RX Series

Direct Drive LCD Design Guide (Internal Memory)

R01AN3384EG0100
Rev.1.00
July 30, 2016

1. Introduction

This document provides information of how to configure the LCD panel parameters required by Renesas LCD Direct Drive (Internal Memory) according to the LCD panel datasheet published by the manufacturers. This document will also describe all the APIs (Application Programming Interface) in the LCD Direct Driver and their usages. An overview of the system hardware is also provided.

Target Device

RX Series

Contents

1.1 Direct Drive LCD (Internal Memory) Overview

The RX device family includes several peripherals that enable the connection of RGB interface TFT panels directly to the MCU data bus and peripheral I/O. These peripherals include the DMA (DMA controller unit) and the TPU or MTU (Timer Units).

This document covers the Direct Drive LCD for Internal Memory. For details of the Direct Drive LCD that uses External Memory (e.g. SDRAM) for driving data to the LCD panel please refer to R21UT0232EG Direct Drive LCD Design Guide.

1.1.1 Philosophy

The Direct Drive LCD solution was developed to provide a low cost, long life solution for driving TFT panels for use in GUI applications with limited animation requirements. This solution reduces the risk for long life products by elimination of reliance on volatile components such as standalone LCD controllers, TFT panels with integrated LCD controllers, or application specific microprocessors.

1.1.2 Capabilities

The current features of the Direct Drive LCD (Internal Memory) solution are:

Frame buffer uses standard internal SRAM.

Create multiple frame buffers within the available SRAM.

Dynamically modify frame rate to accommodate varying system update requirements.

Drive RGB panels at 16bpp up to WQVGA resolution at up to 60Hz frame rates.

Pan larger display regions within a portion of the LCD panel area.

Very simple operation model: user code manipulates images in the frame buffer; the frame buffer is transparently transferred to the LCD panel.

The Direct Drive LCD solution is highly configurable, capable of producing many different timing configurations which drive the input signals of TFT-LCD panels from various panel manufacturers. The signal timing generated from the Direct Drive LCD solution depends on your choice panel resolution, frame buffer memory, and desired panel refresh and animation rates.

Although Renesas provides guidelines and examples for configuring the signal timing, Renesas is not responsible for meeting the AC timing specifications of your specific choice of TFT-LCD panel. Please contact your TFT-LCD panel manufacturer to ensure the Direct Drive LCD solution complies with the panel timing limitations.

2. Peripheral Usage Summary

A great deal of flexibility is offered by the Renesas RX peripherals and the DDLCD configuration macros.

This section provides an overview of how DDLCD uses these peripherals.

Timer channels are used to generate timing signals for the LCD panel. The timers directly drive the external pins without regular software intervention. These signals are: Dot Clock, Horizontal Sync, Vertical Sync and Panel Enable.

At the end of a horizontal line an interrupt is generated by the timer block and the DMA controller is configured to point at the data for the next horizontal line, held in internal RAM.

At the start of a horizontal line, an interrupt is generated that triggers the DMA to start transfer. The DMA controller briefly takes mastership of Internal Main bus 2 and Memory bus 1 to fetch one pixel of display data from the internal RAM. This data is transferred to the External Bus Controller to be driven on the external data bus pins. By virtue of the Write Buffer in the External Bus Control, the DMA controller can release mastership of the internal busses while the External Bus Controller drives the data to the LCD. This is repeated for each pixel in a horizontal line. This transfer mechanism minimises the number of cycles other bus masters are prevented from accessing internal RAM (e.g. the CPU). Access by the CPU to internal Flash and ECC RAM is unimpeded, these use Memory bus 2 and 3 respectively, which are not accessed by the DMA.

For further details please referrer to the ‘Buses’ chapter of the User’s Manual: Hardware for the device being used.

The diagram below is a simplified internal bus structure example to show the path of data from Internal RAM to the LCD panel, via the DMA Block. Data flow is highlighted in green.

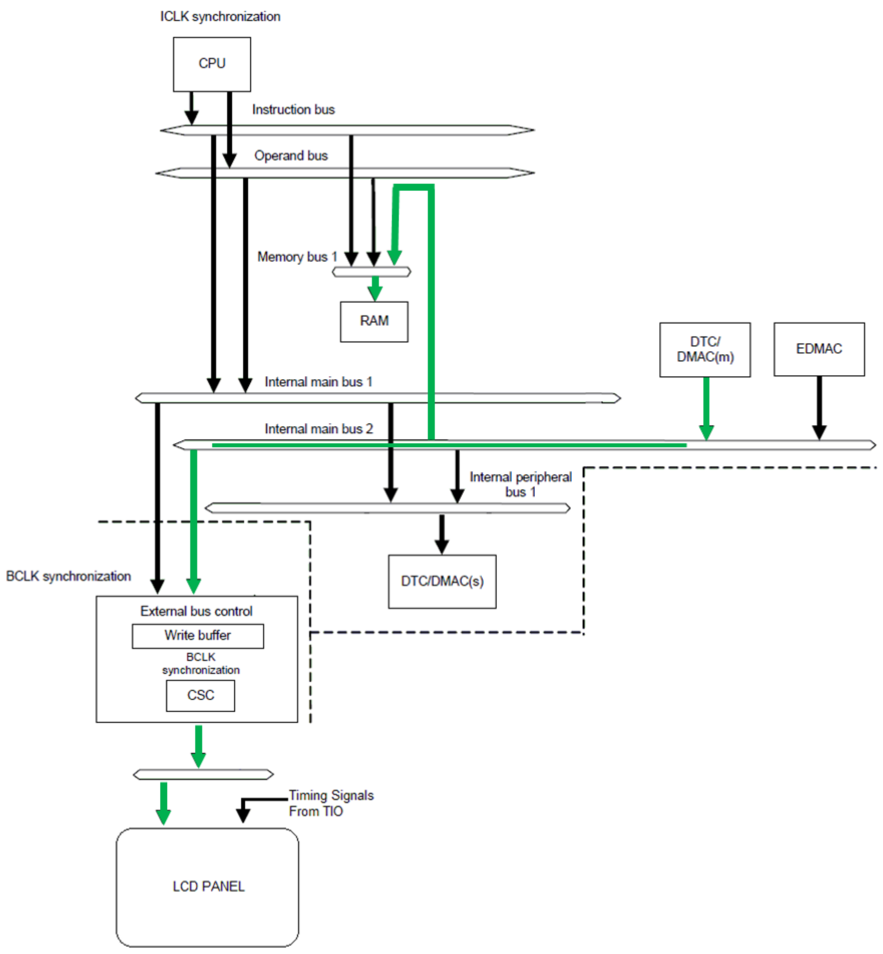


Figure 1. LCD Panel Data Flow

3. Driver Configuration

The LCD Direct Driver is configured through the setting of macro definitions. These macros are illustrated in the sample code. The following table briefly describes the function of each of these macros and their location in LCD Direct Drive demonstration code. For examples of each macro usage, refer to the demonstration code.

3.1 LCD Direct Drive Configuration Macros

Macro Name	Description	Units	File
BSP_ICLK_HZ	Clock frequency of MCU core	Hz	mcu_info.h
BSP_PCLKA_HZ	Clock frequency of peripherals	Hz	mcu_info.h
BSP_BCLK_HZ	Clock frequency of external bus	Hz	mcu_info.h
DD_FRAME_SIZE	Frame Buffer Configuration	Dots	config_r_ddlcd.h
DD_INVERT_V_LINES	Frame Buffer Configuration	Select	config_r_ddlcd.h
DD_INVERT_H_DOT	Frame Buffer Configuration	Select	config_r_ddlcd.h
DD_PANEL_ROTATE*[1]	Frame Buffer Configuration	Select	config_r_ddlcd.h
DD_RASTER_FRAME_COUNT	Frame Buffer Configuration	Frames	config_r_ddlcd.h
DD_FRAME_REGIONS_MAX	Frame Buffer Configuration	Regions	config_r_ddlcd.h
DD_MODE_SRAM_NOMUX*[2]	Driver Mode Selection	Select	config_r_ddlcd.h
DD_DOT_CLOCK_FREQUENCY_DATA	Driver Mode Configuration	Hz	config_r_ddlcd.h
DD_DOT_CLOCK_FREQUENCY_BLANK	Driver Mode Configuration	Hz	config_r_ddlcd.h
DD_DESIRED_FRAME_RATE	Driver Mode Configuration	Hz	config_r_ddlcd.h
DD_MINIMUM_MCU_ACCESS_PCT	Driver Mode Configuration	%	config_r_ddlcd.h
DD_INTERRUPT_PRIORITY	Driver Mode Configuration	Priority Level	config_r_ddlcd.h
DD_NOTIFY_INTERRUPT_PRIORITY	Driver Mode Configuration	Priority Level	config_r_ddlcd.h
DD_USE_DISPMEMORY_MANAGER	Bus Manager	Select	config_r_ddlcd.h
DD_DISPMEMORY_MANAGER_PRIORITY	Display Memory Manager	Task Priority	config_r_ddlcd.h
DD_DISPMEMORY_MANAGER_MAX_TASKS	Display Memory Manager	Count	config_r_ddlcd.h
DD_INVERT_DOT	LCD Panel Configuration	Select	config_r_ddlcd_panel.h
DD_V_LINES_PULSE	LCD Panel Configuration	Lines	config_r_ddlcd_panel.h
DD_V_LINES_BACK_PORCH	LCD Panel Configuration	Lines	config_r_ddlcd_panel.h
DD_V_LINES_DISPLAY	LCD Panel Configuration	Lines	config_r_ddlcd_panel.h
DD_V_LINES_FRONT_PORCH	LCD Panel Configuration	Lines	config_r_ddlcd_panel.h
DD_H_DOT_PULSE	LCD Panel Configuration	Dots	config_r_ddlcd_panel.h
DD_H_DOT_BACK_PORCH	LCD Panel Configuration	Dots	config_r_ddlcd_panel.h
DD_H_DOT_DISPLAY	LCD Panel Configuration	Dots	config_r_ddlcd_panel.h
DD_H_DOT_FRONT_PORCH	LCD Panel Configuration	Dots	config_r_ddlcd_panel.h

Macro Name	Description	Units	File
DD_FRAME_CS	Platform Configuration	CS #	config_r_ddlcd_platform.h
DD_DMAC	Platform Configuration	SFR root	config_r_ddlcd_platform.h
DD_VSYNC_PORT	Platform Configuration	Port #	config_r_ddlcd_platform.h
DD_VSYNC_PIN	Platform Configuration	Pin #	config_r_ddlcd_platform.h
DD_HSYNC_PORT	Platform Configuration	Port #	config_r_ddlcd_platform.h
DD_HSYNC_PIN	Platform Configuration	Pin #	config_r_ddlcd_platform.h
DD_DOTCLK_PORT	Platform Configuration	Port #	config_r_ddlcd_platform.h
DD_DOTCLK_PIN	Platform Configuration	Pin #	config_r_ddlcd_platform.h
DD_PANEL_POWER_PORT	Platform Configuration	Port #	config_r_ddlcd_platform.h
DD_PANEL_POWER_PIN	Platform Configuration	Pin #	config_r_ddlcd_platform.h
DD_BACKLIGHT_PORT	Platform Configuration	Port #	config_r_ddlcd_platform.h
DD_BACKLIGHT_PIN	Platform Configuration	Pin #	config_r_ddlcd_platform.h
DD_TMR_MTU	Platform Configuration	Select	config_r_ddlcd_platform.h
DD_DOTCLK_CHANNEL	Platform Configuration	Channel #	config_r_ddlcd_platform.h
DD_DOTCLK_TIOC	Platform Configuration	Pin Letter	config_r_ddlcd_platform.h
DD_DOTPER_CHANNEL	Platform Configuration	Channel #	config_r_ddlcd_platform.h
DD_DOTPER_TIOC	Platform Configuration	Pin Letter	config_r_ddlcd_platform.h
DD_HSYNC_CHANNEL	Platform Configuration	Channel #	config_r_ddlcd_platform.h
DD_HSYNC_TIOC	Platform Configuration	Pin Letter	config_r_ddlcd_platform.h
DD_HPER_CHANNEL	Platform Configuration	Channel #	config_r_ddlcd_platform.h
DD_HPER_TIOC	Platform Configuration	Pin Letter	config_r_ddlcd_platform.h
DD_VSYNC_CHANNEL	Platform Configuration	Channel #	config_r_ddlcd_platform.h
DD_VSYNC_TIOC	Platform Configuration	Pin Letter	config_r_ddlcd_platform.h
DD_VPER_CHANNEL	Platform Configuration	Channel #	config_r_ddlcd_platform.h
DD_VPER_TIOC	Platform Configuration	Pin Letter	config_r_ddlcd_platform.h
DD_HDEN_CHANNEL	Platform Configuration	Channel #	config_r_ddlcd_platform.h
DD_HDEN_TIOC	Platform Configuration	Pin Letter	config_r_ddlcd_platform.h
DD_HDEN2_CHANNEL	Platform Configuration	Channel #	config_r_ddlcd_platform.h
DD_HDEN2_TIOC	Platform Configuration	Pin Letter	config_r_ddlcd_platform.h
DD_PDEN_CHANNEL	Platform Configuration	Channel #	config_r_ddlcd_platform.h
DD_PDEN_TIOC	Platform Configuration	Pin Letter	config_r_ddlcd_platform.h
DD_PDEN2_CHANNEL	Platform Configuration	Channel #	config_r_ddlcd_platform.h
DD_PDEN2_TIOC	Platform Configuration	Pin Letter	config_r_ddlcd_platform.h

Notes:
*1 Rotation is not supported in Direct Drive LCD Internal Memory.
*2 SRAM_NOMUX is the only memory configuration supported by Direct Drive LCD Internal Memory.

3.2 Frame Buffer Configuration

The frame buffer is the internal SRAM memory area that is used to store the image data that will be presented on the LCD screen. Typically multiple frame buffers are allocated. This allows the MCU to be updating one frame while the LCD Direct Drive is transferring the other frame to the LCD panel, this behaviour allows for fast transitions and the user does not see operations occurring in the non-displayed buffers.

Typically, the frame buffer is configured to the same dimensions as the LCD panel; however the frame buffer can be larger to allow the LCD panel to act as a “window” into the frame buffer (allowing for fast panning of large images).

The following macros control the sizing and behaviour of the frame buffer.

3.2.1 DD_FRAME_SIZE

Defines the number of pixels in each of the frame buffers. The default setting is to make this equivalent to the LCD panel height multiplied by the LCD panel width. This value can be increased to allow panning within a larger display buffer.

3.2.2 DD_INVERT_V_LINES

If defined inverts the order of presentation of lines on the display.

3.2.3 DD_INVERT_H_DOT

If defined inverts the order of presentation of dots (columns) on the display.

3.2.4 DD_PANEL_ROTATE

Rotates the presentation of data (rows/columns) on the LCD panel. Rotation is not available in the Direct Drive LCD Internal Memory. Define maintained for compatibility with the external memory Direct Drive LCD variant.

3.2.5 DD_RASTER_FRAME_COUNT

Defines the number of frame buffers allocated in the driver. This value can be set to zero in which case, the user code is responsible for the allocation of frame buffers.

3.2.6 DD_FRAME_REGIONS_MAX

Defines the number of horizontal screen “splits” that can be used within the driver. The default setting is 1 (no splits). This capability allows different source raster regions to be used for different horizontal screen areas (control GUI + panning image view for example).

Refer to the “R_DDLCD_SetLineSource” API call for more information.

The default display sequence of a LCD panel is shown in **Figure 1**. The origin of the display is shown as the green dot in the picture. By default the driver will send the raster image to the LCD panel in the same sequence. If necessary, there are two macros available to change the sequence of data presented to the panel. **DD_INVERT_V_LINES** sends the top line first and sequences to the bottom and **DD_INVERT_H_DOT** sends the right side of the line first and sequences to the left. Either or both of these macros can be specified at the same time.

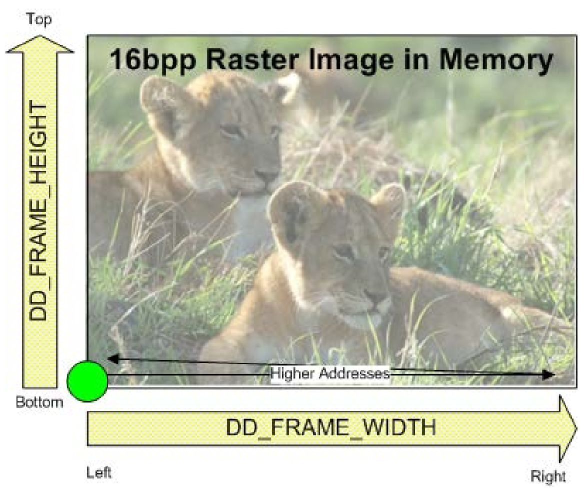


Figure 2. RAM Frame Raster Data

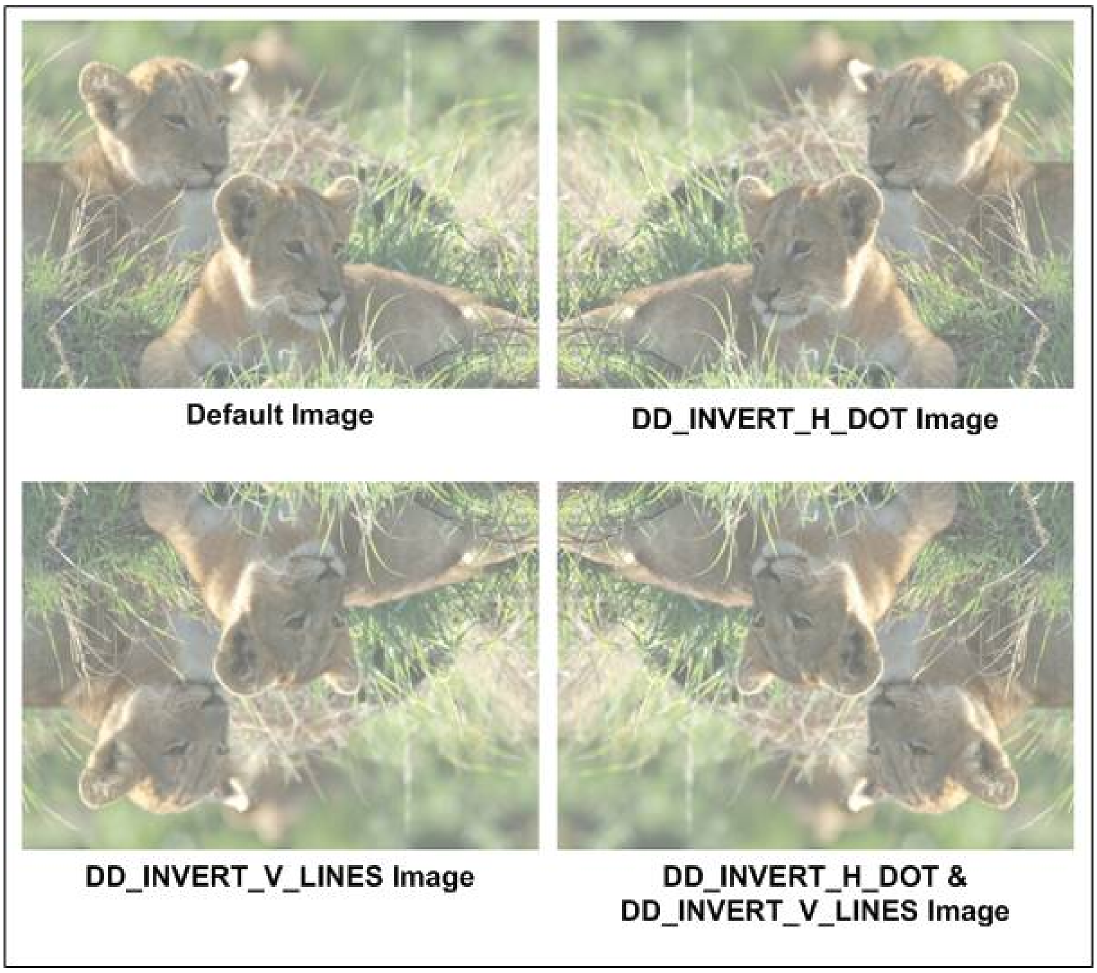


Figure 3. Images from Various Display Settings

3.3 Driver Mode Selection

Direct Drive LCD driver internal memory supports only internal SRAM. For details of the Direct Drive LCD that uses External Memory (e.g. SDRAM) for driving data to the LCD panel please refer to R21UT0232EG Direct Drive LCD Design Guide.

3.3.1 DD_MODE_SRAM_NOMUX

Defining this macro selects a mode of operation that uses SRAM as the frame buffer. In this mode, the timer unit supplies the dot clock during data transfer and blanking. PDEN is used as an interrupt source to trigger the DMA controller to transfer data from internal memory to the LCD Panel.

3.3.2 Driver Mode Configuration

The driver characteristics are configured with the following macros.

3.3.3 DD_DOT_CLOCK_FREQUENCY_DATA

This macro configures the dot clock frequency during the data transfer portion of the LCD update cycle. This value must be achievable by the configured BCLK_FREQUENCY and RAM configuration. This value is checked against other system parameters and an error will be generated if the value is not achievable.

3.3.4 DD_DOT_CLOCK_FREQUENCY_BLANK

This macro configures the dot clock frequency during the blanking portion of the LCD update cycle. This value is checked against other system parameters and an error will be generated if the value is not achievable.

3.3.5 DD_DESIRED_FRAME_RATE

This macro configures the initial selection of LCD frame rate. The frame rate can also be modified at runtime via the “R_DDLCD_SetFrameRate” API call. To achieve the desired frame rate, the vertical blanking time is extended beyond the values configured in the LCD panel configuration. This value is checked against other system parameters and an error will be generated if the value is not achievable.

3.3.6 DD_MINIMUM_MCU_ACCESS_PCT

This macro configures the user’s minimum acceptable percentage of time that Direct Drive LCD is not accessing the active frame buffer (accessing the active frame buffer outside of the vertical blanking period will create visible artefacts). This value interacts with DD_DESIRED_FRAME_RATE macro…higher access percentage is achievable at lower frame rates (as the bus is less consumed with frame updates). This value is checked against other system parameters and an error will be generated if the value is not achievable.

3.3.7 DD_INTERRUPT_PRIORITY

This macro configures the interrupt priority level of the LCD Direct Drive interrupts. As the LCD Direct Drive is sensitive to any latency, this priority should normally be the highest in the MCU (and beyond any RTOS interrupt level). These interrupts need to be serviced in the time allocated by the horizontal blanking period. If visible artefacts (jitter on first pixel location) are created by necessary higher priority interrupts, additional time can be added to the horizontal blanking period by increasing DD_H_DOT_FRONT_PORCH (**Note:** operation cannot be guaranteed if the Direct Drive is not the highest MCU priority).

3.3.8 DD_NOTIFY_INTERRUPT_PRIORITY

This macro configures the interrupt priority level that the LCD Direct Drive uses to signal an RTOS that the vertical blanking period has started/ended. The RTOS should use this signal to disable tasks that may write to the active frame buffer during the DMA transfers. Proper behaviour can be enabled by using the provided display memory manager.

3.4 Display Memory Manager

To coordinate the access to the memory allocated for the display between the MCU tasks and the DMA controller, the display memory manager can be configured for use with the following macros. By default, the display memory manager uses the FreeRTOS package (other RTOS solutions can easily be ported).

3.4.1 DD_USE_DISPMEMORY_MANAGER

Defining this macro includes the display memory manager module in the Direct LCD code.

3.4.2 DD_DISPMEMORY_MANAGER_PRIORITY

This macro definition controls the priority of the RTOS task that will in turn suspend/resume tasks that are currently accessing the active frame buffer. This task priority must be higher than the priority of any controlled task for proper operation.

3.4.3 DD_DISPMEMORY_MANAGER_MAX_TASKS

This macro definition configures the number of tasks that can be controlled by the display memory manager. This value allocates a data structure for each task (4 bytes each).

3.5 LCD Panel Configuration

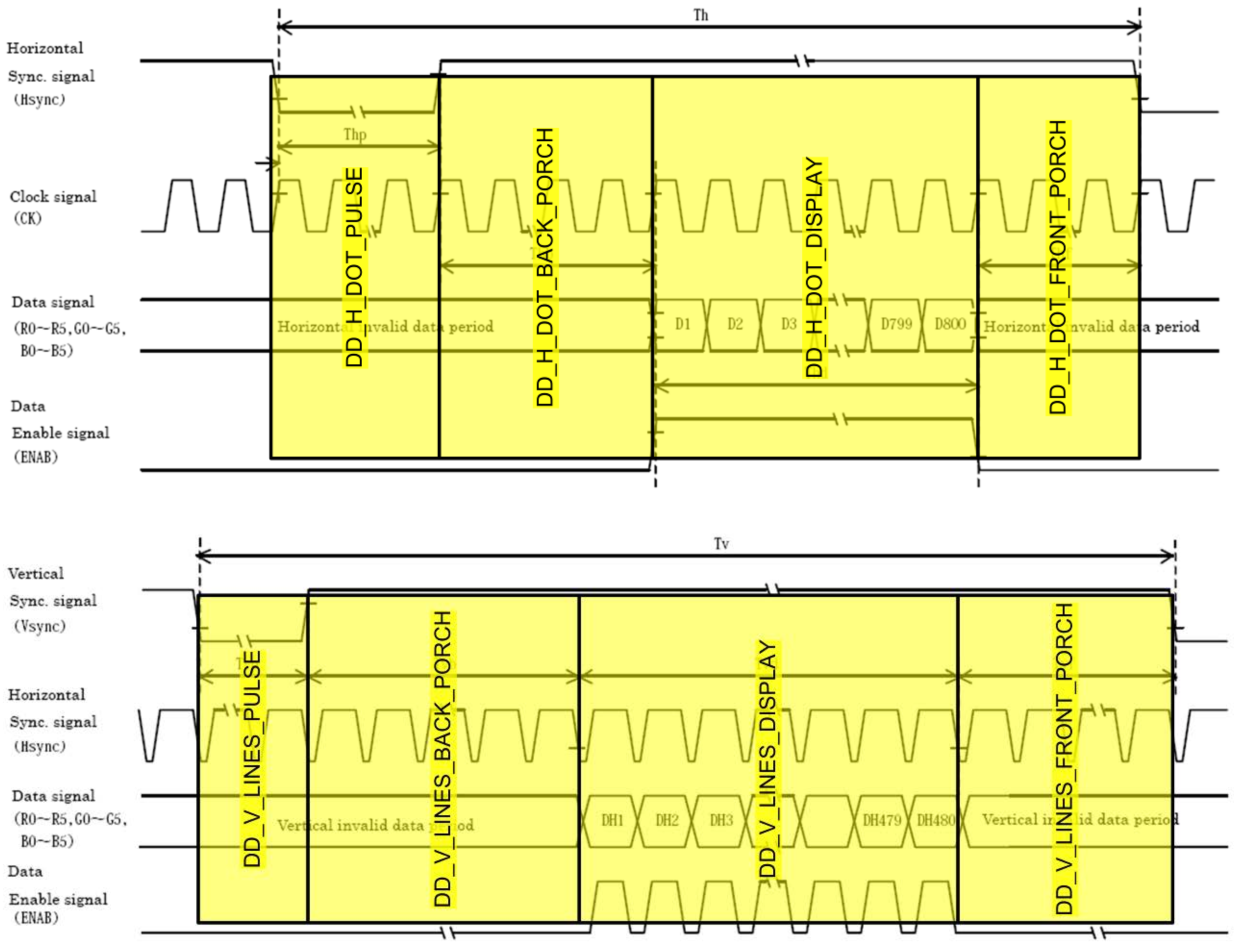
The LCD Direct Driver is configured to operate with a given LCD panel by setting macro definitions. These values are available in the data sheet for the selected panel.

3.5.1 DD_INVERT_DOT

This macro is used to control whether the RGB data is latched on the rising or the falling edge of the dot clock. If the macro is not defined, the data is latched on the rising edge, if it is defined, the data will be latched on the falling edge.

3.5.2 DD_V_LINES_xx and DD_H_DOT_xx

Refer to the following diagram for definition of these values.



Th
Horizontal Sync. signal (Hsync)
Thp
Clock signal (CK)
Data signal (R0~R5, G0~G5, B0~B5)
Data Enable signal (ENAB)
DD_H_DOT_PULSE
DD_H_DOT_BACK_PORCH
DD_H_DOT_DISPLAY
DD_H_DOT_FRONT_PORCH
Horizontal invalid data period
D1
D2
D3
D799
D800
Tv
Vertical Sync. signal (Vsync)
Horizontal Sync. signal (Hsync)
Data signal (R0~R5, G0~G5, B0~B5)
Data Enable signal (ENAB)
DD_V_LINES_PULSE
DD_V_LINES_BACK_PORCH
DD_V_LINES_DISPLAY
DD_V_LINES_FRONT_PORCH
Vertical invalid data period
DH1
DH2
DH3
DH479
DH480

Figure 4 LCD Panel Macro Definitions

3.6 LCD Platform Configuration

The LCD Direct Driver is configured to operate with a given hardware platform by setting macro definitions. These values will have to be determined from the schematics on the hardware platform. As an example, the demonstration code can be compared the LCD direct drive hardware schematics.

3.6.1 DD_FRAME_CS

This is the numeric value of the CS pin used for the frame buffer, for example if CS2 is used, a value of “2” would be entered.

3.6.2 DD_DMAC

Enter the name of the DMA being used for the LCD Direct Drive. For example, if DMA channel 0 is being used, set the value to “DMA0”

3.6.3 DD_<signal>_PORT and DD_<signal>_PIN

Enter the associated port and pin for the requested signal mapping. For example if the BACKLIGHT signal is on port PM1, set the port value to “M” and set the pin value to “1”.

3.6.4 DD_TMR_MTU and DD_TMR_TPU

Define one of these macros based on the timer unit you select for the LCD Direct Drive.

3.6.5 DD_<signal>_CHANNEL and DD_<signal>_TOIC

Enter the associated channel number and TOIC letter for the requested timer signal. For example if the DOTCLK signal is mapped to timer TIOCB0, enter “0” for the channel and “B” for the TOIC.

4. Typical LCD Panel Connections

This section illustrates typical connections on an LCD panel and how they are interfaced to the MCU in a Direct Drive configuration.

The LCD Direct Drive solution coordinates the timer unit with the DMA unit to ensure coherent presentation of these signals to properly place an image on the LCD panel. Proper operation depends on the correct configuration of the defined macros for a given panel.

4.1 LCD panel interface

Pin Number	Symbol	Description
1	**GND**	Ground (Vss)
2	**PCLK**	Pixel Clock
3	**HSYNC**	Horizontal Sync Signal
4	**VSYNC**	Vertical Sync Signal
5	**GND**	Ground (Vss)
6	**R0**	**Red Data Signal (LSB)**
7	**R1**	**Red Data Signal**
8	**R2**	**Red Data Signal**
9	**R3**	**Red Data Signal**
10	**R4**	**Red Data Signal**
11	**R5**	**Red Data Signal (MSB)**
12	**GND**	Ground (Vss)
13	**G0**	**Green Data Signal (LSB)**
14	**G1**	**Green Data Signal**
15	**G2**	**Green Data Signal**
16	**G3**	**Green Data Signal**
17	**G4**	**Green Data Signal**
18	**G5**	**Green Data Signal (MSB)**
19	**GND**	Ground (Vss)
20	**B0**	**Blue Data Signal (LSB)**
21	**B1**	**Blue Data Signal**
22	**B2**	**Blue Data Signal**
23	**B3**	**Blue Data Signal**
24	**B4**	**Blue Data Signal**
25	**B5**	**Blue Data Signal (MSB)**
26	**GND**	Ground (Vss)
27	**DEN**	Data Enable
28	**VDD**	VDD (3.3 VDC)
29	**VDD**	VDD (3.3 VDC)
30	**R/L**	Right/Left configuration
31	**U/D**	Up/Down configuration
32	**V/Q**	VGA/QVGA mode select
33	**GND**	Ground (Vss)

Figure 5 Example Connections for a Kyocera TFT-LCD Panel

4.1.1 Power Supplies

Many panels require multiple supplies. Check your panel's specification to see how many ground and different voltage level connections it requires.

In the example case of a Kyocera 320x240 panel, six 0v (GND) lines are required, along with two +3.3v reference voltages. In addition, the backlight power supply is also required.

4.1.2 Clock

Often referred to as the Dot Clock or Pixel Clock, the panel requires a synchronous clock signal to provide logic edges for clocking in data. The Red-Green-Blue (RGB) parallel data should be present on the data bus at the time of each rising edge of the clock; this provides the color setting for each pixel.

4.1.3 HSync

Hsync provides synchronization for each line of data presented to the panel. Each period of HSync is equivalent in time to one complete line on the panel (including the horizontal blanking time).

4.1.4 VSync

Vsync provides synchronization for each frame of data presented to the panel. Each period of VSync is equivalent in time to one complete refresh of the LCD panel (including the vertical blanking time). Typical VSync rates range from 30Hz to 80Hz depending on the LCD panel.

4.1.5 Data Enable

Many panels require an additional signal to frame the valid data. This enable signal is driven active while valid data is present on the RGB bus. It provides added synchronization for the timing of data, but this signal can often be left in the active state and the panel will revert to a fixed horizontal back porch (refer to LCD panel specification).

4.1.6 RGB (Red Green Blue) Data

The data is presented to the panel in parallel. LCD panels have connections for 6 or 8 bits of data for each color totalling 18 or 24 bits of color resolution. The LCD Direct Drive solution utilizes a 16 bit data bus, the format of the data presented to the RGB bus is R5:G6:B5 (corresponding to D15:D0). The color information presented by the MCU represents the most significant bits of color for the channel. Please refer to the reference designs for proper connections for the undriven least significant bits of each color channel.

5. Hardware Design

Below is a block diagram of a LCD Direct Drive system. Please refer to the schematics for the LCD Direct Drive demonstration kits for detailed reference design information.

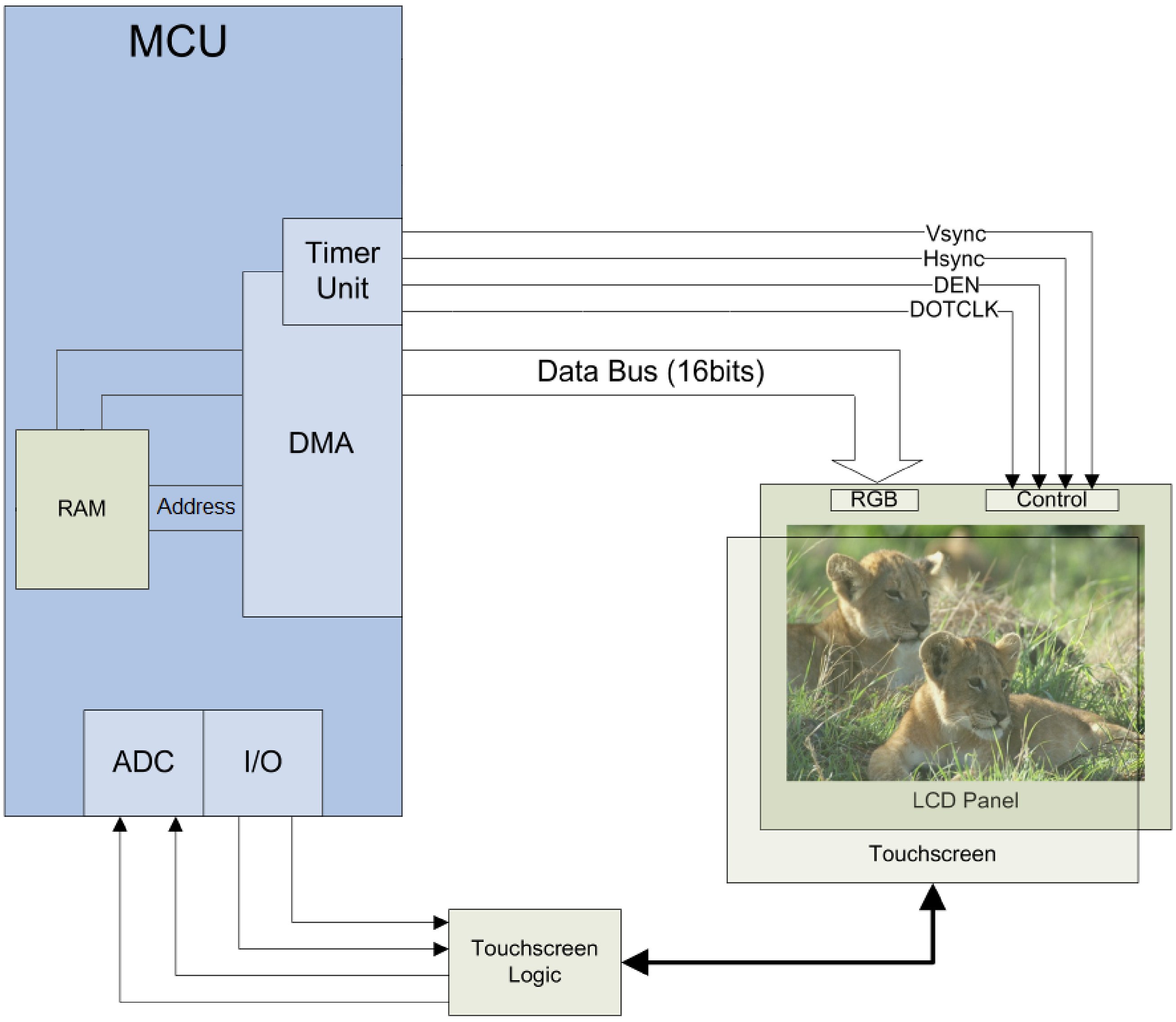


Figure 6 Block Diagram

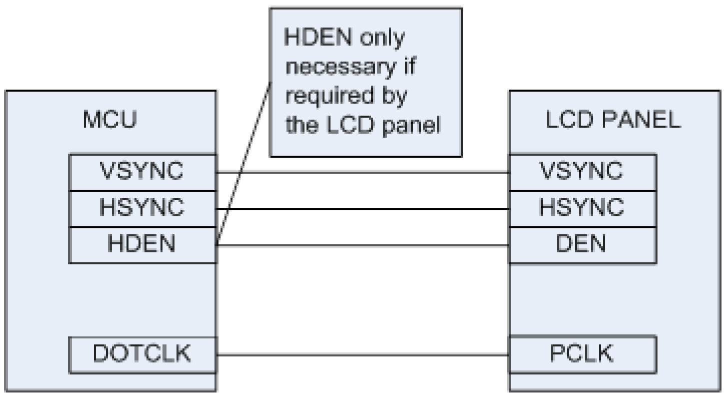
5.1 LCD Panel Control Logic

This section describes the HSYNC, VSYNC, DEN, and DOT CLK connections to the LCD panel. These connections differ based on the driver mode selection.

5.1.1 DD_MODE_SRAM_NOMUX

From the microcontroller, signals are directly connected to the LCD panel.

HDEN connection to the LCD panel is only needed if required by the panel.



5.2 Touch Screen Logic

Our support is currently for popular resistive touch screen panels which have 4 connections (endpoints of an X axis resistance and Y axis resistance). These inputs provide resistances proportional to the touched location particular X and Y coordinates on the panel.

The host system (microcontroller plus logic) drives the resistive endpoints with a known reference voltage, and the level on the channel is read into an analog to digital converter (ADC). With calibration and scaling in the microcontroller driver code, it is possible to pinpoint the area of the panel that was touched. Action can be taken accordingly.

An example of interface circuitry between the MCU and touch-screen is shown below.

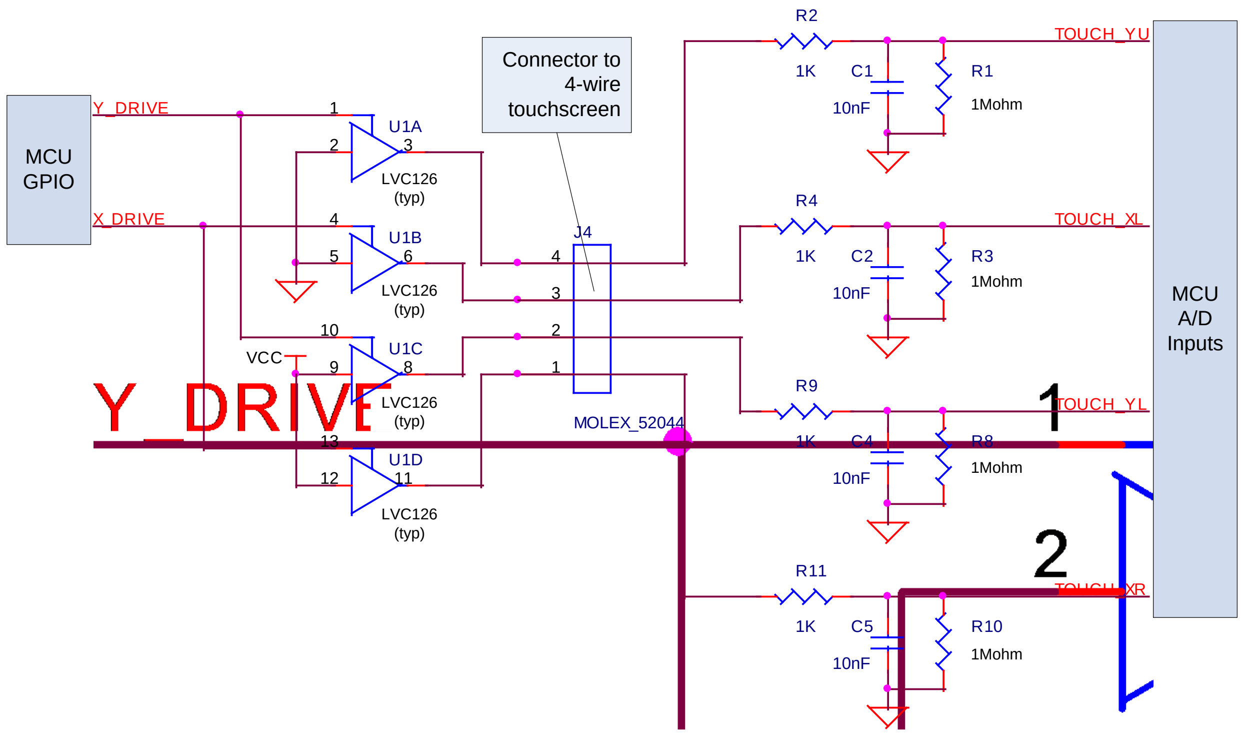


Figure 7 Touch Screen Circuit

6. API Information

The LCD Direct Drive API follows the Renesas API naming standards.

Additionally, it follows the Renesas Integrated Firmware Development conventions.

6.1.1 Header Files

All API calls are accessed by including a single file “r_ddlcd_public.h”. This header file in turn references several header files of the Direct Drive package. These “sub-header” files are all named “r_ddlcd_xxx.h” where “xxx” denotes a class of functions.

6.1.2 Configuration

The LCD Direct Drive package configuration is located in the following files.

Configuration File	**Configuration Contents**
config_r_ddlcd.h	General LCD Direct Drive behaviour configuration.
config_r_ddlcd_panel.h	LCD Direct Drive configuration related to the user selected LCD panel.
config_r_ddlcd_platform.h	LCD Direct Drive configuration related to the target hardware.

In addition to the previously described macro definitions the following data type is configurable.

Attribute	**Purpose**
Raster_t	The Raster_t typedef specifies the data type of the raster memory. Typically this will be set to “uint16_t”.

After any configuration changes, the package will need to be rebuilt.

6.1.3 Integer Types

The LCD Direct Drive API uses ANSI C 99 “Exact width integer types” in order to make the code clearer and more portable. These types are defined in “stdint.h”.

6.1.4 Global Variables

These read only variables are available.

```
   extern volatile uint16_t DDLCD_Vcount;    // frame counter
   extern volatile uint16_t DDLCD_BusActive; // flag indicating is DMA is active
```

6.1.5 Dependencies

The LCD Direct Drive API requires the FreeRTOS package if DD_USE_DISPMEMORY_MANAGER is defined.

6.1.6 Data Sections

In addition to the standard compiler sections, the LCD Direct Drive API uses the BDD_RAS_INT linker section to locate the display frames. The user project linker options must properly map this section to the appropriate memory location in internal SRAM).

In the case where DD_RASTER_FRAME_COUNT is not zero, the API will allocate the frame memory. In the case where DD_RASTER_FRAME_COUNT is zero, it is the user responsibility to allocate the frame memory in the BDD_RASTERS section.

7. LCD Direct Drive API Definition

7.1 LCD Initialization and Control

The following API calls are responsible for initialization of the LCD Direct Drive (based on configuration macro settings) and runtime control.

7.1.1 R_DDLCD_Init

Direct Driver Initialization.

Format

```
   DDLCD_Resp_t R_DDLCD_Init(void);
```

Parameters

none

Return Values

0 if successful, non-zero if failure.

Properties

Prototyped in file ”r_ddlcd_public.h”

Description

This function initializes the hardware necessary for the Direct Drive LCD to execute. This function uses the configuration macros to set up the timer unit and DMA peripherals to transfer data from the frame RAM to the LCD panel. After this function successfully executes the transfer of data to the panel by DMA will start and interrupts will be generated on every line to service the DMA (these interrupts are serviced by code within the LCD Direct Drive).

Example

```
   {
     DDLCD_Resp_t error = R_DDLCD_Init ();
     if (error != 0) …
   }
```

7.1.2 R_DDLCD_Backlight

Direct Driver backlight control.

Format

```
   void R_DDLCD_Backlight(int state);
```

Parameters

state

Requested backlight state 0=off, non-0 = on.

Return Values

None

Properties

Prototyped in file ”r_ddlcd_public.h”

Description

This function is used to control the state of the LCD backlight.

Example

```
   {
     R_DDLCD_Backlight(1);    /* turn backlight on */
   }
```

7.1.3 R_DDLCD_SetFrameRate

Configure the vertical refresh rate of the LCD panel.

Format

```
   int16_t R_DDLCD_SetFrameRate(int16_t rate);
```

Parameters

rate

Requested refresh rate (in Hz). Acceptable values are dependent on the attached LCD panel.

Return Values

Negative value indicates rate was not able to be achieved with system configuration. Positive value indicates success, returned value will be the percent of MCU access time available.

Properties

Prototyped in file ”r_ddlcd_public.h”

Description

This function is used to control the vertical refresh rate of the LCD panel. This function can be used to dynamically adapt the MCU access time based on system conditions. For example, prior to a full buffer refresh, the rate can be dropped to increase access time, than it can be restored to previous value for normal operation.

Example

```
   {
     int16_t success = R_DDLCD_SetFrameRate(60); /* set frame rate to 60Hz */
     if (success < 0)… /* process error */
   }
```

7.1.4 R_DDLCD_GetFrameRate

Request the vertical refresh rate of the LCD panel.

Format

```
   int16_t R_DDLCD_GetFrameRate(void);
```

Parameters

none

Return Values

Current frame rate in Hz.

Properties

Prototyped in file ”r_ddlcd_public.h”

Description

Request the current vertical refresh rate of the LCD panel.

Example

```
   {
     /* get frame rate prior to change */
     int16_t old_rate = R_DDLCD_GetFrameRate();
   }
```

7.1.5 R_DDLCD_SetActiveRaster

Set memory frame to display.

Format

```
   void * R_DDLCD_SetActiveRaster(uint16_t frame);
```

Parameters

frame

Requested frame buffer index.

Return Values

Pointer to first pixel of frame raster.

Properties

Prototyped in file ”r_ddlcd_public.h”

Description

Request to make the requested frame buffer the one actively displayed on the LCD panel.

Note that R_DDLCD_SetActiveRaster, R_DDLCD_SetRasterOffset and R_DDLCD_SetLineSource are similar in function and interact.

Example

```
   uint16_t  frame_request;
   void* select_buffer;
           ……
   frame_request = 1;  /* Select frame 1 to display */
   select_buffer = R_DDLCD_SetActiveRaster(frame_request);   /* switch buffer */
```

7.1.6 R_DDLCD_GetActiveFrame

Request which memory frame is currently displayed

Format

```
   uint16_t R_DDLCD_GetActiveFrame(void);
```

Parameters

none

Return Values

Index of active frame raster.

Properties

Prototyped in file ”r_ddlcd_public.h”

Description

Request which memory frame is currently displayed. Note that this function only returns valid information when LCDSetActiveRaster is used to control the display content (as opposed to LCDSetLineSource).

Note that R_DDLCD_SetActiveRaster, R_DDLCD_SetRasterOffset and R_DDLCD_SetLineSource are similar in function and interact.

Example

```
   uint16_t  frame_request;
            ……
   frame_request = R_DDLCD_GetActiveFrame();
   /* switch buffers */
   if (frame_request == 0)
     R_DDLCD_SetActiveRaster(1);
   else
     R_DDLCD_SetActiveRaster(0);
```

7.1.7 R_DDLCD_SetRasterOffset

Request display location within larger raster image

Format

```
   int16_t R_DDLCD_SetRasterOffset(int16_t x, int16_t y);
```

Parameters

x

X offset in pixels within the raster image.

y

Y offset in pixels within the raster image.

Return Values

0 on success, non-0 on failure

Properties

Prototyped in file ”r_ddlcd_public.h”

Description

R_DDLCD_SetRasterOffset changes the display position within the raster. The offset is limited to be within the area allocated by the FRAME_HEIGHT x FRAME_WIDTH space. If raster is the same size as the panel, the offset cannot be changed (fixed to 0,0).

Note that R_DDLCD_SetActiveRaster, R_DDLCD_SetRasterOffset and R_DDLCD_SetLineSource are similar in function and interact.

Example

```
   int16_t x = 40, y=20;
      ……
   if (R_DDLCD_SetRasterOffset (x, y) !=  0)     //set raster offset
      // handle error;
```

7.1.8 R_DDLCD_SetLineSource

Defines the source regions of the active display window.

Format

```
   int16_t R_DDLCD_SetLineSource
   (int16_t Region, int16_t LineCount, void *pSource, int16_t LineStep);
```

Parameters

Region

Region of display (horizontal strip). Ranging from 0 to MAX_FRAME_REGIONS (configuration MACRO) Normally, region 0 starts at the bottom of the screen. However; when V_LINES_INVERT is defined to change line presentation on the screen, region 0 will start at the top of the screen. MAX_FRAME_REGIONS should be set to 1 if multiple regions are not used (this will eliminate any associated runtime overhead).

LineCount

Is the number of lines associated with this region. This value can vary from 1 to V_LINES_PANEL.

pSource

Address of the first pixel of the first line within the region. The entire memory space of the region must be within the “DD_RASTERS” section, or the request will not be accepted.

LineStep

Distance (in pixels/Raster_t’s) from first pixel of first line to first pixel of second line (source regions can be wider than the panel).

Return Values

0 on success, non-0 on failure

Properties

Prototyped in file ”r_ddlcd_public.h”

Description

R_DDLCD_SetLineSource defines the source regions of the active display window.

Note that R_DDLCD_SetActiveRaster, R_DDLCD_SetRasterOffset and R_DDLCD_SetLineSource are similar in function and interact.

Example

```

   #pragma section LCD_Frames
   // SRAM allocated for GUI display
   uint16_t GUI_buffer[50 * DD_H_DOT_DISPLAY];
   // allocate panning buffer 4x panel
   uint16_t Image_buffer[2* DD_V_LINES_DISPLAY * 2 * DD_H_DOT_DISPLAY];
   #pragma section
      ……
   (void) R_DDLCD_SetLineSource (0,50,GUI_buffer, DD_H_DOT_DISPLAY); //GUI
Region
   (void) R_DDLCD_SetLineSource (1, DD_V_LINES_DISPLAY -50,
    &Image_buffer[ offset], 2*DD_H_DOT_DISPLAY); //Pan Region
      ……
```

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision History

		Description	
Rev.	Date	Page	Summary
1.0	30/07/2016	All	First release

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0