# LCD

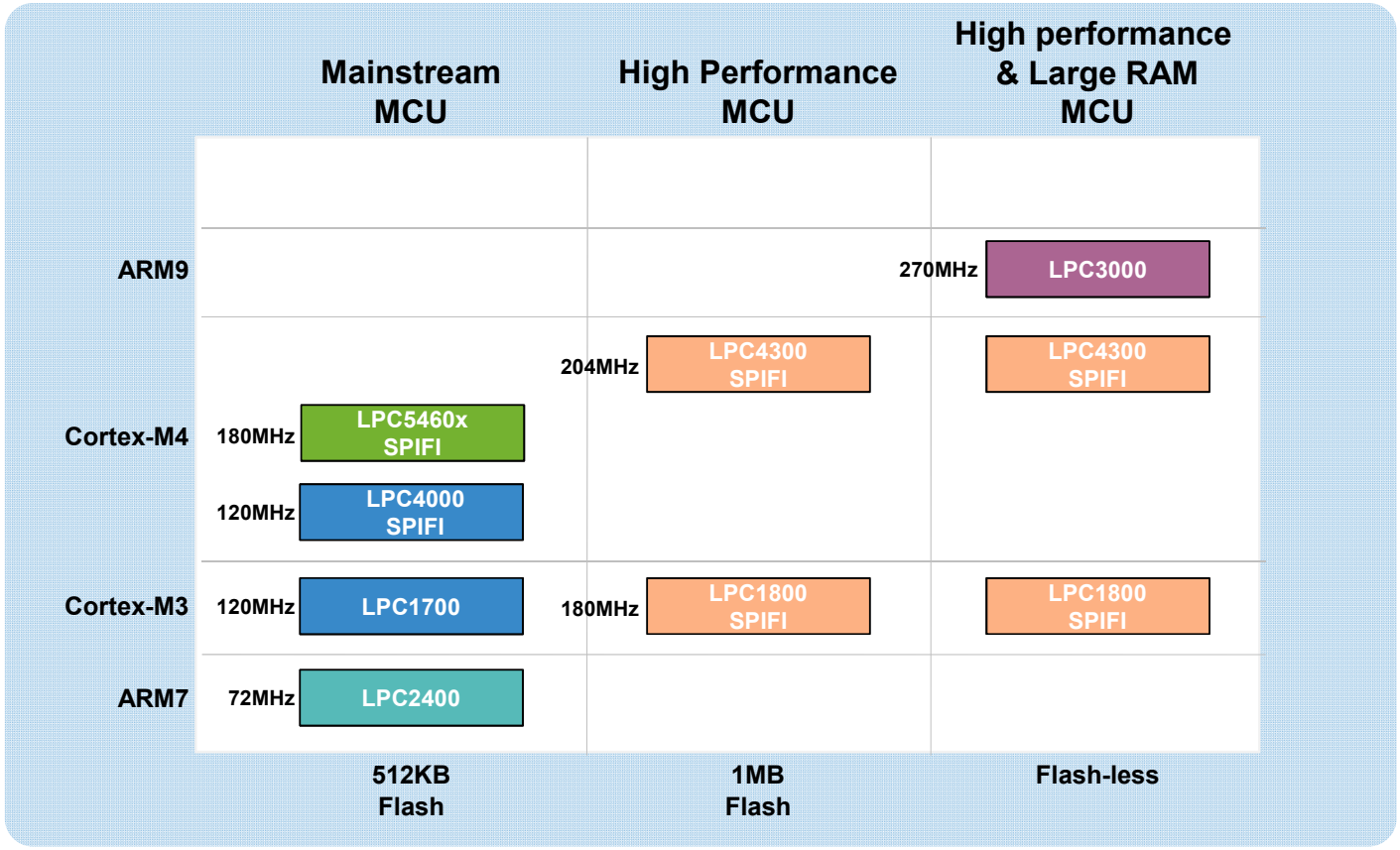
## LPC MCU KEY FEATURES

EXTERNAL USE

SECURE CONNECTIONS
FOR A SMARTER WORLD

# LCD controller availability

## Driving Displays with More than 40 parts & 10 years of market experience



# Graphics LCD Controller


- Key features
  - Support for STN and TFT panels
  - **Up to 1024x768 resolution**
  - 24-bit LCD interface supports 24bpp (16M colors)
  - Palette table allows display of up to 256 of 64K colors
  - Adjustable LCD bus size supports various panel bus configurations
  - Dedicated LCD DMA controller
  - Hardware cursor support
- Graphic Library Support
  - Segger's emWin graphic library free to use with NXP's microcontrollers
  - Other supported graphic libraries include Draupner's TouchGFX and ExpressLogic's GUIX
- Chip drivers and Board Support Packages
  - Significantly reduces your software porting efforts
  - Porting guide available for non-standard LCDs

# APPLICATION OF DISPLAYS

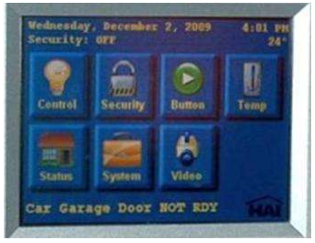
# Graphic versus Segment LCD

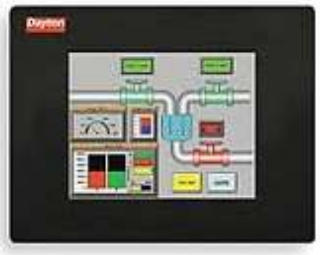

7seg 15seg


**Segment LCDs**
Advantages of segment-driven LCD displays are low communication overhead, low power, and virtually limitless but fixed display configurations.

**Graphic LCDs**
Preferred over the character LCDs for applications where both character and graphical representation are required.

# Example Applications for Graphic LCDs



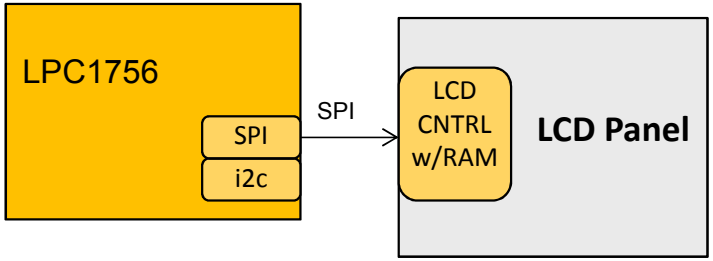


- Embedded applications for vibrant displays.
  - Home Automation and Security
    - Thermostats, security panel, intercom
  - Secure Transactions
    - POS Systems, Access Control, Ticketing
  - White Goods
    - High end Display and Human Interface
  - Industrial Human Machine Interface/ Programmable Logic Controls
    - RPM monitor, temp monitor, alarms
  - Medical Systems
    - Portable meters, large monitoring equipment
- Typical resolutions from CGA (320x200) to XGA (1024x768) and <15fps

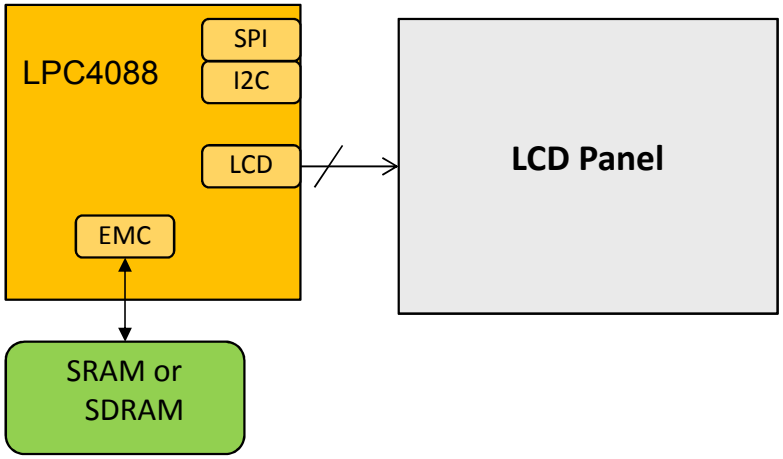
# MCU with external or internal LCD controller

Example interfaces MCU to LCD

– Serial interfaces to reduce pin count to the LCD
  - Lower resolution due to limited SPI bandwidth

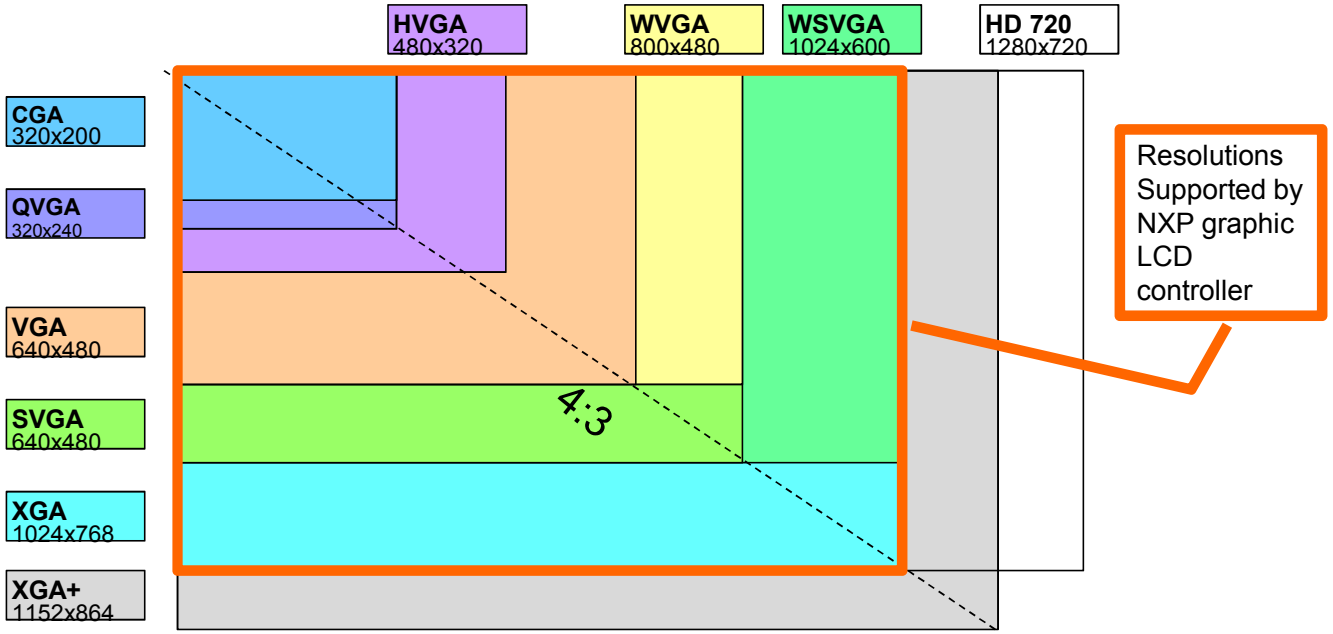



– MCU with parallel LCD controller onchip
  - Can support mid-range resolutions.



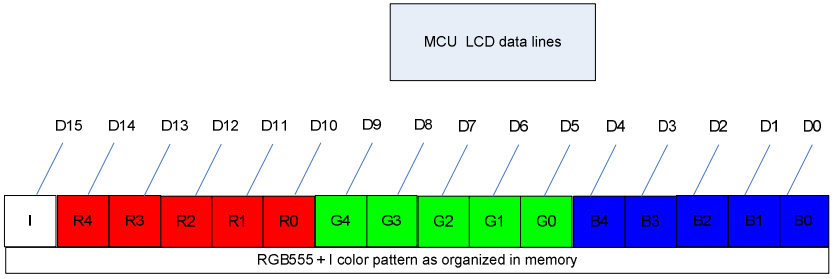
# LCD CONTROLLER

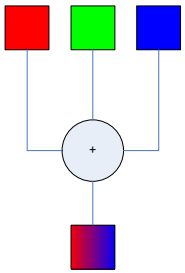
# Choosing an LCD: Resolution:




## Choosing an LCD: Color Depth

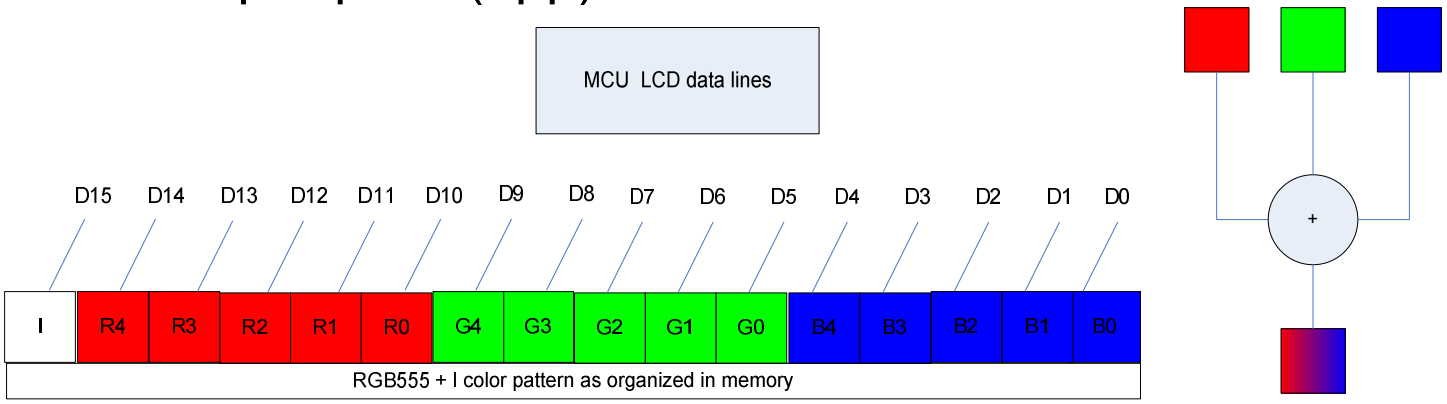
- Color depth or bits per pixel (bpp)



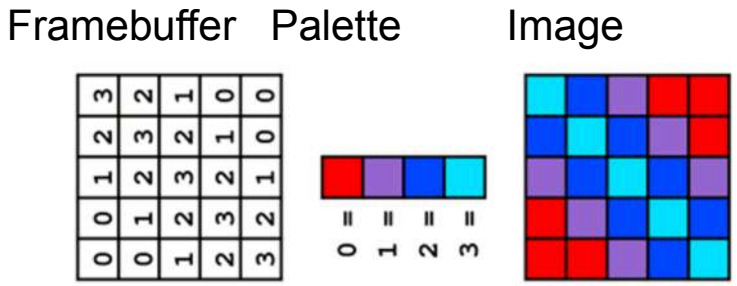
# Resolution and Color Depth

- Resolution is not measured in inches!
  - QVGA 320 X 240
  - VGA 640 x 480
  - SVGA 800 X 600
  - Landscape or portrait orientation
- Color depth or bits per pixel (bpp)



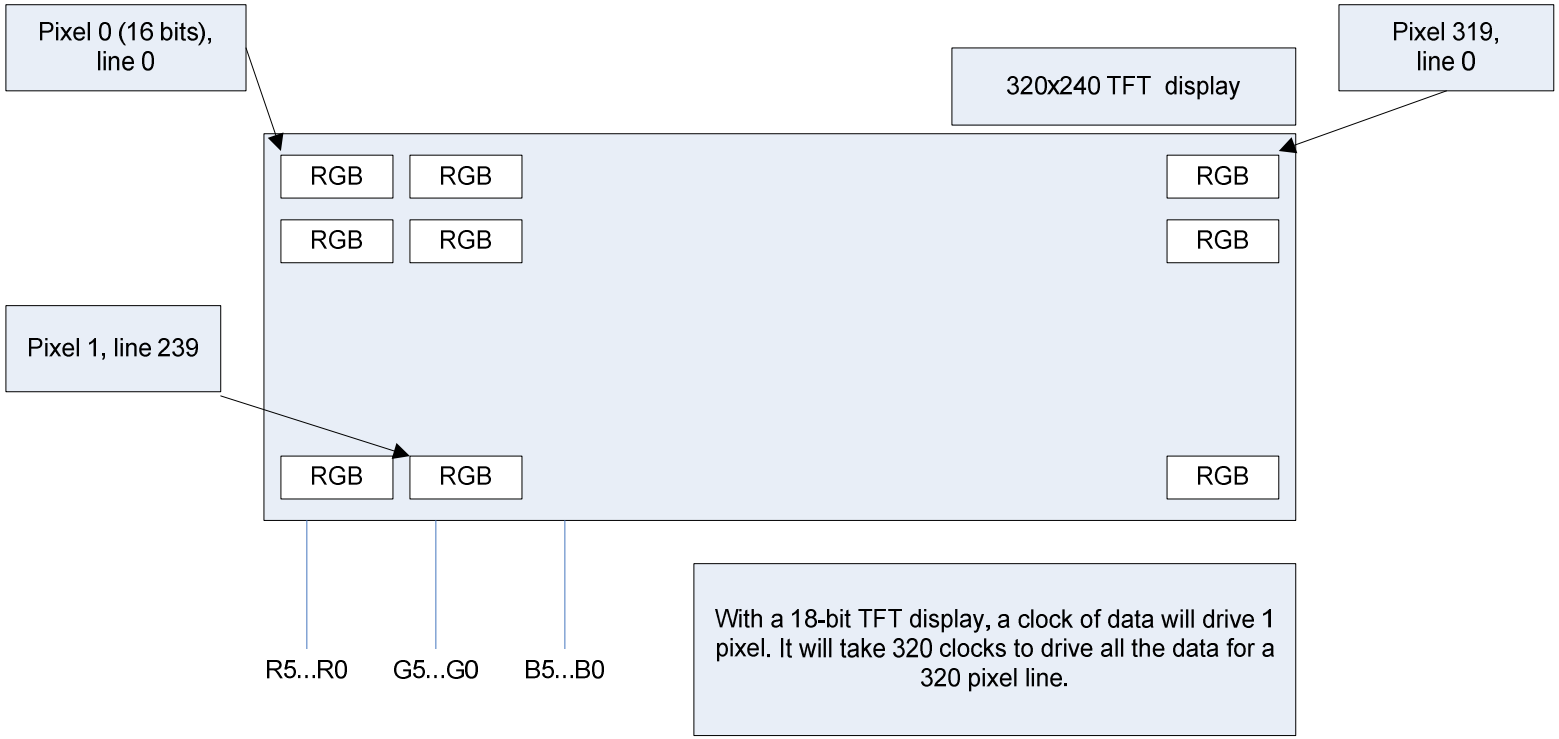
# Palette Based Frame Buffer

- The frame buffer will contain an index value for each pixel
- Palette RAM is pre-filled with 16-bit color value for each index

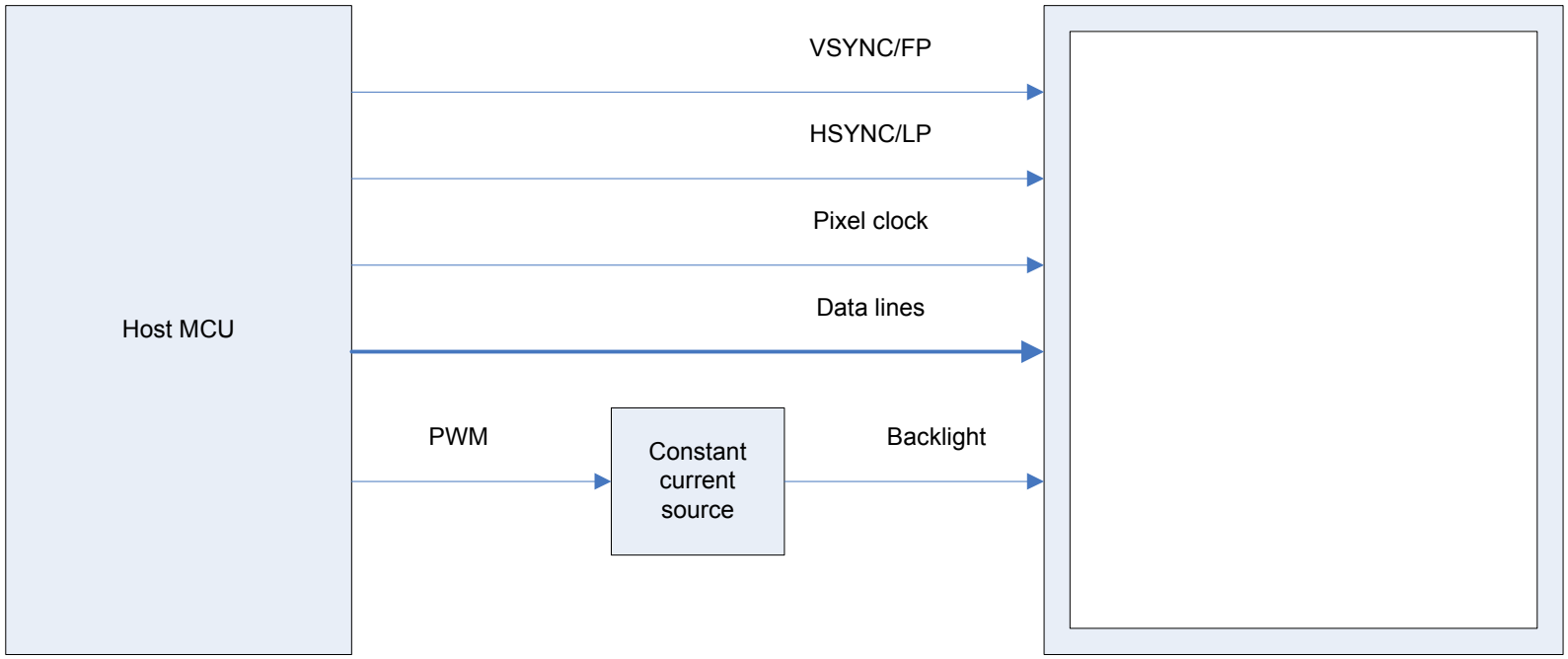



- NXP microcontrollers have 256 entries to support
  - 1, 2, 4, or 8 bpp palletized color displays for color STN and TFT
  - 1, 2, or 4 bits-per-pixel (bpp) palletized displays for mono STN

# LCD Clocked TFT



# Driving a clocked LCD bus



# LCD Signals

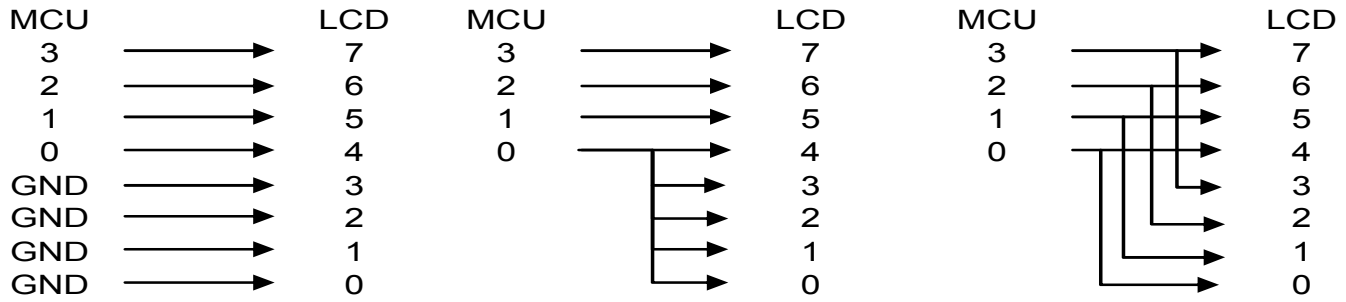
- The largest configuration for the LCD controller uses 31 pins. There are many variants using as few as 10 pins for a monochrome STN panel.

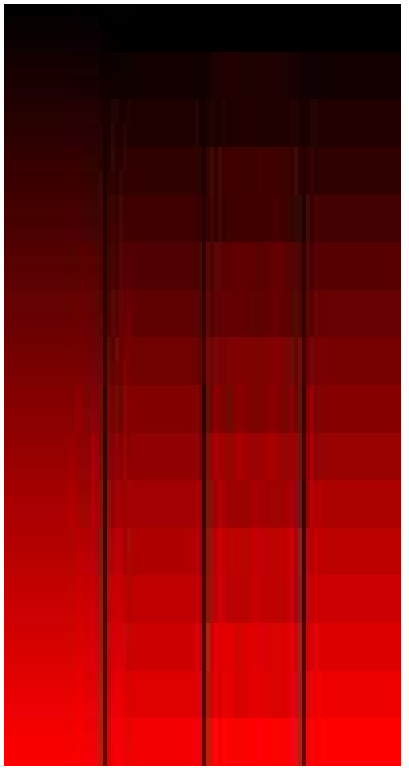
| Pin name | Type | Function |
|---|---|---|
| LCD_PWR | output | LCD panel power enable. |
| LCD_DCLK | output | LCD panel clock. |
| LCD_ENAB_M | output | STN AC bias drive or TFT data enable output. |
| LCD_FP | output | Frame pulse (STN). Vertical synchronization pulse (TFT) |
| LCD_LE | output | Line end signal |
| LCD_LP | output | Line synchronization pulse (STN). Horizontal synchronization pulse (TFT) |
| LCD_VD[23:0] | output | LCD panel data. Bits used depend on the panel configuration. |
| LCD_CLKIN | input | Optional clock input. |

# LCD TFT Signals

| Pin name | 12-bit, 4:4:4 mode (18 pins) | 16-bit, 5:6:5 mode (22 pins) | 16-bit, 1:5:5:5 mode (24 pins) | 24-bit (30 pins) |
|---|---|---|---|---|
| LCD_PWR | Y | Y | Y | Y |
| LCD_DCLK | Y | Y | Y | Y |
| LCD_ENAB_M | Y | Y | Y | Y |
| LCD_FP | Y | Y | Y | Y |
| LCD_LE | Y | Y | Y | Y |
| LCD_LP | Y | Y | Y | Y |
| LCD_VD[1:0] | - | - | - | RED[1:0] |
| LCD_VD[2] | - | - | Intensity | RED[2] |
| LCD_VD[3] | - | RED[0] | RED[0] | RED[3] |
| LCD_VD[7:4] | RED[3:0] | RED[4:1] | RED[4:1] | RED[7:4] |
| LCD_VD[9:8] | - | - | - | GREEN[1:0] |
| LCD_VD[10] | - | GREEN[0] | Intensity | GREEN[2] |
| LCD_VD[11] | - | GREEN[1] | GREEN[0] | GREEN[3] |
| LCD_VD[15:12] | GREEN[3:0] | GREEN[5:2] | GREEN[4:1] | GREEN[7:4] |
| LCD_VD[17:16] | - | - | - | BLUE[1:0] |
| LCD_VD[18] | - | - | Intensity | BLUE[2] |
| LCD_VD[19] | - | BLUE[0] | BLUE[0] | BLUE[3] |
| LCD_VD[23:20] | BLUE[3:0] | BLUE[4:1] | BLUE[4:1] | BLUE[7:4] |

# MCU-to-LCD Interface: Minimizing Artifacts 4:4:4 to 8:8:8



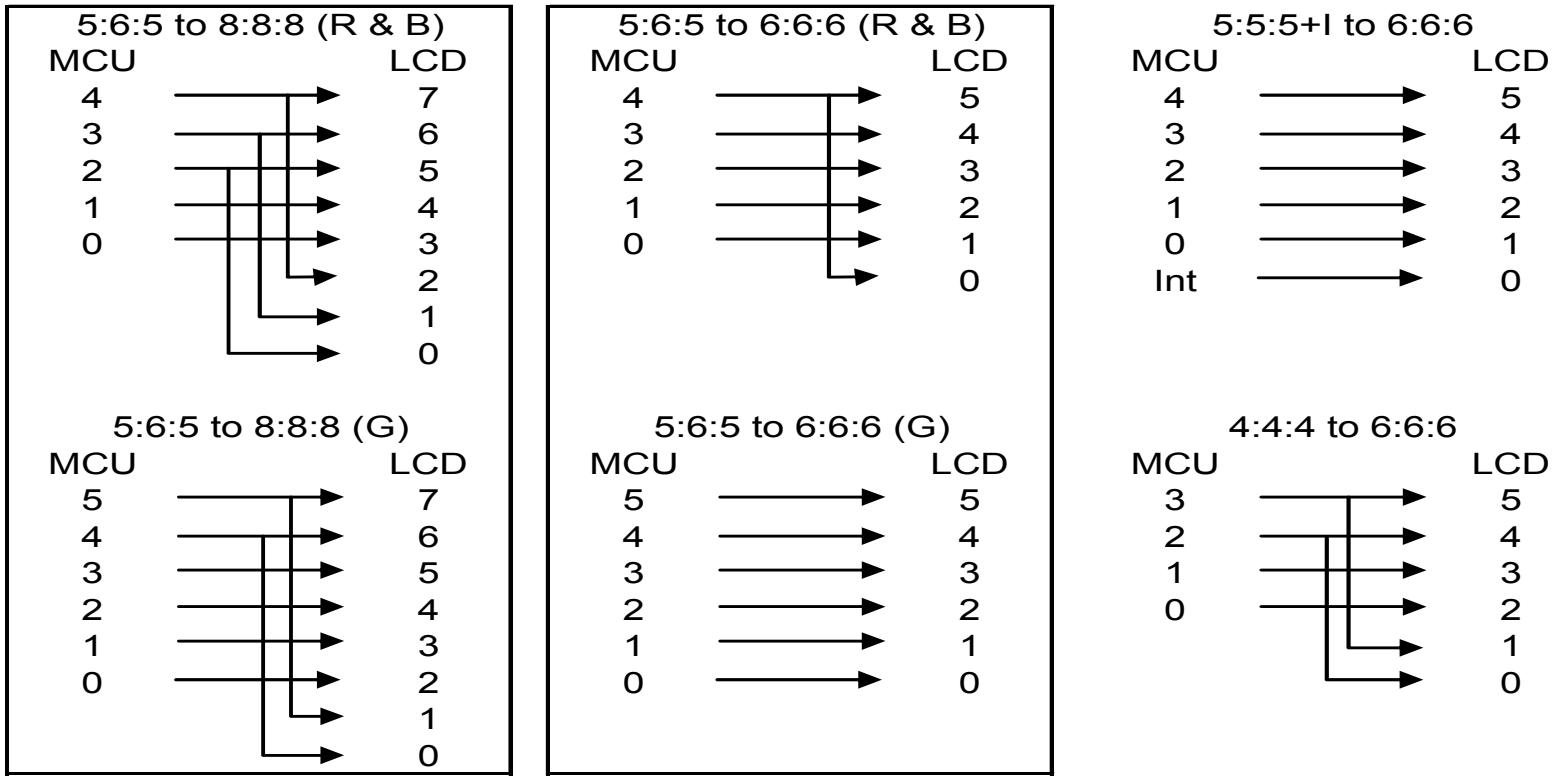
| MCU LCD Data | |
|---|---|
| Dec | Binary |
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 12 | 1100 |
| 13 | 1101 |
| 14 | 1110 |
| 15 | 1111 |

| LCD Receives | |
|---|---|
| Dec | Step |
| 0 | |
| 16 | 16 |
| 32 | 16 |
| 48 | 16 |
| 192 | 16 |
| 208 | 16 |
| 224 | 16 |
| **240** | 16 |
| Avg | 16 |
| Avg Dev | 0 |

| LCD Receives | |
|---|---|
| Dec | Step |
| 0 | |
| 31 | **31** |
| 32 | **1** |
| 63 | **31** |
| 192 | **1** |
| 223 | **31** |
| 224 | **1** |
| 255 | **31** |
| Avg | 17 |
| Avg Dev | 14.9333 |

| LCD Receives | |
|---|---|
| Dec | Step |
| 0 | |
| 17 | 17 |
| 34 | 17 |
| 51 | 17 |
| 204 | 17 |
| 221 | 17 |
| 238 | 17 |
| 255 | 17 |
| Avg | 17 |
| Avg Dev | 0 |

# MCU-to-LCD Data Interface: Other Combinations



# What is a Framebuffer (FB)?

- **Contiguous** memory buffer containing a complete frame of data
- Consists of color values for every pixel
- Color values are commonly represented as
  - 1 bit (1 bpp): Monochrome
  - 2 bit (2 bpp): Palette based (4 colors)
  - 4 bit (4 bpp): Palette (16 colors, controller has a palette look-up table)
  - 8 bit (8 bpp): Palette (256 colors, controller has a palette look-up table)
  - 16 bit (16 bpp): High color format (5:5:5 - 32,768 colors; 5:6:5 - 65,536 colors)
  - 24 bit (24 bpp): True color format (16,777,216 colors)
- LCD controller supports FB address change on the fly
  - New FB will take effect only after current LCD refresh cycle.

# Resolution x Color Depth = Memory Size

- Resolution x Color Depth = total bits needed (divide x8 for bytes)
- Framebuffer = memory buffer containing complete frame (bitmap) of data

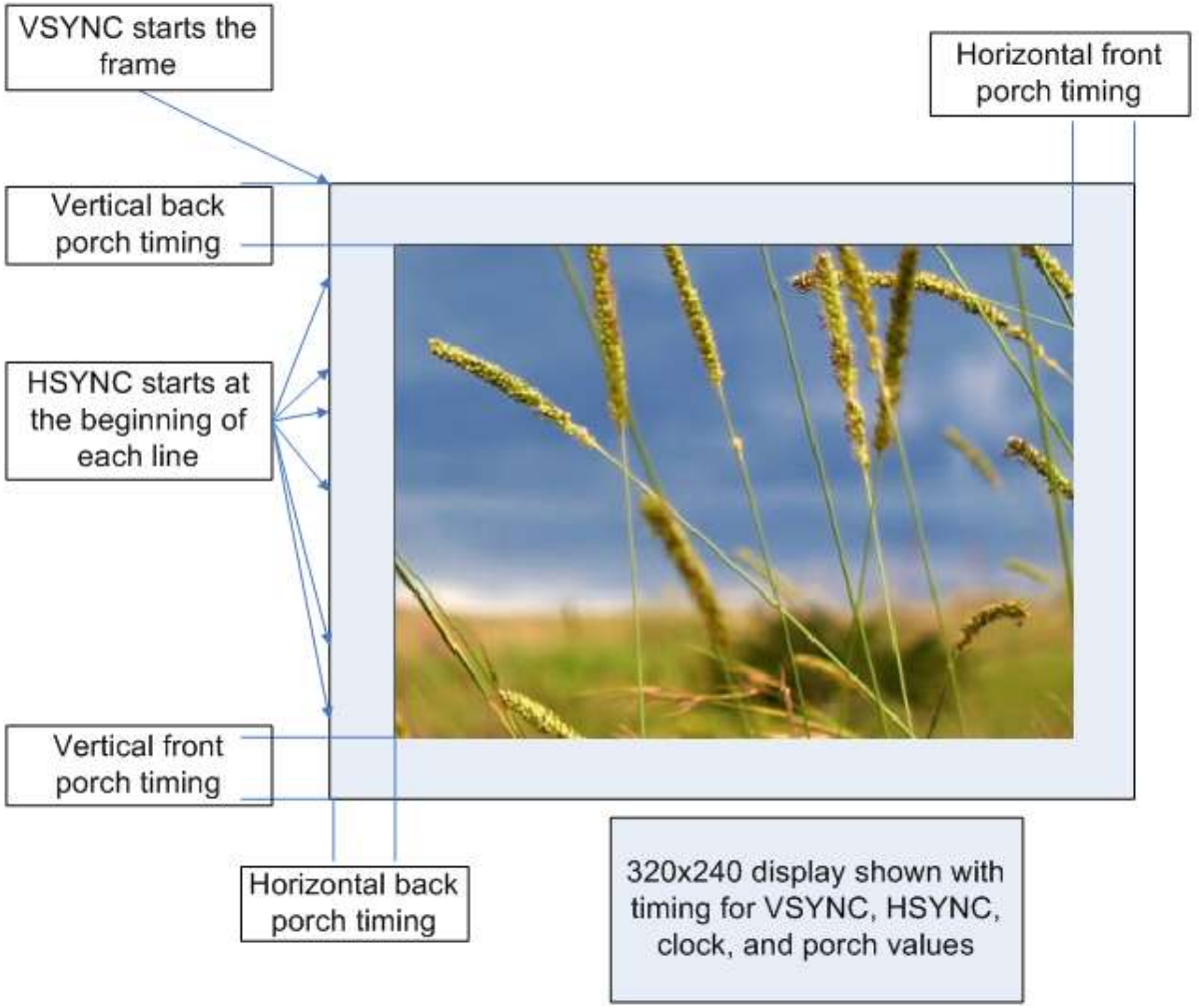
| Resolution | | 1 bits/ pixel | 2 bits/ pixel | 4 bits/ pixel | 8 bits/ pixel | 16 bits/ pixel | 24 bits/ pixel |
|---|---|---|---|---|---|---|---|
| XGA | 1024x768 | 98,304 | 196,608 | 393,216 | 786,432 | 1,572,864 | 2,359,296 |
| WVGA | 800x480 | 48,000 | 96,000 | 192,000 | 384,000 | 768,000 | 1,152,000 |
| VGA | 640x480 | 38,400 | 76,800 | 153,600 | 307,200 | 614,400 | 921,600 |
| WQVGA | 480x272 | 16,320 | 32,640 | 65,280 | 130,560 | 261,120 | 391,680 |
| QVGA | 320x240 | 9,600 | 19,200 | 38,400 | 76,800 | 153,600 | 230,400 |
| CGA | 320x200 | 8,000 | 16,000 | 32,000 | 64,000 | 128,000 | 192,000 |

Example: 480 x 272 x 16bpp x 8bits/byte = 261,120 bytes needed



18

# Driving the LCD – various timings



# Refresh Rate

- REFRESH_RATE (Hz) =
  pixel_clock_rate / [(vertical_resolution + vertical_front_porch + vertical_back_porch) * (pixel_clocks_per_data_line + horizontal_front_porch + horizontal_back_porch))]

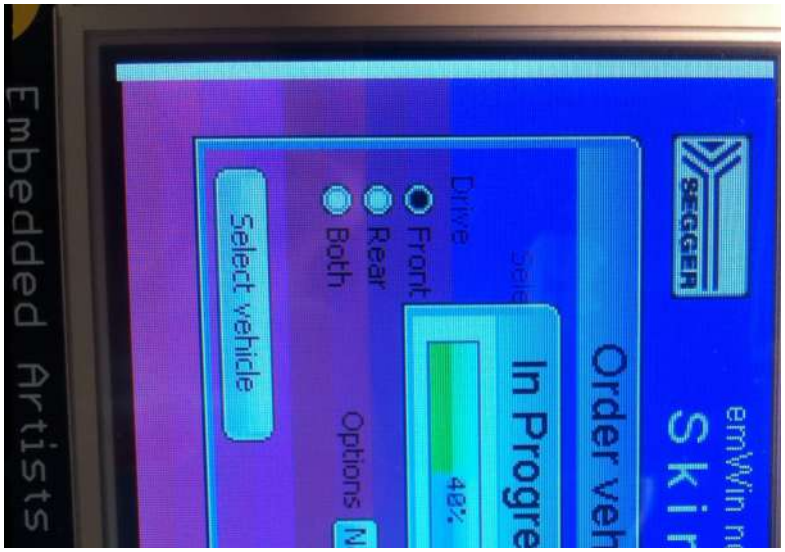
- Example :
  - 6.5MHz pixel clock
  - vertical resolution=240 lines,
  - vertical front porch=5 lines,
  - vertical back porch=1 line,
  - pixel clocks per data line = 320 pixels,
  - horizontal front porch=20 clocks,
  - horizontal back porch=10 clocks

  - REFRESH_RATE = 6,500,000 / [(240 + 5 + 1) * (320 + 20 + 10)] = 75.5Hz

# Example: Truly 240 x 320 TFT RGB666

| Characteristics | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| DOTCLK Frequency | $f_{DOTCLK}$ | - | 5.5 | 8.22 | MHz |
| DOTCLK Period | $t_{DOTCLK}$ | 122 | 182 | - | nSec |
| Horizontal Frequency (Line) | $f_H$ | - | 19.6 | 29.3 | kHz |
| Vertical Frequency (Refresh) | $f_V$ | - | 60 | 90 | Hz |
| Horizontal Back Porch | $t_{HBP}$ | - | 30 | - | $t_{DOTCLK}$ |
| Horizontal Front Porch | $t_{HFP}$ | - | 10 | - | $t_{DOTCLK}$ |
| Horizontal Data Start Point | $t_{HBP}$ | - | 30 | - | $t_{DOTCLK}$ |
| Horizontal Blanking Period | $t_{HBP} + t_{HFP}$ | - | 40 | - | $t_{DOTCLK}$ |
| Horizontal Display Area | HDISP | - | 240 | - | $t_{DOTCLK}$ |
| Horizontal Cycle | $H_{cycle}$ | - | 280 | - | $t_{DOTCLK}$ |
| Vertical Back Porch | $t_{VBP}$ | - | 4 | - | Line |
| Vertical Front Porch | $t_{VFP}$ | - | 2 | - | Line |
| Vertical Data Start Point | $t_{VBP}$ | - | 4 | - | Line |
| Vertical Blanking Period | $t_{VBP} + t_{VFP}$ | - | 6 | - | Line |
| Vertical Display Area | VDISP | - | 320 | - | Line |
| Vertical Cycle | $V_{cycle}$ | - | 326 | - | Line |

# Snapshot of incorrect LCD settings

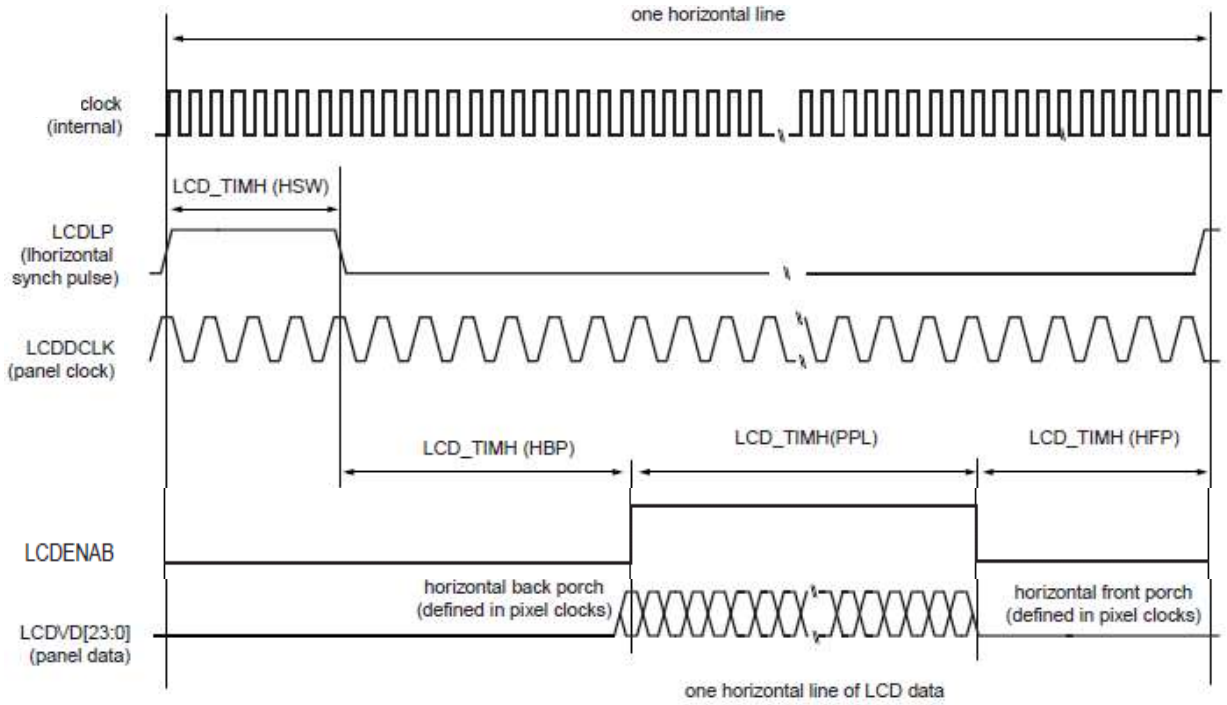



```
static const LCD_PARAM_T truly_g240320ltsw =
{
   30, //28,       /* Horizontal back porch */
   10,            /* Horizontal front porch */
    2,            /* HSYNC pulse width */
  240,            /* Pixels per line */
    8,            /* Vertical back porch */
    2,            /* Vertical front porch */
    2,            /* VSYNC pulse width */
  320,            /* Lines per panel */
```

| | | | |
|---|---|---|---|
| Vertical Back Porch | $t_{VBP}$ | - | 4 |

# LCD – TFT Horizontal Timings

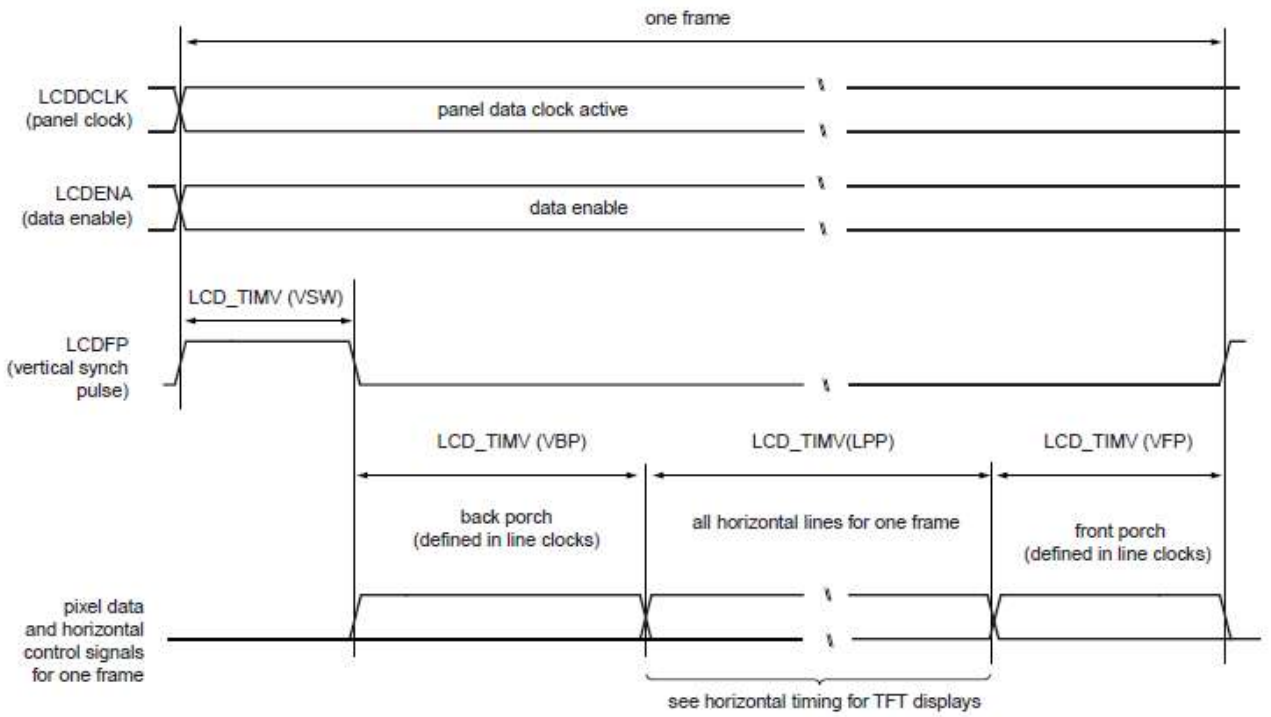
- HSync pulse starts a new line
- Horizontal back porch follows Hsync
- Pixel data follows Horizontal back porch
  - LCDVD[23:0] only valid during this stage
- Final stage is the horizontal front porch.




(1) The active data lines will vary with the type of TFT panel.
(2) The LCD panel clock is selected and scaled by the LCD controller and used to produce LCDCLK.
(3) The duration of the LCD_LP is controlled by the HSW field in the LCD_TIMH register.
(4) The polarity of the LCD_LP signal is determined by the IHS bit in the LCD_POL register.

# LCD – TFT Vertical Timings

- VSync pulse starts a new frame
- Vertical back porch follows Hsync
- Horizontal lines follow vertical back porch
  - LCDVD[23:0] only valid during this stage
- Final stage is the vertical front porch.




(1) Polarities may vary for some displays.

# LCD Tearing

- Tearing:


- Result of LCD DMA unable to service the LCD FIFO in time
- Use the FIFO Underflow to monitor for this
- Workarounds
  - Change AHB priority – next slide
  - Slow down frame refresh rate, pixel clock if possible
  - Use 32-bit wide external memories
  - Increase the SDRAM clock speed, use faster SRAM
  - Profile code and move time critical code to internal SRAM

# Bus Bandwidth Calculator

| LPC178x Bus Bandwidth on Various LCD Resolutions and Color Depths at Various Refresh Rate | |
|---|---|
| Bus Clock (MHz): | 80 |
| Static External Memory Configuration - | |
| Bus Width: | 32 |
| Read Delay, WAITRD: | 1 |
| Dynamic External Memory Configuration - | |
| Bus Width: | 32 |
| Precharge Command Period, $t_{RP}$: | 2 |
| RAS Latency (Active to Read/Write Delay), RAS ($t_{RCD}$): | 2 |
| CAS Latency, CAS: | 2 |
| LCD Resolution - | |
| Horizontal (Pixels): | 640 |
| Vertical (Pixels): | 480 |
| Refresh Rate - | |
| Refresh Rate (Hz): | 60 |
| | |
| LCD Color Depths - | |
| Color Depth (bpp): | 16 |
| | |
| Frame Buffer - | |
| Frame Buffer (KB): | 600 |
| LCD Data Rate - | |
| Data Rate (Mpixels/s): | 18.432 |
| Data Rate (MWords/s): | 9.216 |
| Data Rate (Mbursts/s): | 2.304 |
| Static External Memory Burst - | |
| Burst (clocks): | 13 |
| Dynamic External Memory Burst - | |
| Burst (clocks): | 15 |
| Bus Bandwidth Needed by LCD: | |
| Static External Memory (%): | 37.44 |
| Dynamic External Memory (%): | 43.2 |

- http://www.lpcware.com/content/nxpfile/lcd-bus-bandwidth-calculator-lpc177x8x

# LPC5460x and LPC4088 LCD AHB Priority

- AHB Matrix Arbitration register (Matrix_Arb - 0x400F C188)
- The values used for the various priorities are 3 = highest, 0 = lowest

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 1:0 | PRI_ICODE | I-Code bus priority. Should be lower than PRI_DCODE for proper operation. | 0x1 |
| 3:2 | PRI_DCODE | D-Code bus priority. | 0x3 |
| 5:4 | PRI_SYS | System bus priority. | 0 |
| 7:6 | PRI_GPDMA | General Purpose DMA controller priority. | 0 |
| 9:8 | PRI_ETH | Ethernet DMA priority. | 0 |
| 11:10 | PRI_LCD | LCD DMA priority. | 0 |
| 13:12 | PRI_USB | USB DMA priority. | 0 |
| 15:14 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 16 | ROM_LAT | ROM latency select. Should always be 0. | 0 |
| 31:17 | - | Reserved. Read value is undefined, only zero should be written. | NA |

- To give priority to the LCD DMA use the value 0x0000 0C09

# Simple usage: Display images on LCD

- CPU can draw pixels to FB directly, or draw in background memory then switch FB address.
- For displaying full screen images, one image can be one FB, FB can locate in SRAM, SDRAM or QSPI flash.
- Switching FB can change displayed image.
  - No animation effect.
- Dual-framebuffering for flicker-free animation
  - 1 foreground FB + 1 background FB
  - Foreground FB is being refreshed by LCD controller
  - Background FB is being rendered by graphics software, which can do animation effects.
  - Program LCD controller to use background FB after rendered.
    - LCD controller only apply new FB after current LCD refresh cycle, and previous foreground FB becomes background FB.

# GRAPHICS SOFTWARE PACKAGE

## Challenges of graphics UI

- More modern and Attractive Interfaces are expected
- Uis with iPhone / Smart Phones' look and feel
- Graphics and color schemes
- White goods with FULL COLOR and INTERACTIVE displays
- Marketing and Management expect Better Looking but don't provide more time for development.
- Graphics require lots of cumbersome tasks.
- Embedded GUI libraries takes responsibilities of most fundamental GUI development.



# Basic Graphic's Library - SWIM

- A free 'basic' graphics library
  - **S**imple **W**indows **I**nterface **M**anager
  - Basic fonts ASCII characters, 6x7, 6x13, 8x8
  - Draw boxes, lines, set colors
  - Display and scale bitmap images


- AN10815
  - API documentation for SWIM Library
  - Quick start guide document
  - IAR, Keil, Rowley, GNU projects
  - http://ics.nxp.com/support/documents/microcontrollers/zip/an10815.zip

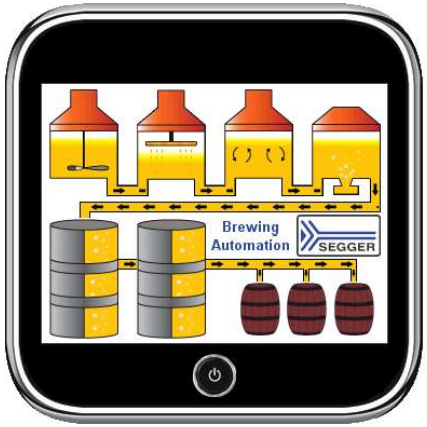
# emWin overview

- graphical user interface solution for embedded system by Segger.
- processor- and display controller-independent for any application that operates with a graphical display.
- compatible with single-task and multitask environments, with a proprietary operating system or with any commercial RTOS.

# Typical emWin HMI's








## What NXP customers used emWin for…

- Paper money counter (LPC4300 + 3.1” LCD)
- ATM (LPC1800 + 14”, 1024x768 LCD)
- Industrial touch panel (LPC1788 + 10.1”, 640x480 LCD)
- Washing machine (LPC3000)
- Elevator control with LCD (LPC1788)
- High accuracy scales with LCD (LPC1788)
- Security Panel (LPC2132)

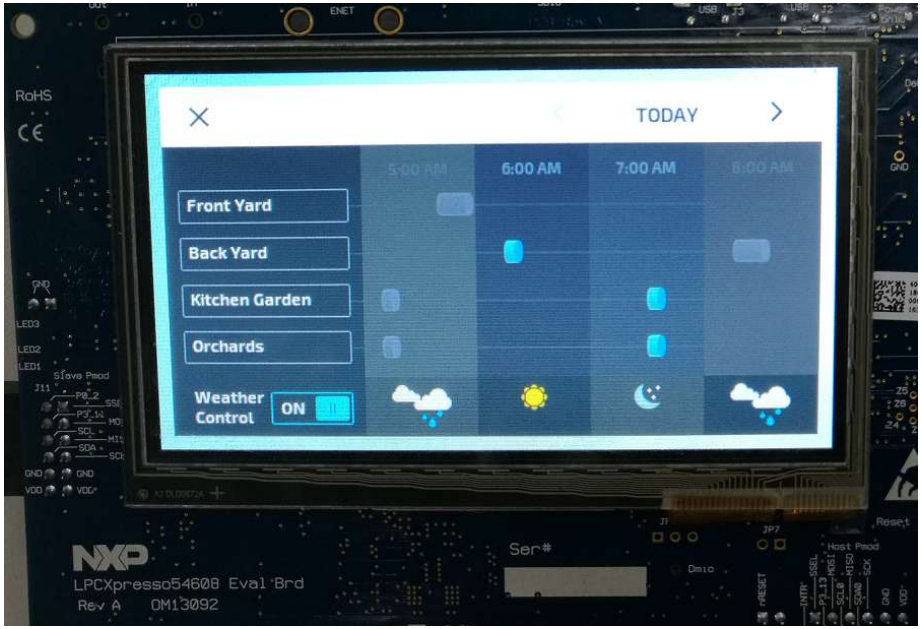
# License terms

- Free to use with any current NXP ARM Cortex M0, M3 or M4 MCU
- No royalties or licensing fees when used with NXP MCUs
- No source (provided as a pre-compiled library)
- Source code available under license agreement from SEGGER

- The full license agreement is included in every installer, but there are essentially no limitations on the use of emWin with NXP MCUs
- The only restriction is that the emWin library is provided solely in object code ("library") format. Customers may use these libraries on NXP MCUs free of charge (without royalty or additional license fees), for both personal and commercial development
- As part of the licensing agreement with Segger, the source code for emWin can not be provided, but if you require the original source code for your own project, Segger offers special pricing for NXP customer's when upgrading from the NXP emWin library

# TouchGFX – A newer and rich UI graphics library

- TouchGFX unlocks the graphical user interface (GUI) performance of low-resource hardware. It lets users create sophisticated GUIs that fully live up to today's smart phone standards at a fraction of the cost. By using TouchGFX, embedded product gets outstanding graphics and smooth animations with minimal resource and power consumption.
- TouchGFX provides a very powerful PC designer studio TouchGFX Designer, it is an easy-to-use GUI builder that supports the development of embedded GUIs based on TouchGFX



SECURE CONNECTIONS
FOR A SMARTER WORLD