixel/segment value from the LCDCA display memory.

### 7.4.26. Function lcdca_get_status()

Get the LCDCA status register.

```
uint32_t lcdca_get_status( void )
```

**Returns**
The LCDCA status register.

### 7.4.27. Function lcdca_lock_shadow_dislay()

Lock the LCDCA shadow display memory.

```
void lcdca_lock_shadow_dislay( void )
```

### 7.4.28. Function lcdca_set_blink_pixel()

Start an LCDCA pixel/segment blinking.

```
void lcdca_set_blink_pixel(
        uint8_t pix_com,
        uint8_t pix_seg)
```

**Table 7-15 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | pix_com | Pixel/segment COMx coordinate |
| **[in]** | pix_seg | Pixel/segment SEGy coordinate (range 0 to 1 inclusive) |

### 7.4.29. Function lcdca_set_callback()

Set the callback for the LCDCA 'beginning of frame' interrupt.

```
void lcdca_set_callback(
        lcdca_callback_t callback,
        uint8_t irq_line,
        uint8_t irq_level)
```

**Table 7-16 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | callback | Pointer to an interrupt callback function |
| **[in]** | irq_line | Interrupt line |
| **[in]** | irq_level | Interrupt priority level |

### 7.4.30. Function lcdca_set_config()

Configure the LCDCA controller.

```
void lcdca_set_config(
        struct lcdca_config * lcdca_cfg)
```

**Table 7-17 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | lcdca_cfg | Pointer to an LCDCA configuration structure |

### 7.4.31. Function lcdca_set_contrast()

Set the LCDCA fine contrast.

```
void lcdca_set_contrast(
        int8_t contrast)
```

Transfer function: VLCD = 3.0V + (fcont[5:0] * 0.016V)

**Table 7-18 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | contrast | Contrast value (range -32 to 31 inclusive) |

### 7.4.32. Function lcdca_set_display_memory()

Set all bits in the LCDCA display memory high.

```
void lcdca_set_display_memory( void )
```

### 7.4.33. Function lcdca_set_pixel()

Enable the specified pixel/segment in the LCDCA display memory.

```
void lcdca_set_pixel(
        uint8_t pix_com,
        uint8_t pix_seg)
```

**Table 7-19 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | pix_com | Pixel/segment COMx coordinate |
| **[in]** | pix_seg | Pixel/segment SEGy coordinate |

### 7.4.34. Function lcdca_toggle_pixel()

Toggle the specified pixel/segment in the LCDCA display memory.

```
void lcdca_toggle_pixel(
        uint8_t pix_com,
        uint8_t pix_seg)
```

**Table 7-20 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | pix_com | Pixel/segment COMx coordinate |
| **[in]** | pix_seg | Pixel/segment SEGy coordinate |

### 7.4.35. Function lcdca_unlock_shadow_dislay()

Unlock the LCDCA shadow display memory.

```
void lcdca_unlock_shadow_dislay( void )
```

### 7.4.36. Function lcdca_write_packet()

Send a sequence of ASCII bytes to the LCDCA via the digit decoder.

```
void lcdca_write_packet(
        uint8_t lcd_tdg,
        uint8_t first_seg,
        const uint8_t * data,
        size_t width,
        uint8_t dir)
```

**Note:** If a NULL byte is encountered, or if the *width* count expires, data will no longer be sent via the digit decoder and the function returns.

**Table 7-21 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | lcd_tdg | Type of digit decoder |
| **[in]** | first_seg | First SEG where the data will be written |
| **[in]** | data | ASCII data buffer pointer |
| **[in]** | width | Maximum number of data bytes |
| **[in]** | dir | Direction (0 for left to right, otherwise right to left) |

# 8. Extra Information for Liquid Crystal Display Controller Driver

## 8.1. Acronyms

Below is a table listing the acronyms used in this module, along with their intended meanings.

| Acronym | Definition |
|---|---|
| COM | Common |
| NVIC | Nested Vectored Interrupt Controller |
| PDCA | Peripheral DMA Controller |
| PMC | Power Manager Controller |
| QSG | Quick Start Guide |
| SEG | Segment |

## 8.2. Dependencies

This driver has the following dependencies:

- Peripheral DMA Controller (PDCA) driver
- System clock (sysclk)
- Interrupt management
- Sleep manager

## 8.3. Errata

There are no errata related to this driver.

## 8.4. Module History

An overview of the module history is presented in the table below, with details on the enhancements and fixes made to the module since its first release. The current version of this corresponds to the newest version in the table.

| Changelog |
|---|
| Initial document release |

# 9. Examples for Liquid Crystal Display Controller

This is a list of the available Quick Start Guides (QSGs) and example applications for SAM4L Liquid Crystal Display (LCDCA) Driver. QSGs are simple examples with step-by-step instructions to configure and use this driver in a selection of use cases. Note that a QSG can be compiled as a standalone application or be added to the user application.

- Quick Start Guide for the LCDCA Driver
- Liquid Crystal Controller - Example Interfacing to a C42048A Display

## 9.1. Quick Start Guide for the LCDCA Driver

This is the quick start guide for the SAM4L Liquid Crystal Display (LCDCA) Driver, with step-by-step instructions on how to configure and use the driver for a specific use case.

The use cases contain several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, for example, the main application function.

### 9.1.1. Use Cases

- LCDCA Basic Usage

### 9.1.2. LCDCA Basic Usage

This use case will demonstrate how to configure and use the LCDCA module to address an external LCD segment (C42048A).

### 9.1.3. Setup Steps

#### 9.1.3.1. Prerequisites

This module requires the following services:

- System clock (sysclk)
- Sleep Manager
- Peripheral DMA Controller (PDCA) driver

#### 9.1.3.2. Setup Code Example

Add these code segments to the main loop, or to a setup function in your application's C-file:

```
#define PORT_MASK  40
#define LCD_DUTY   LCDCA_DUTY_1_4
#define LCD_CONTRAST_LEVEL 30

struct lcdca_config lcdca_cfg;
```

```
// LCDCA Controller initialization
// - Clock,
// - Connect to C42364A glass LCD component,
// - Timing:  64 Hz frame rate & low power waveform, FC0, FC1, FC2
// - Interrupt: off.
    lcdca_clk_init();
lcdca_cfg.port_mask = PORT_MASK;
lcdca_cfg.x_bias = false;
```

```
lcdca_cfg.lp_wave = true;
lcdca_cfg.duty_type = LCD_DUTY;
lcdca_cfg.lcd_pres = false;
lcdca_cfg.lcd_clkdiv = 3;
lcdca_cfg.fc0 = 16;
lcdca_cfg.fc1 = 2;
lcdca_cfg.fc2 = 6;
lcdca_cfg.contrast = LCD_CONTRAST_LEVEL;
lcdca_set_config(&lcdca_cfg);
lcdca_enable();
lcdca_enable_timer(LCDCA_TIMER_FC0);
lcdca_enable_timer(LCDCA_TIMER_FC1);
lcdca_enable_timer(LCDCA_TIMER_FC2);

/* Turn on LCD back light */
ioport_set_pin_level(LCD_BL_GPIO, IOPORT_PIN_LEVEL_HIGH);
```

#### 9.1.3.3. Basic Setup Workflow

1. Initialize the LCDCA clock:
   ```
   lcdca_clk_init();
   ```
2. Configure the LCDCA module:
   ```
   lcdca_set_config(&lcdca_cfg);
   ```
3. Enable the LCDCA module:
   ```
   lcdca_enable();
   ```
4. Enable the frame counter timers:
   ```
   lcdca_enable_timer(LCDCA_TIMER_FC0);
   lcdca_enable_timer(LCDCA_TIMER_FC1);
   lcdca_enable_timer(LCDCA_TIMER_FC2);
   ```
5. Turn on the LCD backlight:
   ```
   ioport_set_pin_level(LCD_BL_GPIO, IOPORT_PIN_LEVEL_HIGH);
   ```

### 9.1.4. Usage Steps

#### 9.1.4.1. Normal Usage

The following functions can be used to set/clear/toggle one pixel/segment:

```
lcdca_set_pixel(ICON_ARM);
```

```
lcdca_clear_pixel(ICON_ARM);
```

```
lcdca_toggle_pixel(ICON_ARM);
```

The function lcdca_write_packet() can be used to display ASCII characters:

```
// Display in alphanumeric field.
lcdca_write_packet(LCDCA_TDG_14SEG4COM, FIRST_14SEG_4C, data, \
        WIDTH_14SEG_4C, DIR_14SEG_4C);
```

```
// Display in numeric field.
lcdca_write_packet(LCDCA_TDG_7SEG4COM, FIRST_7SEG_4C, data, \
        WIDTH_7SEG_4C, DIR_7SEG_4C);
```

The LCD contrast can be changed using:

```
lcdca_set_contrast(contrast_value);
```

#### 9.1.4.2. Using Hardware Blinking

To use hardware blinking:

```
blink_cfg.lcd_blink_timer = LCDCA_TIMER_FC1;
blink_cfg.lcd_blink_mode = LCDCA_BLINK_SELECTED;
lcdca_blink_set_config(&blink_cfg);
lcdca_set_pixel(ICON_ERROR);
lcdca_set_blink_pixel(ICON_ERROR);
lcdca_blink_enable();
```

#### 9.1.4.3. Using Hardware Autonomous Animation

To use the hardware's autonomous segment animation:

```
cs_cfg.lcd_csr_timer = LCDCA_TIMER_FC1;
cs_cfg.lcd_csr_dir = LCDCA_CSR_RIGHT;
cs_cfg.size = 7;    // Total 7-pixels.
cs_cfg.data = 0x03; // Display 2 pixel at one time.
lcdca_circular_shift_set_config(&cs_cfg);
lcdca_circular_shift_enable();
```

#### 9.1.4.4. Using Hardware Automated Character

To use hardware automated character (e.g., scrolling here):

```
struct lcdca_automated_char_config automated_char_cfg;
```

```
uint8_t const scrolling_str[] = \
    "Scrolling string display, Press PB0 to cont.  ";
```

```
automated_char_cfg.automated_mode = LCDCA_AUTOMATED_MODE_SCROLLING;
automated_char_cfg.automated_timer = LCDCA_TIMER_FC2;
automated_char_cfg.lcd_tdg = LCDCA_TDG_14SEG4COM;
automated_char_cfg.stseg = FIRST_14SEG_4C;
automated_char_cfg.dign = WIDTH_14SEG_4C;
/* STEPS = string length - DIGN + 1 */
automated_char_cfg.steps = sizeof(scrolling_str) - WIDTH_14SEG_4C + 1;
automated_char_cfg.dir_reverse = LCDCA_AUTOMATED_DIR_REVERSE;
lcdca_automated_char_set_config(&automated_char_cfg);
lcdca_automated_char_start(scrolling_str, strlen((char const
*)scrolling_str));
```

## 9.2. Liquid Crystal Controller - Example Interfacing to a C42048A Display

### 9.2.1. Purpose

This example demonstrates how to use the LCDCA driver to interface to an external LCD (C42048A).

### 9.2.2. Requirements

This example can be used with the SAM4L EK evaluation kit.

### 9.2.3. Description

The example configures the LCDCA Controller in a mode defined by the LCD glass board connections, and the technical characteristics of the LCD component used. It uses the following LCDCA controller features:

- LCD contrast control
- Hardware blinking
- Autonomous segment animation
- Automated character (sequential and scrolling)
- ASCII digit encoder
- LCD beginning of frame interrupt

The LCDCA is set up to use the external 32.768kHz clock to generate LCD frames at 64Hz, using a low power waveform to reduce toggle activity, and hence power consumption. In order to show the LCD Controller's capability of running in a power-saving mode, sleep mode is entered whenever possible.

### 9.2.4. Main Files

- lcdca.c: Liquid Crystal Display Controller driver
- lcdca.h: Liquid Crystal Display Controller driver header file
- lcdca_example.c: Liquid Crystal Display Controller example application
- conf_example.h: Liquid Crystal Display Controller example configuration header file

### 9.2.5. Compilation Information

This software is written for GNU GCC and IAR Embedded Workbench® for Atmel®. Other compilers may or may not work.

### 9.2.6. Usage

1. Build the program, and download it onto the evaluation board.
2. On the computer, open, and configure a terminal application (e.g., HyperTerminal on Microsoft® Windows®) with these settings:
   - 115200 baud
   - 8 bits of data
   - No parity
   - 1 stop bit
   - No flow control
3. Start the application.
4. In the terminal window, the following text should appear:

```
-- LCDCA Controller Example --
-- xxxxxx-xx
-- Compiled: xxx xx xxxx xx:xx:xx --

Press PB0 to stop automated sequential mode and continue.
```

# 10. Document Revision History

| Doc. Rev. | Date | Comments |
|---|---|---|
| 42296B | 07/2015 | Updated title of application note and added list of supported devices |
| 42296A | 05/2014 | Initial document release |

**Atmel Corporation** 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**