LCD

LPC5460X FEATURES

HOT 1
LCD BASIC

HOT introduction

- **Objective：** Understanding how to
  - Defines LCD parameters and use SDK APIs to initialize LCD controller, start LCD operation
  - Allocate framebuffer in SDRAM w/o having to initialize SDRAM before main()
  - Draw on framebuffer
- **Description：** Initializes LCD controller, SDRAM, and draw on framebuffer
- **Result:** 8 color stripes moving on LCD screen

Environment

- Hardware
  - Evaluation board：LPC54600Xpresso board

  - Project location
  - (keil) .\boards\lpcxpresso54608\examples\examples_lcdc\lcdc_1_tft16bpp\mdk
  - (mcux) import “lcdc_1_tft16bpp” project

Board configuration

- Default configuration
- (Optional) Could use debug UART to output trace log
  - Connect micro-USB to J8
  - Settings: 115200, n, 8, 1

HOT steps

- 1 – Watch result：Compile, download, run.

- 2 – Debug：
  - Enter debug mode，press ”F10” to step over or “F11” to step into

- 3 – Extension：
  - Change the color of stripes by modifying “colTab” array.
  - Change panel clock frequency macro “LCD_PANEL_CLK”, suggested range 4-16 MHz
  - Learn the members of “lcdc_config_t”, how they map to LCD controller to registers.
  - Check the “Flash.sct” and the definition of “s_FB” to see how to make framebuffer placed into SDRAM w/o involving compiler to generate initialization code before main().
    - Otherwise, w/o SDRAM initialized, accessing SDRAM will cause hard fault.

Key API usage and code

Initialize SDRAM for framebuffer availability

```
BOARD_InitSDRAM();
```

Enabled clock to LCD controller

```
CLOCK_SetClkDiv(kCLOCK_DivLcdClk, 1, true);
```

Initialize LCD controller with specified parameters, including panel clock, resolution, color format, timings, framebuffer address.

```
LCDC_GetDefaultConfig(&lcdConfig);
lcdcInFreq = CLOCK_GetFreq(kCLOCK_LCD);
LCDC_Init(LCD, &lcdConfig, lcdcInFreq);
```

Start LCD controller and power up LCD

```
LCDC_Start(LCD);
LCDC_PowerUp(LCD);
```

Drawing on frame

```
Refer to main loop.
```

HOT 2
DUAL
FRAMEBUFFER

HOT introduction

- **Objective：** Based on HOT1, uderstanding how to
  - Defines 2 framebuffers
  - Using LCD's "base address update" interrupt to safely draw to background FB.
  - Draw on framebuffer
- **Description：** Repeating drawings in main loop: first clear the screen to black, then draw color stripes. Use SysTick timer to limit draw rate.
  - If "SW5" is not pressed, then use one FB to draw,
  - if "SW5" is pressed, then waits for "base address update" IRQ, then draws on backup FB (the previous active FB), after drawing, set the next active FB to this FB.
- **Result:** If "SW5" is not pressed, then black screen and color stripes shows on screen interleaved, get flicker feeling; if "SW5" is pressed, only the rotating color stripes are showed (like HOT1).

Environment

- Hardware
  - Evaluation board：LPC54600Xpresso board

  - Project location
  - (keil) .\boards\lpcxpresso54608\examples\examples_lcdc\lcdc_2_tft16bpp_2fb\mdk
  - (mcux) import “lcdc_2_tft16bpp_2fb” project

Board configuration

- Default configuration
- (Optional) Could use debug UART to output trace log
  - Connect micro-USB to J8
  - Settings: 115200, n, 8, 1

HOT steps

- 1 – Watch result：Compile, download, run. See different drawing effects when “SW5” is pressed and not pressed.
- 2 – Debug：
  - Enter debug mode，press ”F10” to step over or “F11” to step into
  - Analyze and check how the FBs are switched with “s_actFBNdx” variable.
- 3 – Extension：
  - Change SysTick rate, check if it can resolve the flicker effect w/o dual-FB, and/or affects dual-FB effect.
  - Comment out the “while (!s_frameAddrUpdated){}”, see if it affects dual-FB effect.
  - Switch stage1 and stage2, check the differences of LCD display for single FB and dual-FB respectively.

Key API usage and code (Omitted common parts in HOT1)

Enable LCD “base address update” interrupt

```
LCDC_EnableInterrupts(LCD, kLCDC_BaseAddrUpdateInterrupt);
```

IRQ handler: Get LCD interrupt flag and clear in LCD IRQ handler, set the s/w level notify ---- “s_frameAddrUpdated = true;”

```
void LCD_IRQHandler(void) {…}
intStatus = LCDC_GetEnabledInterruptsPendingStatus(LCD);
LCDC_ClearInterruptsStatus(LCD, intStatus);
if (intStatus & kLCDC_BaseAddrUpdateInterrupt) {…}
```

Update FB address after background FB drawing is done, and switch the active/background FB.

```
LCDC_SetPanelAddr(LCD, kLCDC_UpperPanel, (uint32_t)(pFB32));
s_actFBNdx = !s_actFBNdx;
```

Background code: Wait for “s_frameAddrUpdated” to become true before drawing next frame.

```
while (!s_frameAddrUpdated){}
```

HOT 3
PALETTE

HOT introduction

- **Objective：**
  - Understands how to use palette to put framebuffer in SRAM, instead of SDRAM
  - Palette color settings
- **Description：** Draw moving rectangle periodically. Every period is synchronized to a new LCD base address update IRQ. The examples implements a rectangle draw & fill routine with 2bpp mode.
- **Result:** There is a rectangle moving smoothly and when reach a edge (either left, top, right ,bottom), it changes color and bounces.

Environment

- Hardware
  - Evaluation board：LPC54600Xpresso board

  - Project location
  - (keil) .\boards\lpcxpresso54608\examples\examples_lcdc\lcdc_3_palette\mdk
  - (mcux) import “lcdc_3_palette” project

Board configuration

- Default configuration
- (Optional) Could use debug UART to output trace log
  - Connect micro-USB to J8
  - Settings: 115200, n, 8, 1

HOT steps

- 1 – Watch result：Compile, download, run.
- 2 – Debug：
  - Enter debug mode，press ”F10” to step over or “F11” to step into
- 3 – Extension：
  - Locate palette color settings and change color to see the result. Color is RGB565 format.
  - Change the moving speed of rectangle, either X or Y.

Key API usage and code (Omitted common parts in previous HOT)

Setup palette colors

```
static const uint32_t palette[] = {0x001F0000U, 0x7C0003E0U};
```

Set palette buffer

```
LCDC_SetPalette(APP_LCD, palette, ARRAY_SIZE(palette));
```

Update FB address after background FB drawing is done, and switch the active/background FB.

```
LCDC_SetPanelAddr(LCD, kLCDC_UpperPanel, (uint32_t)(pFB32));
s_actFBNdx = !s_actFBNdx;
```

HOT 4
H/W CURSOR

HOT introduction

- **Objective：**
  - Understands how to setup cursor bitmap, including transparent and XOR colors
  - Set new position of cursor synchronized with LCD vertical back porch.
- **Description：** Draw and moves cursor periodically. Every period is synchronized to a new LCD vertical back porch IRQ.
- **Result:** There is a cursor moving smoothly and when reach a edge (either left, top, right ,bottom), it bounces.

Environment

- Hardware
  - Evaluation board：LPC54600Xpresso board

  - Project location
  - (keil) .\boards\lpcxpresso54608\examples\examples_lcdc\lcdc_4_cursor\mdk
  - (mcux) import “lcdc_4_cursor” project

Board configuration

- Default configuration
- (Optional) Could use debug UART to output trace log
  - Connect micro-USB to J8
  - Settings: 115200, n, 8, 1

HOT steps

- 1 – Watch result：Compile, download, run.
- 2 – Debug：
  - Enter debug mode，press ”F10” to step over or “F11” to step into
- 3 – Extension：
  - cursor position update synchronized to vertical back porch, to verify if this is required, comment out the “while (!s_frameEndFlag){}” and uncomment “while (!s_isNewTick){}” to see the change of cursor movement (Which one is more smoother?).

Key API usage and code

Defines the bitmap (w/ transparency and XOR “colors”) of cursor

```
static const uint8_t cursor32Img0[]
```

Configure cursor

```
lcdc_config_t lcdConfig;
LCDC_CursorGetDefaultConfig(&cursorConfig);
cursorConfig.size = kLCDC_CursorSize32;
cursorConfig.syncMode = kLCDC_CursorSync;
cursorConfig.image[0] = (uint32_t *)cursor32Img0;
```

Select cursor image number (0 to 3). LCDC supports 64x64, divided into 4 32x32 images like a “田”. As we just setup one left-top 32x32, the number is 0.

```
LCDC_ChooseCSeursor(APP_LCD, 0);
```

Cursor update is synchronized to vertical back porch, so enabled the “vertical compare” IRQ and select vertical back porch as IRQ trigger source.

```
LCDC_SetVerticalInterruptMode(APP_LCD, kLCDC_StartOfBackPorch);
LCDC_EnableInterrupts(APP_LCD, kLCDC_VerticalCompareInterrupt);
NVIC_EnableIRQ(APP_LCD_IRQn);
```

Enable H/W cursor layer

```
LCDC_EnableCursor(APP_LCD, true);
```

Update cursor location after a new vertical back porch IRQ fired.

```
LCDC_SetCursorPosition(APP_LCD, cursorPosX, cursorPosY);
```

SECURE CONNECTIONS
FOR A SMARTER WORLD