Freescale Semiconductor
Application Note

Document Number: AN3977
Rev. 0, 01/2010

Different Display Configurations on the i.MX25 WinCE PDK

by Multimedia Applications Division
Freescale Semiconductor, Inc.
Austin, TX

This application note provides the necessary information and the procedure to add or adapt a new Liquid Crystal Display (LCD) panel to the BSP distribution for the i.MX25 WinCE PDK. The document describes, in brief, the principles of an LCD and the generalities of the display controller module. It describes the development process to adapt a new LCD panel to the Board Support Package (BSP), considering that the framework driver structure is already provided by the operating system. This application notes assumes that the reader is familiar with the Microsoft Platform Builder packages and the WINCE device driver concepts.

Contents

1 Introduction

As a multimedia processor, the i.MX25 supports several types of displays. The Liquid Crystal Display Controller (LCDC) handles the display devices.

The process of selecting an LCD for a mobile device involves several conflicts with varying requirements. Some of the conflicts are as follows:

- Large amount of data, implying high rate of data transfer and processing, requiring significant resources
- Flexibility, to support a variety of use cases
- Size, cost and power consumption

Freescale provides reference designs for the i.MX family where the functionality of LCD is demonstrated. However, according to the requirements, developers find many reasons to replace the display in their products. Features such as screen size, resolution, weight, power consumption, and price are important in a commercial multimedia product. Another important fact about LCD panels is that many displays become obsolete quickly and it is hard to find the same LCD panel included in the reference design.

This application note is intended only for dumb displays and mainly those displays which do not have the Sharp synchronous interface. However, some information in the application notes is also useful for smart displays.

NOTE

Do not confuse Sharp panels with Sharp interface as there are a number of Sharp panels which do not use the Sharp interface.

2 LCD Principles

The following sections discuss the principles of LCD.

2.1 LCD Basics

Liquid Crystal Display (LCD) is an electronic device which consists of an array of pixels of color or monochrome units. Every pixel in the array consists of a special material that allows them to change the characteristics of the light that passes through them. These devices do not emit light and thus, another element named backlight is shipped with the panel to create a fully functional device.

2.1.1 Resolution

In this application note, the term resolution is used to refer the number of pixels in an LCD array. It has two dimensions: horizontal and vertical.

Table 1 lists the most common video resolution standards available in the market.

Table 1. Common Video Resolution Standards

Video Name	Description	Width	Height	Aspect Ratio
CGA	Color Graphics Adapter	320	200	8:5
QVGA	Quarter VGA	320	240	4:3
VGA	Video Graphics Array	640	480	4:3
NTSC	National Television System Committee	720	480	3:2
PAL	Phase Alternating Line (TV)	720	576	5:4
WVGA	Wide VGA	800	480	5:3
SVGA	Super VGA	800	600	4:3

The maximum resolution that i.MX25 supports is SVGA and hence resolutions greater than SVGA are not included in the table above.

All resolutions mentioned in Table 1 refer to a landscape orientation, which means that there are more pixels in the horizontal axis than in the vertical axis. However, there are also portrait LCD panels available in the market with the same standard resolution but with inverted horizontal and vertical dimensions. Portrait LCD panel has more vertical pixels than horizontal pixels.

Figure 1 shows the portrait and landscape orientation of an LCD panel.

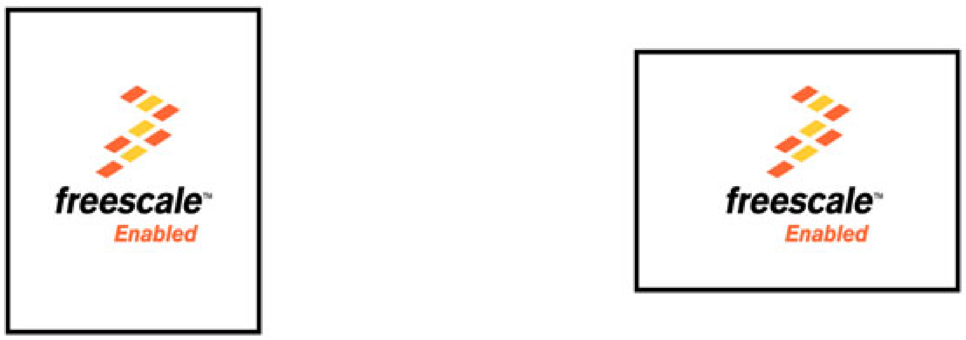


Figure 1. Portrait and Landscape Orientation of an LCD Panel

It is important to select the proper orientation of an LCD panel, because both electronic and optical features are optimized for applications that use the native orientation of the panel. Besides the optical characteristics, dumb displays include an embedded LCD controller and which draws the pixels from left to right and also from top to bottom. However, to show images or videos on the LCD panel using a non-native orientation, the display content is pre-processed. The image is stored in a buffer in a way (order) the LCD controller expects the pixel information to be sent to it. This operation is called rotation but the i.MX25 does not include hardware to perform this operation. This operation is called rotation but the i.MX25 does not include hardware to perform this operation. To avoid additional image processing, select an LCD panel that mostly uses native orientation.

Figure 2 shows both portrait and landscape LCD panels displaying images in the non-native orientation.

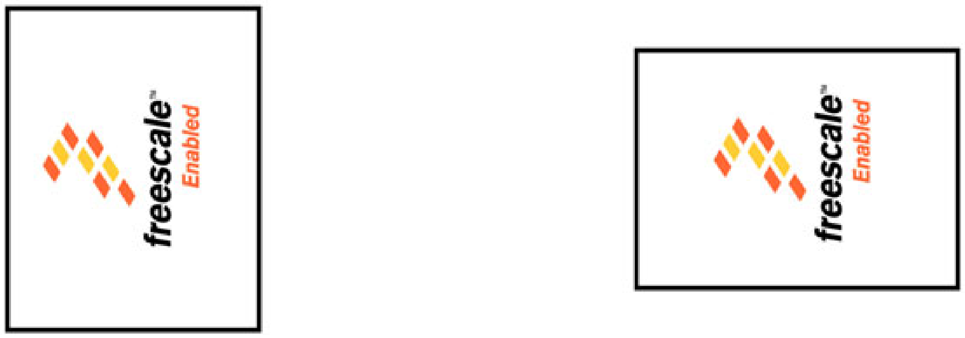


Figure 2. Non-native Portrait and Landscape Orientation of an LCD Panel

Rotation is not just limited to 90°, it can be either 90°, 180° or 270°. Also, note that every frame has to be rotated before it is driven to the display.

2.1.2 Size

The size of an LCD panel is measured diagonally in inches, from corner to corner. It is common to assume the size of a VGA (640x480) panel to be larger than a QVGA (320x240) panel since VGA has a greater number of pixels. But, this is not true always. LCD manufacturing processes allow the size and resolution to be independent variables; it is difficult to determine the size of a panel from its resolution alone. Screens that are larger in size tend to consume more power than smaller ones and also impact the size and weight of the final product. On the other hand, higher resolutions on smaller LCD panels can complicate the visibility of on-screen objects for the final user. Based on the information available in the datasheet, it is difficult to determine if a particular LCD panel fits the application.

2.1.3 Color Spaces

A color space is a way to represent colors. There are two main color spaces, RGB (that is, RGB444, RGB565, RGB666, RGB888, and RGBA8888) and YUV (that is, YUV 4:4:4, YUV 4:2:2, and YUV

4:2:0). Since the i.MX25 has a 24-bit data width, it supports up to RGB888. Thin Film Transistor (TFT) panels can receive data only by using the RGB interface.

2.2 LCD Types

The LCD panels are categorized as synchronous and asynchronous panels. The following sections discuss the types of LCD panels.

2.2.1 Synchronous Panel (Dumb Display)

Dumb displays or synchronous displays are panels which require the microprocessor to send all pixels in the image periodically and continuously. In the these panels, screen refresh is performed on the fly and each pixel is drawn as soon as the panel gets the pixel data. After drawing a pixel, the LCD waits for the next pixel data. This process is repeated until the complete array is drawn. In general smart displays are more expensive than dumb displays, and this is the reason why synchronous panels are more commonly used in the final product. This application note focuses on TFT panels which belong to a special group of synchronous panels.

2.2.2 Asynchronous Panel (Smart Display)

The advantage of smart displays is that the i.MX25 only has to send display data when the image has changed, and most of the times send only the portion that has changed. Images can be sent at any time, the screen refresh is handled by the Smart Liquid Crystal Display Controller (SLCDC).

3 Synchronous Display Interface

The i.MX25 LCDC can be configured to handle black-and-white, grey-scale, passive-matrix color (passive color or CSTN), and active-matrix color (active color or TFT) LCD panels. This document focuses only on the synchronous TFT color interface.

For LCD TFT color panels, the i.MX25 provides a 28-line interface which is described in Table 2.

Table 2. LCDC Display Interface Signals

Signal	LCDC Signal	Description
HSYNC	LP/HSYNC	Horizontal synchronization
VSYNC	FLM/VSYNC	Vertical synchronization
Data enable	ACD/OE	Data enable or Data ready
PIXCLK	LSCLK	Pixel clock
Red Data[7:0]	LD[23:16]	Pixel Red component
Green Data[7:0]	LD[15:8]	Pixel Green component
Blue Data[7:0]	LD[7:0]	Pixel Blue component

The description of the signals referred in Table 2 are as follows:

HSYNC	Horizontal synchronization (HSYNC) signal, also known as FPLINE or LP, when active, indicates the LCD that a line has ended and the valid pixels to follow are part of the next line.
VSYNC	Vertical synchronization (VSYNC) signal, also known as FPFRAME, FLM, SPS or TV, when active, indicates the LCD that the current frame has ended. The LCD display must then restart the line index to zero to draw the next valid data in the first line of the panel.
Data Enable	The data enable (DE) signal, when active, indicates the LCD that the data in the RGB bus is valid and must be latched. While DE is active, every PIXCLK pulse indicates the LCD to draw a pixel using the color described in the RGB bus. The width of this signal must be enough to store all pixels (that is, as long as all pixel clock cycles of a single line).
PIXCLK	The polarity of the pixel clock (PIXCLK) signal indicates when the RGB data is placed on the bus. The following are the two different scenarios. • The high polarity of PIXCLK indicates that RGB data is written on the bus on falling edges and data is latched onto the LCD panel on rising edges. This is valid when data enable (DE) is active. • The low polarity of PIXCLK indicates that RGB data is written on the bus on rising edges and data is latched onto the LCD panel on falling edges. This is valid when data enable (DE) is active.
RGB Data (Display Interface)	The i.MX25 can internally use different types of bits per pixel such as RGB565, RGB666, RGB888 and RGBA8888 and so on. In the same manner, the display interface can be configured to support more than one color depth. The i.MX25 processor can use up to 24 data lines (RGB888) as display interface bus.
Extra Signals	There are also some other signals that are usually included in the panel interface. These signals are not part of the 28-line display interface, but they are required for a module to be fully functional. For example, it is common that some panels need a reset signal, and also initialization commands. These commands are usually sent by a serial interface such as I2C or SPI. Display panels sometimes have backlight unit and touch panels embedded on them, which require additional signals.
SPI Interface	Some LCD displays require an initialization routine through a serial interface, 3-wire, 4-wire, or 5-wire.

3.1 Examples of Synchronous Display Interfaces

The following section describes a few examples of the interface between the i.MX25 PDK and synchronous display panels.

3.1.1 i.MX25 PDK Chunghwa 5.7” VGA LCD Interface

Figure 3 shows the interface between i.MX25 PDK and Chunghwa CLAA057VA01CT VGA panel.

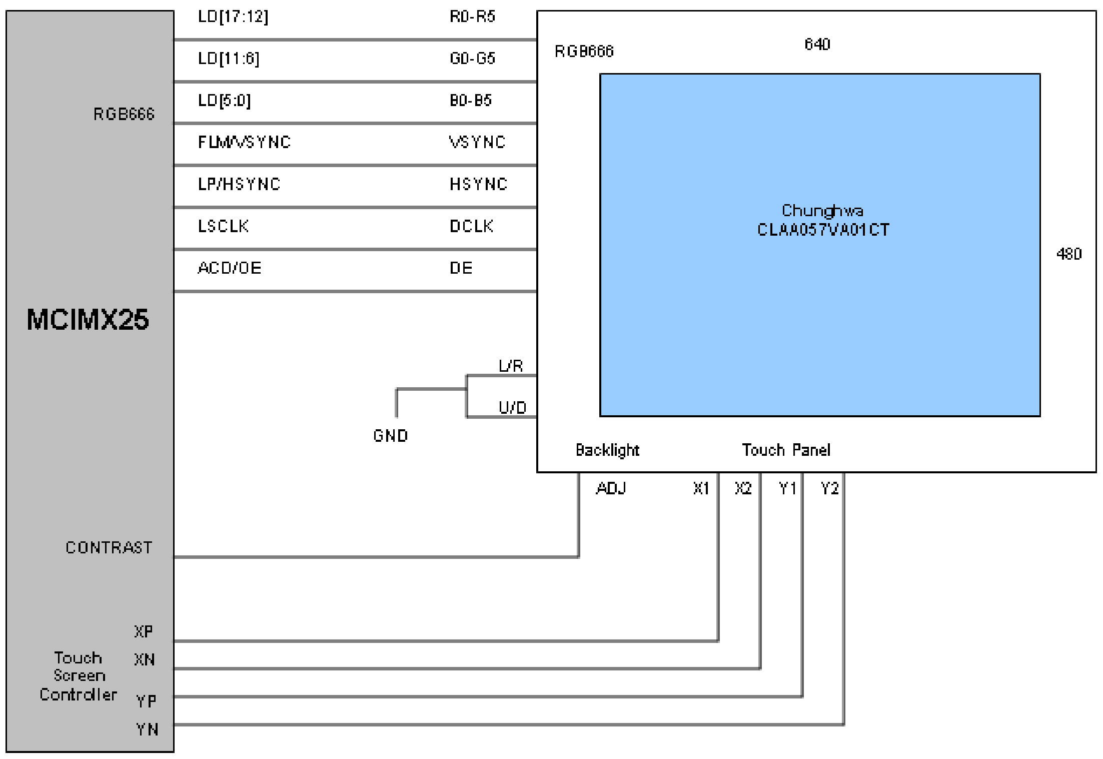


Figure 3. Interface between i.MX25 and Chunghwa CLAA057VA01CT VGA Panel

The LCD panel is shipped with the i.MX25 PDK as shown in Figure 3. The LCD panel requires HSYNC, VSYNC, DE, PIXCLK and part of the RGB data interface (LD[17:0]). But no additional signals such as RESET signal or serial interface initialization routine commands (SPI or I2C) are required. Also, the backlight unit is controlled by using a PWM signal generated by the i.MX25 (Contrast) and finally the touch panel interface is handled by the built-in Touch Screen Controller (TSC).

Every panel has its own interface and requirements, but in general terms the example above illustrates a typical synchronous panel interface. As it is difficult to determine what is typical about LCD panels, consider the scenario shown in Figure 3 as the base for the panel interface. The base interface scenario is useful when there are many panels which do not use the complete interface. For example, some of them require either HSYNC signal or VSYNC signal or neither of them, and only require DRDY, PIXCLK and RGB data. Similarly, many others do not require a RESET or a serial initialization routine to handle display signals. When there is no serial interface; the timing, signals, porches, and polarities are already specified. Also, the panels expect the microprocessors to comply with these waveforms as these panels cannot handle a different configuration.

3.1.2 i.MX25 PDK Chunghwa CLAA070VC01 7” WVGA LCD Interface

Figure 4 shows the interface between i.MX25 PDK and Chunghwa CLAA070VC01 7” WVGA panel.

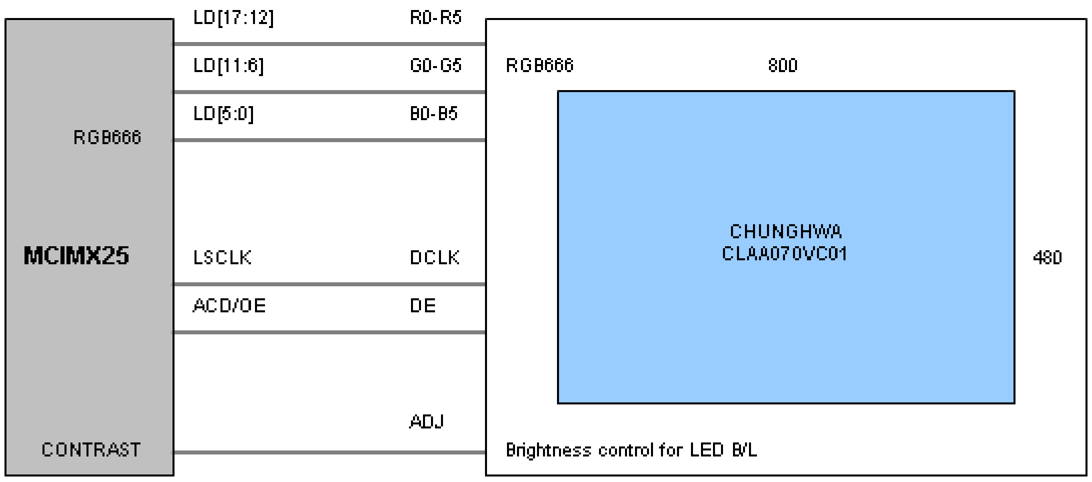


Figure 4. Interface between i.MX25 and Chunghwa CLAA070VC01 WVGA Panel

Figure 4 shows a simple display interface where HSYNC and VSYNC signals are not used. If the HSYNC and VSYNC pins are not used, the pins can be used for other purposes, such as a GPIO configuration. The J15 connector for this LCD is included on the i.MX25 PDK.

Also, SPI interface is not required and the power booster for the backlight unit is included into the module and thus the CONTRAST signal is directly connected to the display connector. This display module does not include a touch panel and it is necessary to add an external touch screen to this LCD panel.

NOTE

The display module does not include a touch panel and it is necessary to add an external touch screen to this LCD panel.

In this example and also in 5.7” VGA LCD interface, mentioned earlier, only 18 RGB data (LD[17:0]) is required. All the remaining RGB data lines (LD[23:18]) can be mapped to different pins and used for other purposes such as, GPIO or any other alternate function which the pin can support or perform.

It is important to remember that, since these LCD modules neither have RESET signal nor a SPI interface, the display cannot be turned off. This feature is important for mobile devices where power consumption is an issue. External circuits are required to control the power consumption of the LCD. In the case of the PDK, the power ON/OFF settings is handled by an external MOSFET circuit driven by an i.MX25 GPIO.

Based on the above observations, the complete LCD circuit for the i.MX25 PDK is as shown in Figure 5.

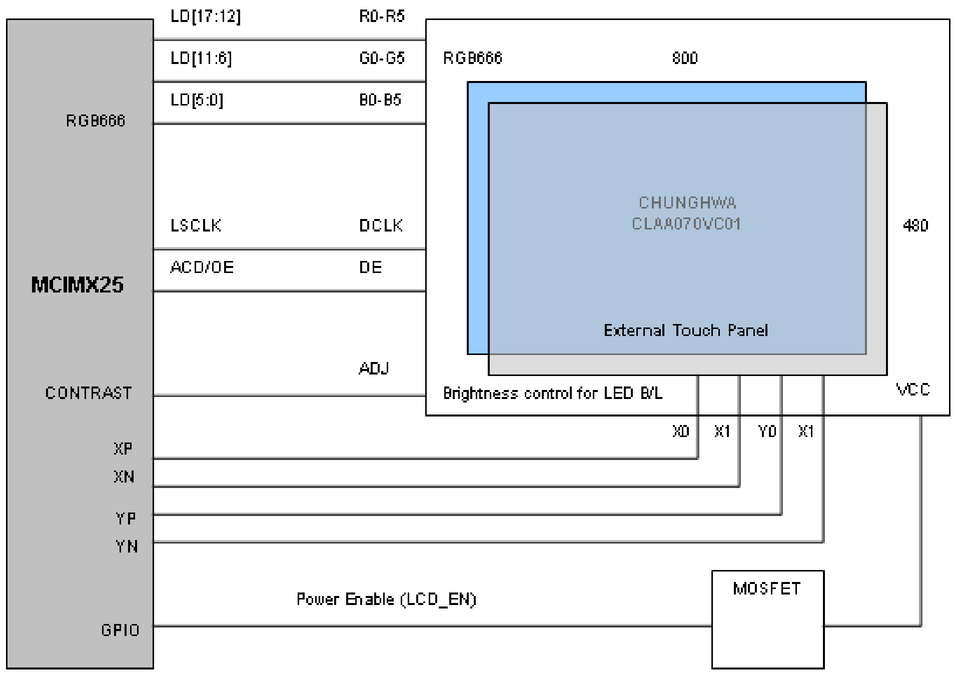


Figure 5. Interface between i.MX25 and Chunghwa CLAA070VC01VWGA Panel + Touch Panel

The examples above help the user to select an LCD display and also determine the advantages and disadvantages of any panel.

3.2 Synchronous Display Timing and Signals

This section focusses on the timing and signal waveforms and how to configure the i.MX25 LCDC with these features. The first step in selecting an LCD module is to refer to its datasheet. The datasheet must contain the pin interface, the initialization routine, the timing charts for the RGB interface, and also the serial interface. Many times a shorter version of the datasheet is available which may not contain all this information. In this case, it is advisable to request for full documentation from the supplier. Many datasheets that are available may be the preliminary version. Though there is not much difference between preliminary and final versions, it is always better to have the final version.

3.2.1 Timing Concepts

This section explains certain important concepts which form the base for timing in an LCD interface.

Horizontal Back porch (HBP)	HBP denotes the number of pixel clock pulses between the beginning of HSYNC signal and the first valid pixel data.
Horizontal Front Porch (HFP)	HFP denotes the number of pixel clock pulses between the last valid pixel data in the line and the next HSYNC pulse.
Vertical Back Porch (VBP)	VBP denotes the number of lines (HSYNC pulses) between the beginning of VSYNC signal and the first valid line.
Vertical Front Porch (VFP)	VFP denotes the number of lines (HSYNC pulses) between the last valid line of the frame and the next VSYNC pulse.
VSYNC Pulse Width	It is the number of HSYNC pulses during which the VSYNC signal is active.
HSYNC Pulse Width	It is the number of pixel clock pulses during which the HSYNC signal is active.
Active Frame Width	This value is basically the horizontal resolution or the number of pixels in one line. For example, for a WVGA display (800H x 480V), the value of active frame width is equal to 800 pixels.
Active Frame Height	This value is equal to the vertical resolution of the LCD. For example, for a WVGA display (800Hx480V), the value of frame height is equal to 480 pixels.
Screen Width	This is misleading as screen width does not denote the horizontal resolution of the LCD panel.(number of pixels in one line). For the i.MX25, the screen width is the number of pixel clock periods between the last HSYNC and the new HSYNC. So, this value includes the valid pixels and also the horizontal back and front porches.

$$SCREEN\_WIDTH = ACTIVE\_FRAME\_WIDTH + HBP + HFP \qquad \textbf{Eqn. 1}$$

Screen Height	For the i.MX25, the screen height is the number of rows between the last VSYNC pulse and the new VSYNC pulse. It includes all valid lines and also the vertical back and front porch.

$$SCREEN\_HEIGHT = ACTIVE\_FRAME\_HEIGHT + VBP + VFP \qquad \textbf{Eqn. 2}$$

VSYNC Polarity	It is the value of VSYNC which indicates the starting of a new frame. It is active low when the value is 0 or active high when it is 1.
HSYNC Polarity	It is the value of HSYNC which indicates the starting of a new line. It is active low when the value is 0 or active high when it is 1.

3.2.2 Timing Charts

To understand the timing issues in a LCD interface, review the following charts in the datasheet.

- Vertical timing characteristics
- Horizontal timing characteristics
- Pixel clock characteristics

Additionally, if the display uses a serial interface, refer to another chart describing the serial interface and the RESET. This is the information to be extracted from the datasheet when a support for a new LCD panel is added.

Consider a VGA (640H x 480V) LCD panel using the same interface as Chunghwa CLAA057VA01CT as shown in Figure 3. The display uses the complete RGB interface: RGB666, VSYNC, HSYNC, data enable and pixel clock.

3.2.2.1 Vertical Timing

The following sections describe the VGA and WVGA vertical timing characteristics.

VGA Vertical Timing

Figure 6 shows the vertical timing for a hypothetical synchronous display VGA (640H x 480V).

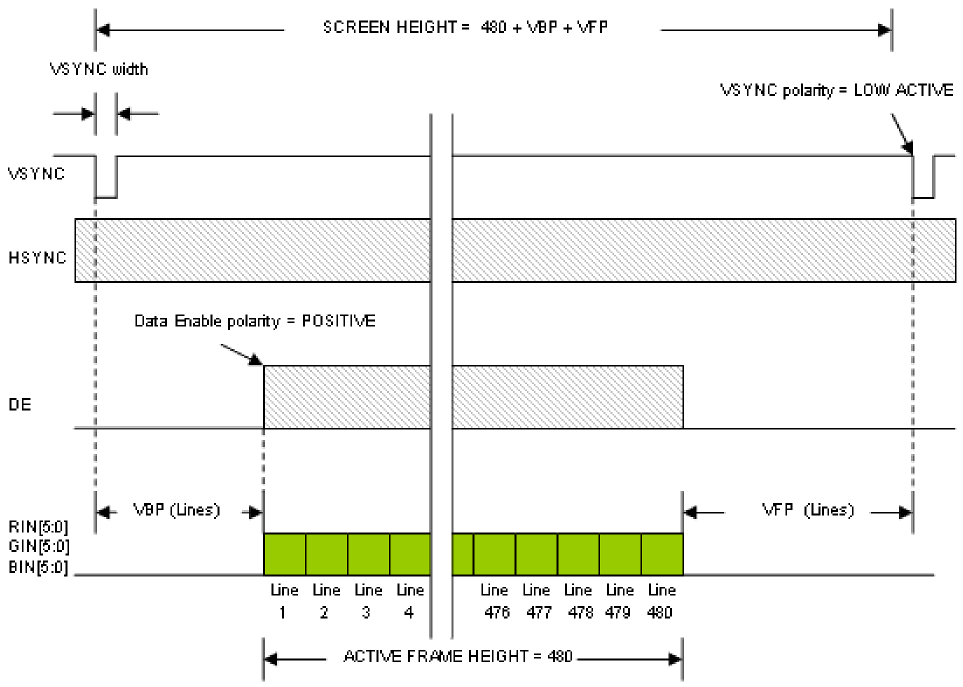


Figure 6. VGA Vertical Timing Example

It is important to mention how signals appear during the VSYNC period. VSYNC period involves a complete frame cycle; every pixel and every line in the frame is sent to the panel during this cycle. The beginning of the frame is set by the VSYNC signal, in this case when signal goes low. Then HSYNC immediately marks the beginning of the first line, in this example it is when HSYNC goes low. To meet the LCD timing requirements, the first lines are designated for the VBP. During VBP, data enable signal is not present and the pixel data on the bus is ignored by the panel. After VBP, data enable signal appears inside the boundaries of the HSYNC period. Details about DE during a line cycle are reviewed in the next section. DE appears consequently during all valid lines (Vertical resolution = 480V). During this time (Active frame height) the LCD panel latches the RGB data on all lines and draws it on the screen. The final stage in the frame cycle is the VFP, where extra lines (HSYNC cycles) appear. During this time, DE remains inactive and again the panel discards any information on the RGB bus. The frame ends when VSYNC signal is set again (goes low).

Table 3 shows the range of the timing parameters shown in Figure 6.

Table 3. VGA Vertical Timing

Parameter	Symbol	Min	Typ	Max	Unit
Screen height or Vertical period	VP	515	525	560	Line
VSYNC pulse width	VSW	1	1	1	Line
Vertical back porch	VBP	34	34	34	Line
Vertical front porch	VFP	1	11	46	Line
Active frame height	VDISP	—	480	—	Line
Vertical refresh rate	FV	55	60	65	Hz

From the information described above, verify some of the timing features. In the first waveform, it is shown that VSYNC polarity is active low, which means that vertical synchronization is normally high, but goes low to indicate the beginning of the new frame. Another feature is the VSYNC width (VSW). Timing has certain flexibility and more than one value can be used to set the timing. It is highly recommended to use the typical values or any values close to them. Here, consider 1 line as VSYNC width.

Vertical back porch (VBP) and vertical front porch (VFP) are shown too; notice that these values are measured in lines or which translates into HSYNC pulses. In this example, vertical back porch is 34 lines, and vertical front porch is of 11 lines width. VSYNC width is included into the VBP stage. This means that VBP starts when VSYNC is set, and not when the VSYNC returns to normal state. VFP is set using H_WAIT_1, but the i.MX25 programming model requires H_WAIT_2 which is defined as the delay from end of HSYNC to the beginning of the OE pulse (VBP - HSYNC pulse width). Using the values described in Table 3, the value of screen height or vertical cycle is 525. In some cases, the value of the VBP and VFP is not given in lines; instead it is expressed in nanoseconds or milliseconds. If this case, an additional calculation must be performed to find the number of lines needed to meet those timings.

WVGA Vertical Timing

If an LCD panel like the hypothetical WVGA (800H X 480V) are used as described in Figure 4 and Figure 5, which does not use HSYNC and VSYNC signals, the waveforms must be analyzed in another perspective as shown in Figure 7.

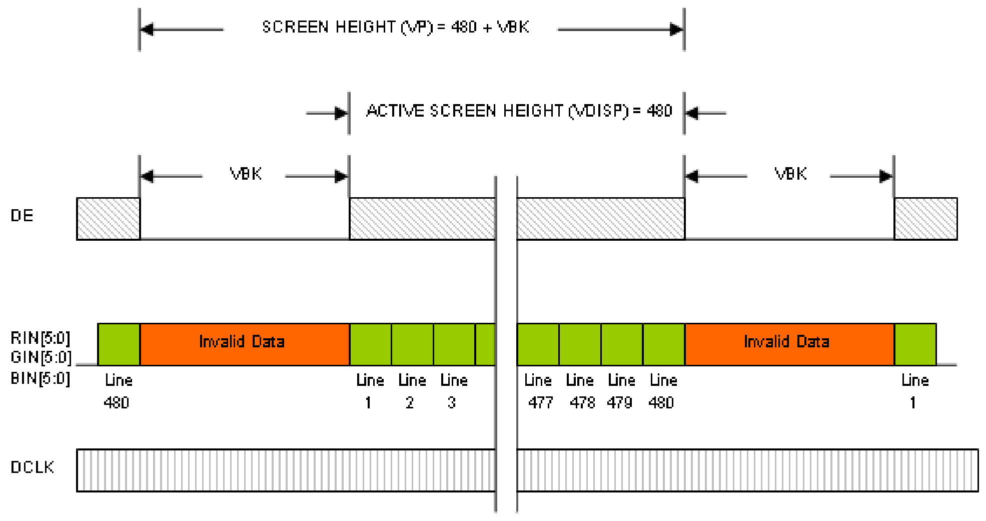


Figure 7. WVGA Vertical Timing Example

Table 4 shows the range of the timing parameters shown in Figure 7.

Table 4. WVGA Vertical Timing

Parameter	Symbol	Min	Typ	Max	Unit
Screen height or Vertical period	VP	490	500	520	Line
Vertical blank	VBK	10	20	40	Line
Active frame height	VDISP	480	480	480	Line
Vertical refresh rate	FV	55	60	65	Hz

In these cases, VSYNC width, VSYNC polarity, vertical back porch and vertical front porch are not shown in the chart. Even when VSYNC is not used, these values are required for configuring the i.MX25 LCDC interface. These waveforms are used to understand the vertical cycle behavior. For the i.MX25, the sequence remains the same; vertical cycle starts with the VSYNC signal, then the rest of the VBP, the active frame area, and finally the VFP appears until the next VSYNC is set. It is important to remember that the values of the VSYNC width, VBP, and VFP can be found during the Vertical blank period.

Figure 8 shows the WVGA vertical timing example with imaginary VSYNC signal.

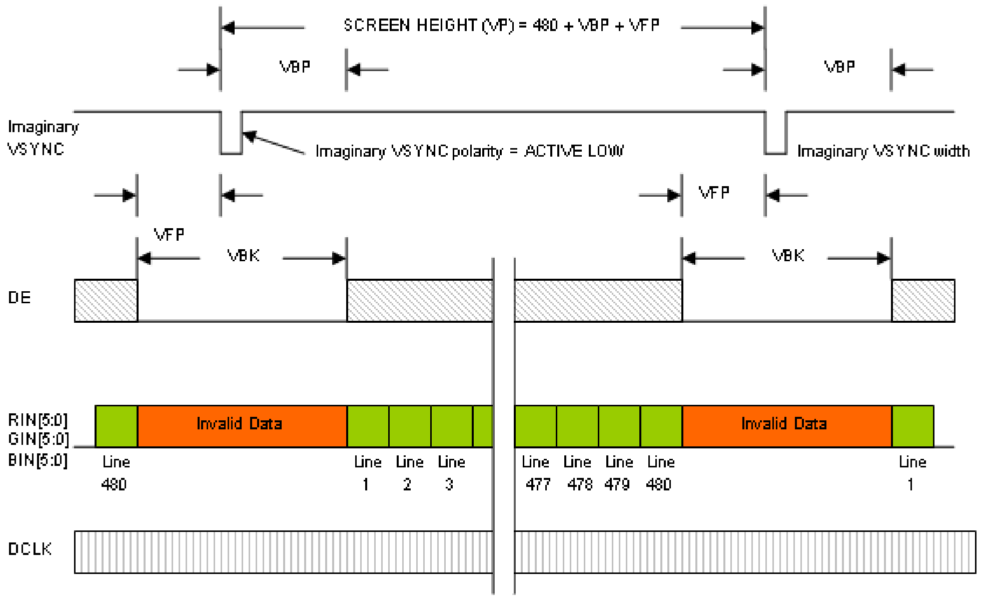


Figure 8. WVGA Vertical Timing Example with Imaginary VSYNC Signal

The VSYNC signal is only used to calculate vertical back porch. When VSYNC is not used, it can be operated with any polarity. However, it is recommended to use VSYNC as an active low signal. VSYNC is usually one line long and thus the value of VSW is 1.To determine the values of VBP and VFP, divide the VBK period into two parts; the first part being the VBP and the second part, the sum of VSYNC and VBP. It is recommended to have an imaginary VSYNC in the VBK and thus a nearly equal values of VFP and VBP.

For example, if VBK is 20 lines (typical), the value of VBP is 10 lines which is equal to VFP. Based on the information described above, a vertical timing table such as Table 5 can be created.

Table 5. WVGA Vertical Timing and Porches

Parameter	Symbol	Min	Typ	Max	Unit
Screen Height or Vertical cycle	VP	490	500	520	Line
VSYNC pulse width	VSW	1	1	1	Line
Vertical back porch	VBP	1	10	40	Line
Vertical front porch	VFP	0	10	39	Line
Vertical blank	VBK	10	20	40	Line
Active frame height	VDISP	480	480	480	Line
Vertical refresh rate	FV	55	60	65	Hz

3.2.2.2 Horizontal Timing

The following sections describe the VGA and WVGA horizontal timing characteristics.

VGA Horizontal Timing

The datasheet, apart from the charts discussed earlier, also includes another chart which describes the line period.

Figure 9 shows the horizontal timing for a hypothetical synchronous display VGA (640H x 480V).

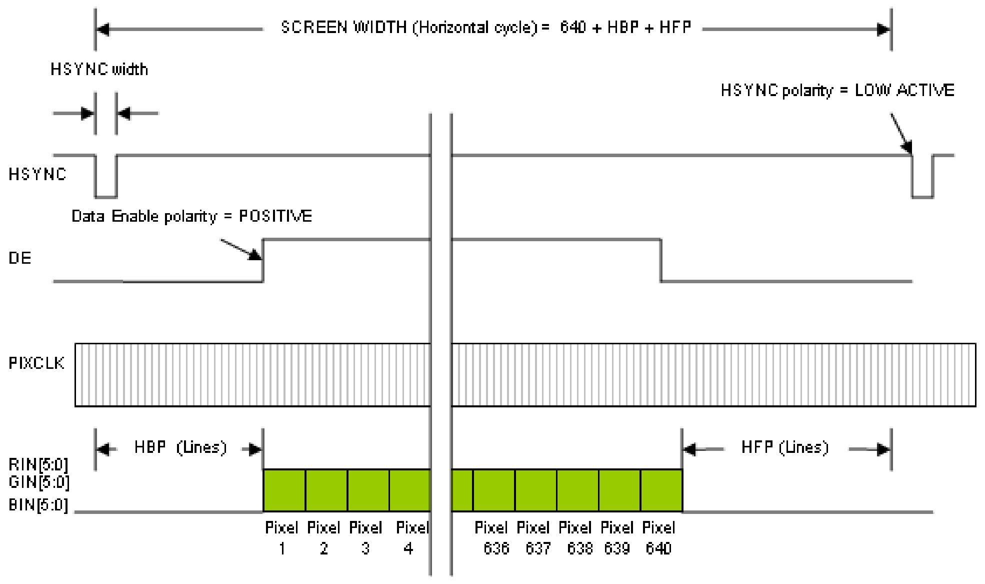


Figure 9. VGA Horizontal Timing Example

The line cycle begins when the HYSNC signal is set, that is when it is low. This is followed by the horizontal back porch stage. During this stage, the data enable signal is inactive. When the data enable signal is set, the next stage, which is the horizontal back porch, begins. During this, the panel latches the RGB data on the bus and draws a new pixel on the screen for every pixel clock pulse. The width of the data enable signal is always equal to the horizontal resolution of the panel. For this example, the width of DE is 640. Once all the pixels in the line are drawn, DE is inactive and the next stage, which is the horizontal front porch, begins. The line cycle ends when HYSNC pulse is set again. Similar to the vertical timing characteristics, a table for the horizontal timing characteristics can be found as shown in Table 6.

Table 6. VGA Horizontal Timing

Parameter	Symbol	Min	Typ	Max	Unit
Screen width or Horizontal cycle	HP	750	800	800	PIXCLK
HSYNC pulse width	HSW	1	1	1	PIXCLK
Horizontal back porch	HBP	46	46	46	PIXCLK

Table 6. VGA Horizontal Timing (continued)

Parameter	Symbol	Min	Typ	Max	Unit
Horizontal front porch	HFP	64	114	214	PIXCLK
Active frame width	HDISP	—	640	—	PIXCLK

WVGA Horizontal Timing

The chart and table that are found in the datasheet is similar to the WVGA (800H X 480V) example and are as shown in Figure 10 and Table 7 respectively.

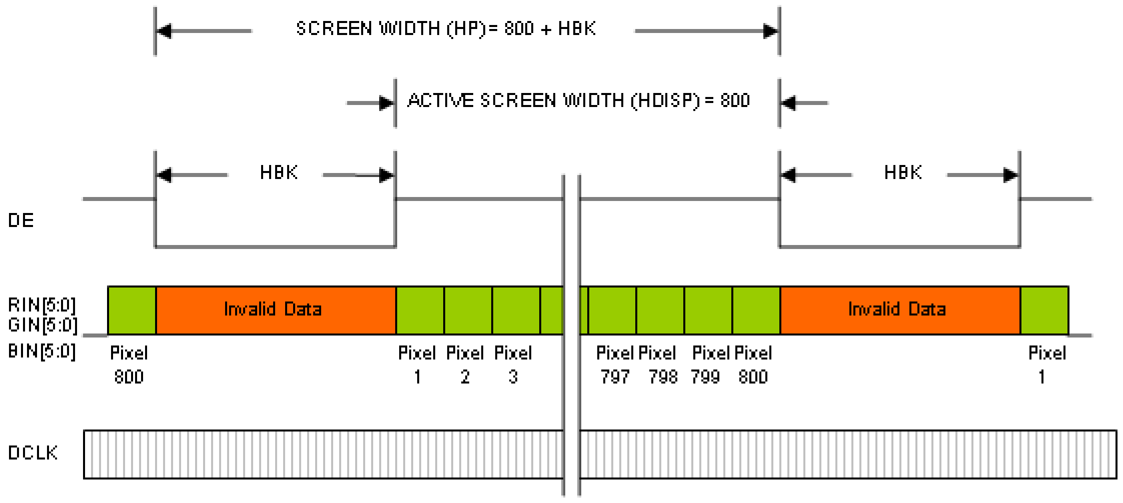


Figure 10. WVGA Horizontal Timing Example

Table 7. WVGA Horizontal Timing

Parameter	Symbol	Min	Typ	Max	Unit
Screen width or Horizontal cycle	HP	850	900	950	PIXCLK
Horizontal blank period	HBK	50	100	150	PIXCLK
Active frame width	HDISP	800	800	800	PIXCLK

The values of HBP, HFP and the HSYNC width are calculated using the same procedure used in the WVGA Vertical Timing to calculate VBP, VHP and the VSYNC width respectively.

The horizontal timing diagram and the table using an imaginary HSYNC signal are as shown in the Figure 11 and Table 8 respectively.

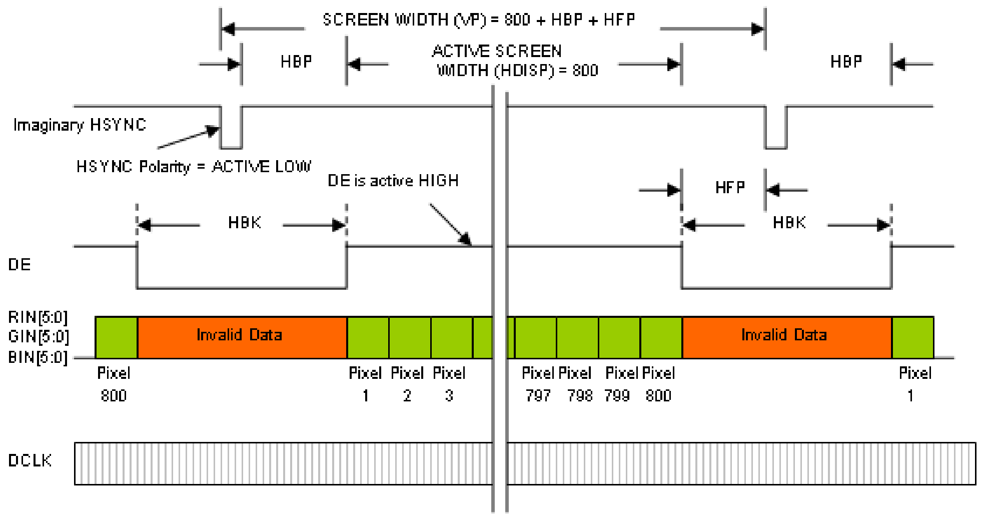


Figure 11. WVGA Horizontal Timing Example with Imaginary HSYNC Signal

Table 8. WVGA Horizontal Timing and Porches

Parameter	Symbol	Min	Typ	Max	Unit
Screen Width or Horizontal cycle	HP	850	900	950	PIXCLK
HSYNC width	HSW	1	1	1	PIXCLK
Horizontal back porch	HBP	1	50	150	PIXCLK
Horizontal front porch	HFP	0	50	149	PIXCLK
Horizontal blank period	HBK	50	100	150	PIXCLK
Active frame width	HDISP	800	800	800	PIXCLK

3.2.2.3 Pixel Clock Timing

The following sections describe the VGA and WVGA pixel clock timing characteristics.

VGA Pixel Clock Timing

The datasheet also contains pixel clock waveform characteristics similar to the timing characteristics.

The waveform characteristics chart and the table for a VGA pixel clock are as shown in the Figure 12 and Table 9 respectively.

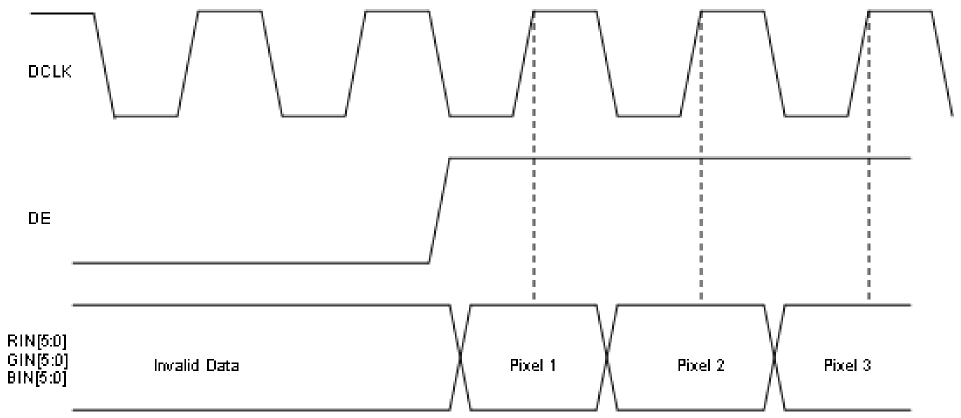


Figure 12. VGA Pixel Clock Timing Example

Table 9. VGA Pixel Clock Timing

Parameter	Symbol	Min	Typ	Max	Unit
Pixel clock frequency	PCLK	23	25	30	MHz

Pixel clock frequency is important to determine the frame refresh rate. Also, it is important to find when RGB data is latched by the panel. This is an important characteristic as the i.MX25 must write the data on the bus one edge before the LCD panel latches the data. In this example, data is latched by the LCD panel on DCLK rising edges and thus the i.MX25 must be configured to write the RGB data on the bus on falling edges. The waveform shows the typical active positive edge of the DCLK. Clock polarity is set in the CLKPOL bit-field in the LPCR register.

The maximum display clock rate can not be greater than one-third of the high speed processing clock rate. HCLK in the i.MX25 PDK BSP is 133 MHz, therefore, the maximum pixel clock is 133 MHz / 3 = 44.4 MHz. However, most LCD displays work at lower frequencies than the typical values. In the i.MX25, the pixel clock rate is the same as LSCLK. PCD value must be set such that LSCLK frequency is not more than one-third of HCLK frequency in TFT.

WVGA Pixel Clock Timing

The waveform characteristics chart and table for a WVGA pixel clock is as shown in the Figure 13 and Table 10 respectively.

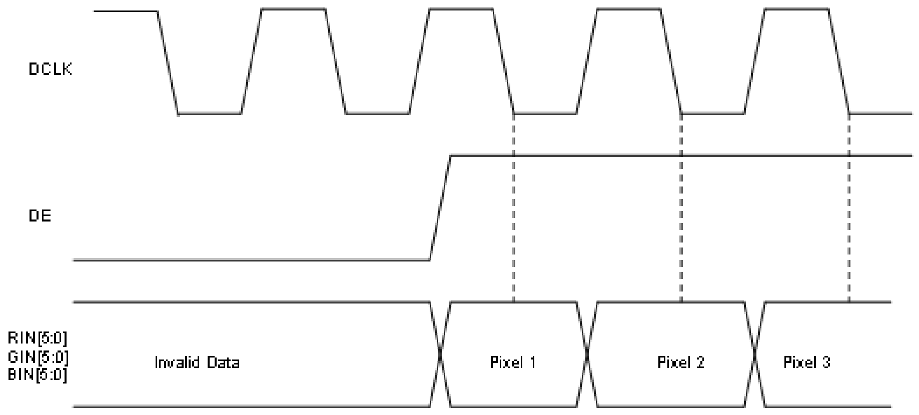


Figure 13. WVGA Pixel Clock Timing Example

Table 10. WVGA Pixel Clock Timing

Parameter	Symbol	Min	Typ	Max	Unit
Pixel clock frequency	PCLK	25	27	32	MHz

Unlike the VGA panel, the WVGA panel latches RGB data on DCLK falling edges. Thus, the i.MX25 must be configured to write the RGB data on the bus on the rising edges. This ensures that the data on the bus is ready and stable when the panel reads it. This waveform shows the active negative edge of LSCLK.

Pixel Polarity

Pixel polarity is the polarity of the signals in the RGB bus that an LCD recognizes as active. For example, consider that i.MX25 must draw a red pixel (only red component) using an RGB565 interface.

- If the LCD is of active high pixel polarity, all the bits other than the RGB bits are low. Thus, the data on the bus would be 0xF800.
- If the LCD is of active low pixel polarity, all the bits other than the RGB bits are high. Thus, the data on the bus would be 0x07FF.

Both 0xF800 and 0x07FF represent the red color but the difference in the values is because of the pixel polarity on the LCD panel. Pixel polarity is configured using the PIXPOL bit-field in the LPCR register.

3.2.3 Custom LCD Timing

Neither of the examples in this application note needs extra signals for LCD functionality. But if the LCD requires a reset signal or initialization routine through a synchronous serial interface, refer to charts similar to the following charts.

3.2.3.1 Reset

Reset

Many LCD panels include an LCD controller which needs an external system reset. If the LCD requires the usage of this signal; finding the timing regarding this pulse is useful.

The Reset width and timing are desribed in Figure 14 and Table 11.

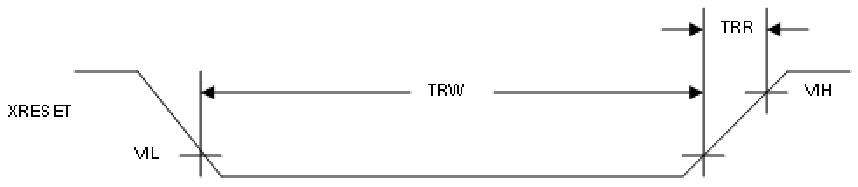


Figure 14. Reset Signal Example

Table 11. Reset Timing

Parameter	Symbol	Min	Typ	Max	Unit
Reset width	TRW	15	—	—	ns
Reset rising time	TRR	—	—	10	ns

$\overline{\text{RESET}}$ must be low for at least 15 ns to ensure a valid reset. The $\overline{\text{RESET}}$ is controlled by i.MX25 GPIO.

NOTE

It is recommended not to use RC circuit to generate this signal as it restricts the rising time of the signal to 10 ns.

Serial command Interface

If the LCD panel has a serial command interface, the datasheet also contains a chart as shown in Figure 15.

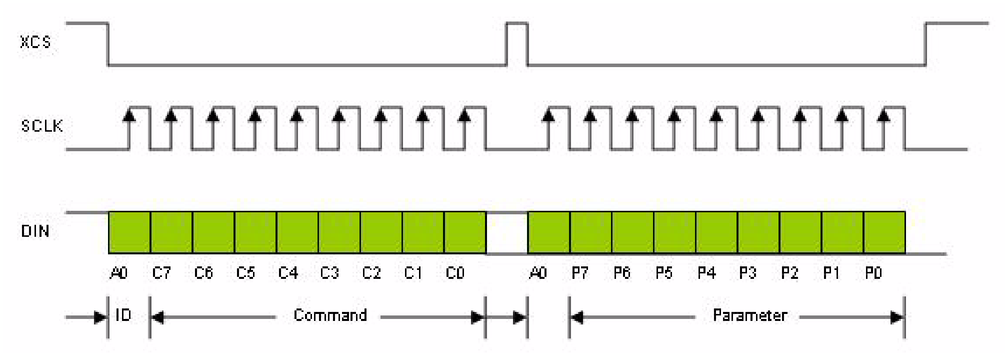


Figure 15. SPI Command Interface Signals Example

NOTE

This application does not review all the serial interfaces that an LCD has. The protocols and data formats are described in the datasheet and it is important to have knowledge of synchronous serial interfaces to configure these settings. For more information, see Chapter 18, Configurable Serial Peripheral Interface (CSPI) of the *i.MX25 Multimedia Applications Processor Reference Manual* .

3.3 LCD Panels Supported by the i.MX25

The i.MX25, at a time, can handle a synchronous panel or an asynchronous panel, but cannot handle both at the same time.

Table 12 lists the various types of displays that the i.MX25 supports.

Table 12. Displays Supported by the i.MX25

Display Controller	Display Type	Interface
LCDC	Synchronous	Parallel interface only
LCDC	Sharp	Parallel interface (Only Sharp 240x320 HR-TFT Panel)
SLCDC	Asynchronous	Serial and Parallel interface

This application note focuses on the LCDC controller. The synchronous LCD interface on the i.MX25 handles LCD devices with the following characteristics:

- Synchronous display (Dumb display)
- RGB interface (RGB888 maximum)
- Resolution not larger than SVGA
- Utilize at least data enable and pixel clock to latch RGB data (some LCD panels need HSYNC and VSYNC signals as well, which are also supported by the i.MX25)
- Pixel clock frequency lower than 44.4 MHz

In addition, the i.MX25 also handles dumb displays with a Sharp interface, but its support is limited to certain models. Since this application note is only intended for non-sharp dumb displays, smart displays and Sharp displays interfaces are not included in this reading.

The i.MX25 is not designed to support LVDS panels and Freescale does not recommend selecting an LCD which uses this interface even to interface the i.MX25 and the LCD by using an external serializer. For more information, see Section 6.1, “Using LVDS with the i.MX25”.

4 Display Configuration in WINCE 6.0

LCD support is one of the most important features for any multimedia device. Display support enables the device to have a graphical user interface and the possibility of becoming an entertainment artifact.

The graphic context is composed of several layers, where the i.MX25 display interface is the final part in the abstraction. All the LCDC and display interface characteristics that were reviewed in previous sections only describes the way the i.MX25 sends the frame buffer to the panel. However, it is important to know who is going to create the frames that need to be sent to the panel. If the screen is refreshed at 60 times per second (60 Hz), every line and every single pixel has to be created to maintain the coherence of the graphic context.

The i.MX25 PDK BSP bases its display driver on the Display Driver Interface (DDI) defined by Microsoft for all WINCE600 devices. Implementing a driver using this model ensures the compatibility of the hardware with the operating system. In other words, once WINCE is loaded and if the driver was created using the MS model, the operating system handles the graphic context, providing all frames.

4.1 WINCE600 Display Driver Development Concepts

Display drivers are loaded and called directly by the graphics, windowing, and event subsystem, called Gwes.exe. Drivers are most commonly written using a layered architecture because of the number of hardware-independent operations.

The Graphics Primitive Engine (GPE) library handles the default drawing, acting as the display driver's model device driver (MDD) upper layer.

The user develops the hardware-specific code that corresponds to the display driver's lower layer, called the platform-dependent driver (PDD).

Table 13 shows the elements that constitute the Windows CE graphics pipeline.

Table 13. Elements of WINCE Graphics Pipeline

Element	Description
Application	The application can be simple, such as a Hello World application, or complex, such as a three-dimensional engineering application. Whichever it is, the application calls GDI functions. Coredll.dll exposes these functions.
Coredll.dll	The major set of functions is exposed through a single DLL, called Coredll.dll. In most cases, this library does not perform the work. Instead, the library packages the parameters for the function call and then triggers a Local Procedure Call (LPC) to another process. The specific process depends on the function call. All drawing and windowing calls are sent to Gwes.exe.

Table 13. Elements of WINCE Graphics Pipeline (continued)

Element	Description
Gwes.exe	The Graphics, Windowing and Events Subsystem (GWES) is responsible for all graphical output and all interactions with the user. The drivers that reside in the GWES address space include display drivers, printer drivers, keyboard drivers, mouse drivers, and touch screen drivers.
Ddi.dll	The default name for the display driver is Ddi.dll. As with most DLLs, Ddi.dll communicates through exported functions. Ddi.dll exports only the DrvEnableDriver function, which returns a pointer to an array of 27 function pointers to the caller. When GWES requires a display driver, it calls one of these 27 functions. Writing a device driver involves writing the code for these 27 functions. Three of these functions are specific to printer drivers, which leaves 24 for the display driver developer.
Hardware	The graphic pipeline ends at the hardware. The display driver communicates to the hardware using the mechanism required by the hardware. This process typically involves a combination of memory-mapped video buffers and I/O registers.

Figure 16 shows the Windows CE graphics architecture.

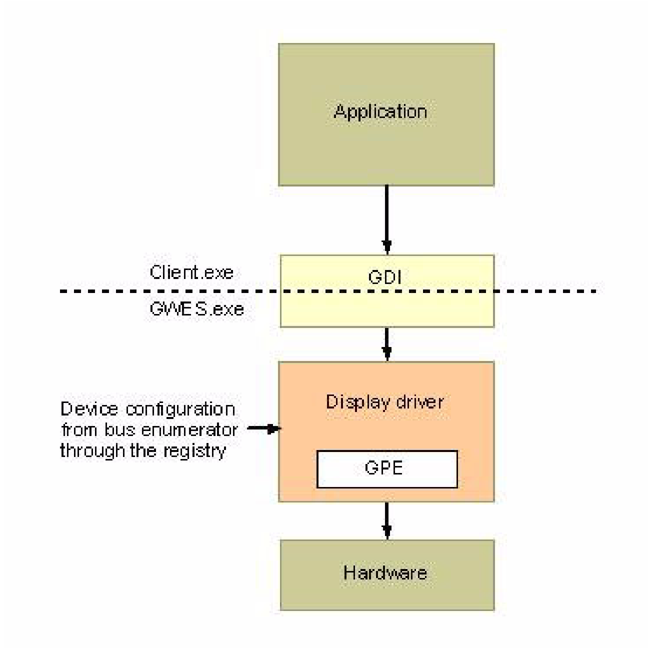

Application
Client.exe
GDI
GWES.exe
Device configuration from bus enumerator through the registry
Display driver
GPE
Hardware

Figure 16. Windows CE Graphics Architecture

More details can be found under Display Drivers (Developing a Device Driver > Windows CE Drivers > Display Drivers) topic of Platform Builder for Microsoft Windows CE 6.0 help.

It is important to mention that developing a WINCE Display driver from scratch implies a considerable effort and knowledge, specially the WINCE architecture. Freescale provides the i.MX25 display driver for synchronous displays in the WINCE600 BSP. The i.MX25 Windows CE 6.0 BSP display driver is based on the Microsoft DirectDraw Graphics Primitive Engine (DDGPE) classes and supports the Microsoft DirectDraw interface. It combines the functionality of a standard LCD display with DirectDraw support and interfaces with LCDC module. It also supports more than one panel which is selected by using the

Windows Register. Support for a new synchronous panel can be added using the procedure described in the following section.

4.2 Adding Support for a New LCD Panel

The following sections describes the procedure used to add the support for a new synchronous panel.

4.2.1 Identifying LCD Characteristics and Timing

To add the support for a new synchronous LCD display for the Freescale i.MX25 BSP, ensure that this panel is compatible with the i.MX25. The panel must have the following interface characteristics.

- Synchronous display (Dumb Display)
- RGB interface (RGB888 maximum)
- Resolution not bigger than SVGA
- Utilize at least data enable and pixel clock to latch RGB data (some LCD's need HSYNC and VSYNC signals which are also supported by the i.MX25).
- Pixel clock frequency lower than 44.4 MHz

Once a compatible LCD panel is selected, it is important to find the timing characteristics of the display interface.

Table 14 and Table 15 can be used to note the timing parameters.

Table 14. LCD Timing Features

Parameter	Symbol	Min	Typ	Max	Unit
Screen height or Vertical cycle	VP				Line
Active frame height	VDISP				Line
VSYNC pulse width	VSW				Line
Vertical back porch	VBP				Line
Vertical front porch	VFP				Line
Vertical refresh rate	FV				Hz
Screen width or Horizontal cycle	HP				PIXCLK
Active frame width	HDISP				PIXCLK
HSYNC pulse width	HSW				PIXCLK
Horizontal back Porch	HBP				PIXCLK
Horizontal front porch	HFP				PIXCLK
Pixel clock frequency	PCLK				MHz

Table 15. LCD Signal Polarities

Parameter	Symbol	Polarity
HSYNC Polarity (LPPOL)	HSP	
VSYNC Polarity (FLMPOL)	VSP	
Data Enable Polarity (OEPOL)	DEP	
Pixel Clock Polarity (CLKPOL)	CLKPOL	
Data Polarity (PIXPOL)	DP	

4.2.1.1 i.MX25 WINCE600 PDK LCD Driver Initialization Flow

The following snippet represents the WINCE600 PDK LCD driver initialization flow.

```
GPE *GetGPE(VOID)
+ MXDDLcdc::MXDDLcdc()
        + MXDDLcdc::Init()
                + BSPInitLCDC()
                        + GetDisplayModeFromRegistry()
                        + InitializeGPEMode()
                        + Enable LCDC module pins (LD0-LD17,HSYNC,VSYNC,OE,LSCLK,CONTRAST)
                + BSPGetModes()
                + BSPGetModeDescriptor()
        + MXDDLcdc::InitHardware()
                + CreateThread(..,MXDDLcdcIntr,..)
MXDDLcdc::SetMode()
+ AllocVideoMemory()
+ BSPGetModeDescriptor()
+ GetRotateModeFromReg()
+ SetRotateParams()
+ AllocSurface() - Create primary surface and blank surface
+ SetModeHardware()
        + LCDCEnableClock()
        + Configure i.MX25 LCDC with panel settings (LCDC_MODE_DESCRIPTOR)
        + BSPTurnOnDisplay()
+ DynRotate()
+ OffsetInVideoMemory()
+ DDIPUSurf::SetVisibleSurface()
+ SetRotation()
```

LCD driver initialization begins with the `GetGPE()` function and `MXDDLcdc` (i.MX25 Direct Draw LCD Controller) object is created. Initialization begins in `MXDDLcdc::Init` by calling `BSPInitLCDC()` when the panel that must be used is read from the registry. Then the `GPEMode` is created for the LCD resolution specified in the `LCD_MODE_DESCRIPTOR` of that panel and the i.MX25 LCDC pins are configured to be used by the LCD interface. Then, `MXDDLcdc::SetMode()` is executed and video memory size is read from registry to allocate the video buffer. Rotation settings are also configured according with the register settings. After that `SetModeHardware()` is called, this function is used to configure all LCDC registers using the `LCDC_MODE_DESCRIPTOR` and also to execute the LCD panel initialization routine.

4.2.1.2 i.MX25 WINCE600 PDK SDK1.6 LCD Display Related Files

The files related to the i.MX25 WINCE600 PDK SDK1.6 LCD display are found in the following folder locations.

`WINCE600\PLATFORM\iMX25-3DS-PDK1_6\SRC\DRIVERS\LCDC\bsplcdc.h`

The `bsplcdc.h` file contains the structures and definitions required for the LCDC driver.

`WINCE600\PLATFORM\iMX25-3DS-PDK1_6\SRC\DRIVERS\LCDC\bsplcdc.cpp`

The `bsplsdc.cpp` file contains the BSP part of the LCDC driver implementation.

`WINCE600\PLATFORM\COMMON\SRC\SOC\COMMON_FSL_V2_PDK1_6\LCDC\DDLcdc.cpp`

The `DDLcdc.c` file contains the implementation of Direct Draw LCDC class (DDLCDC).

`WINCE600\PLATFORM\iMX25-3DS-PDK1_6\SRC\DRIVERS\LCDC\mx25_lcdc.reg`

The `mx25_lcdc.reg` file contains the LCDC driver register configuration settings.

4.2.1.3 i.MX25 PDK LCD Structures and Bit Fields

It is important to understand how and where the information related to the LCD panels settings (see Section 4.2, "Adding Support for a New LCD Panel.") are stored.

The `LCDC_MODE_DESCRIPTOR` structure contains information of the synchronous LCD or TV display. This structure contains `GPEMode` and `PCR_CFG`. `LCDC_MODE_DESCRIPTOR` structure is the structure that is modified to add support to a new panel.

The `ModeArray[]` in bsplcdc.cpp file is the global array that stores the `LCDC_MODE_DESCRIPTOR` for all supported displays (LCD, NTSC TV and PAL TV). Though, there are many ways to modify the driver, it is recommended to add a new `LCDC_MODE_DESCRIPTOR` entry with the new LCD features in the `ModeArray[]`.

LCDC_MODE_DESCRIPTOR

`LCDC_MODE_DESCRIPTOR` is the main structure for LCD timing and features, and it contains information related to the signal polarity, width and height of the primary surface, bits per pixel, refresh rate and so on.

The `LCDC_MODE_DESCRIPTOR` structure is given below.

```
typedef struct {
         DWORD    dwPanelType;
         GPEMode  GPEModeInfo;
         UINT32   Hwidth;
         UINT32   Hwait1;
         UINT32   Hwait2;
         UINT32   Vwidth;
         UINT32   Vwait1;
         UINT32   Vwait2;
         union
         {
                  PCR_CFG  PCRCfg;
                  UINT32   uPCRCfg;
         } PCR_CFG;
}LCDC_MODE_DESCRIPTOR, *PLCDC_MODE_DESCRIPTOR;
```

Table 16 shows the description of each element of the `LCDC_MODE_DESCRIPTOR` structure.

Table 16. LCDC_MODE_DESCRIPTOR Structure Members

Data type	Var. name	Description	Symbol	Unit
DWORD	dwPanelType	Panel type index on ModeArray structure[1]	—	—
GPEMode	GPEModeInfo	Primary surface graphic features	—	—
UINT32	Hwidth	HSYNC pulse width	HSW	Pixels
UINT32	Hwait1	Horizontal front Porch	HFP	Pixels
UINT32	Hwait2	Horizontal back porch without HSYNC width	HBP-HSW	Pixels
UINT32	Vwidth	VSYNC pulse width	VSW	Lines
UINT32	Vwait1	Vertical back porch without VSYNC width	VBP-VSW	Lines
UINT32	Hwait2	Vertical front porch	VFP	Lines
PCR_CFG	PCRCfg	Bit field of the PCR register	—	—

[1] This number represents the PanelType enum for this LCD panel into the bsplcdc.h header file. This enumeration is also used to distinguish the `LCDC_MODE_DESCRIPTOR` index of the ModeArray[] in the bsplcdc.cpp file. CHUNGHWA CLAA057VA01CT VGA panel is the final element in both PanelType enum and ModeArray[]. The value is taken from the register settings placed in the mx25_lcdc.reg file [HKEY_LOCAL_MACHINE\Drivers\Display\LCDC].

GPEMode

The WINCE graphics engine uses the `GPEMode` structure to determine details such as, size and format of the screen. The operating system uses this information to create frames of appropriate size and sends them to the panel using the display interface.

The `GPEMode` structure is given below.

```
struct GPEMode {
         int modeId;
         int width;
         int height;
         int Bpp;
         int frequency;
         EGPEFormat format;
};
```

Table 17 shows the description of each element in the `GPEMode` structure.

Table 17. GPEMode Structure Members

Data type	Var. name	Description	Symbol	Unit
int	modeId	Number determined by the developer[1]	—	—
int	width	Width of primary surface, or screen in pixels	HDISP	Pixels
int	height	Height of primary surface, or screen in lines	VDISP	Lines
int	bpp	Bits per pixel[2]	—	—
int	frequency	Frame rate frequency	FV	Hz
EGPEFormat	format	RGB Representation[3]	—	—

[1] The element modeId is an enumeration and is located in the file bsplcdc.cpp. beginning with modeid_1, but this number is only used as classification method. There is no action regarding the value of this parameter. As a suggestion, use the next value of the modeid_# enum or any other value.

[2] Bits per pixel (bpp) of the frame buffer can be 4, 8, 12, 16, 18 or 24. For RGB666 panels,16 bpp has half the number of memory transfers than 18 bpp with a nearly equal image quality.

[3] This value must be equivalent to the bits per pixel settings. It means that for 2 bpp the format must be `gpe2Bpp`, for 16 bpp `gpe16Bpp` and so on.

```
static EGPEFormat eFormat[] =
{
  gpe1Bpp,
  gpe2Bpp,
  gpe4Bpp,
  gpe8Bpp,
  gpe16Bpp,
  gpe24Bpp,
  gpe32Bpp,
  };
```

PCR_CFG Register Bit Fields

The `PCR_CFG` register structure is given below.

```
// Bitfield of the PCR register
typedef struct {
        UINT32 PCD:6;               // set by driver
        UINT32 SHARP:1;             // 0 Disable Sharp signals 1 Enable Sharp signals
        UINT32 SCLKSEL:1;           // 0 Disable OE and LSCLK in TFT mode when no data output
                                    // 1 Always enable LSCLK in TFT mode
        UINT32 ACD:7;               // set by driver
        UINT32 ACDSEL:1;            // ADC Clock source selection
        UINT32 REV_VS:1;            // Vertical scan mode (normal, reverse)
        UINT32 SWAP_SEL:1;          // Bit swap for endianess conversion
        UINT32 END_SEL:1;           // 0 Little Endian 1 Big Endian
        UINT32 SCLKIDLE:1;          // 0 Disable LSCLK 1 Enable LSCLK
        UINT32 OE_POL:1;            // 0 active high 1 active low
        UINT32 CLK_POL:1;           // 0 active negative edge, 1 active positive edge
        UINT32 LP_POL:1;            // 0 active high  1 active low
        UINT32 FLM_POL:1;           // 0 active high  1 active low
        UINT32 PIXEL_POL:1;         // 0 active high  1 active low
        UINT32 BPIX:3;              // BPP (set by driver)
        UINT32 MONO_BUS_SIZE:2;     // mono bus width 1 4 or 8 bit
        UINT32 COLOR_MODE:1;        // Panel color (mono or color)
        UINT32 TFT_MODE:1;          // Panel mode (Actif or Passif)
} PCR_CFG, *PPCR_CFG;
```

Table 18 shows the description of each element in the PCR_CFG register structure.

Table 18. PCR_CFG Register Bitfields

Offset	Bit field name	Description	Symbol
0	PCD	Pixel clock divider (set by driver)	—
6	SHARP	Sharp Panel Enable (For Non-Sharp panels set to: LCDC_PCR_SHARP_DISABLE)	—
7	SCLKSEL	LSCLK select[1]	—
8	ACD	Alternate crystal direction (set by driver)	—
15	ACDSEL	ACD Clock source select, always use: LCDC_PCR_ACDSEL_USE_LPHSYNC	—
16	REV_VS	Reverse vertical scan[2]	—
17	SWAP_SEL	Swap select[3]	—
18	END_SEL	Image download into memory Endian select[4]	—
19	SCLKIDLE	Enables/disables LSCLK when VSYNC is idle in TFT mode[5]	—
20	OEPOL	Output enable polarity	DEP
21	CLKPOL	Pixel clock polarity	CLKPOL
22	LPPOL	HSYNC Polarity	HSP
23	FLMPOL	VSYNC Polarity	VSP
24	PIXPOL	Pixel Polarity	DP

Table 18. PCR_CFG Register Bitfields (continued)

Offset	Bit field name	Description	Symbol
25	BPIX	Bits per pixel	BPP
28	PBSIZ	Panel Bus Width (Not used for TFT)	—
30	COLOR	Interfaces to Color Display, set to LCDC_PCR_COLOR_COLOR	—
31	TFT_MODE	Interfaces to TFT Display, set to LCDC_PCR_TFT_ACTIVE	—

[1] `SCLKSEL` selects whether to enable or disable LSCLK in TFT mode when there is no data output. Unless there is any LCD panel restriction, this field must be `LCDC_PCR_SCLKSEL_ENABLE`.

[2] Selects the vertical scan direction as normal or reverse image flips along the x-axis. Unfortunately there isn't horizontal flip hardware acceleration, so 180° rotation can not be done by the LCDC. Usually this field is `LCDC_PCR_REV_VS_NORMAL` to avoid vertical flipping.

[3] LCDC operates in big endian mode internally. Swap select controls the swap of data before operation in little endian mode.

[4] Endian select. Selects the image download into memory as big or little endian format. (`LCDC_PCR_END_SEL_LITTLE_ENDIAN` or `LCDC_PCR_END_SEL_BIG_ENDIAN`)

[5] Enables/disables LSCLK when VSYNC is idle in TFT mode. Unless there is a waveform restriction this field must be set to `LCDC_PCR_SCLKIDLE_ENABLE`. Additionally, use `LCDC_PCR_SCLKIDLE_DISABLE` but most of the times panels does not expect this waveform behavior.

5 Modifying the BSP

To add support the i.MX25 PDK WINCE600 BSP, there are a few steps to be followed.

1. Modify the catalog
2. Modify mx25_lcdc.reg
3. Modify platform.bib
4. Set up power LCD voltages
5. Set up reset signal
6. Set up backlight
7. Add a Panel type enum for the new LCD
8. Create the `LCDC_MODE_DESCRIPTOR` entry for the new LCD
9. Write initialization command routine
10. Rebuild the project

Consider using an LCD panel similar to Chunghwa CLAA070VC01. The panel is an LCD module composed of a synchronous panel, internal backlight unit, and an external touch screen. RGB interface is 18 bpp and it uses HSYNC, VSYNC, DE and PIXCLK. Figure 3 and Figure 4 show the connections between i.MX25 and this type of LCD.

5.1 Modifying the WINCE600 Catalog

The procedure to modify WINCE600 catalog using Microsoft Visual Studio is as follows:

1. If the iMX25-3DS-PDK1_6-Mobility project is open, click File > Close Solution to close the project.
2. Click File > Open > File to open the catalog file under \WINCE600\PLATFORM\iMX25-3DS-PDK1_6\CATALOG folder as shown in Figure 17.

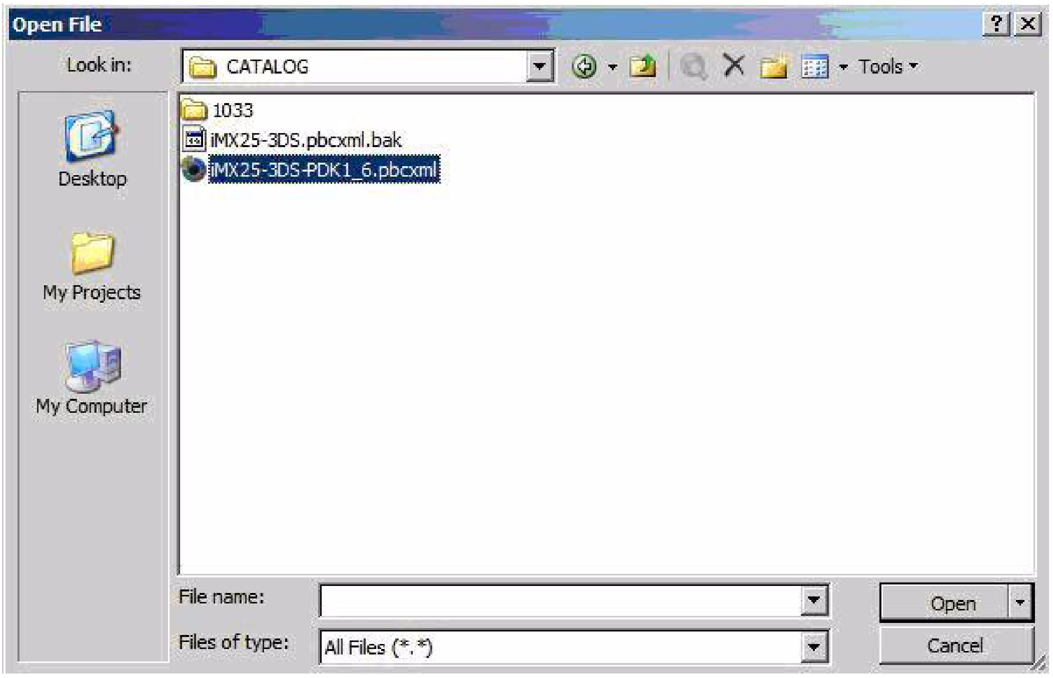


Figure 17. Opening Catalog File

3. Create a new entry for the CHUNGHWA display under Catalog > Third Party > BSP > Freescale i.MX25 3DS PDK1_6: ARMV4I > Device Drivers > Display folder.

Right-click the Display folder and select Add Catalog Item as shown in Figure 18.

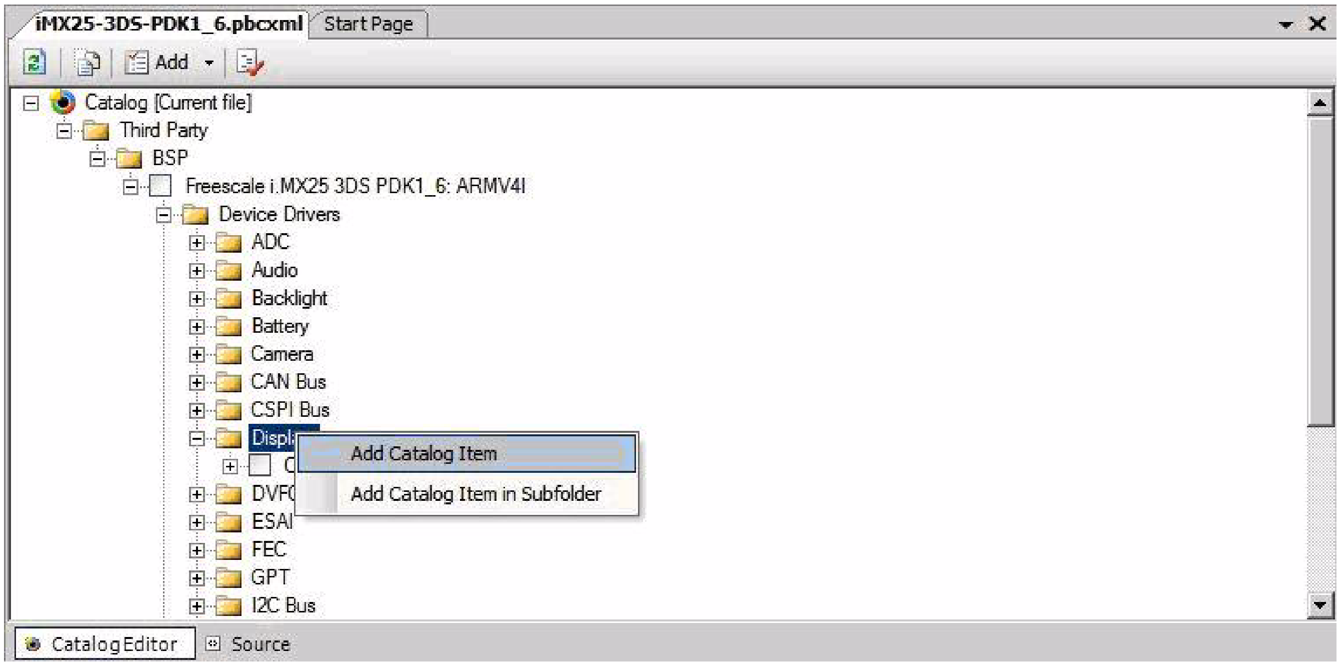


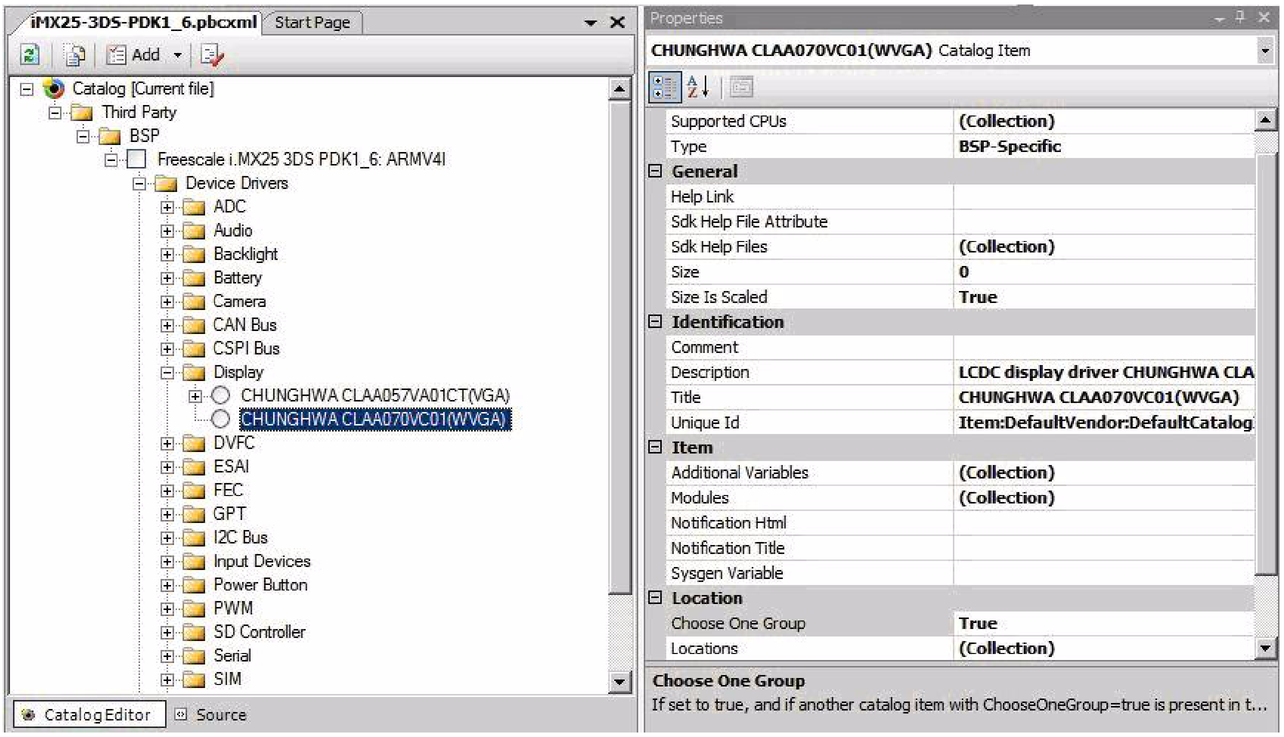
Figure 18. Add Catalog Item

4. Modify the properties of the catalog item as listed in Table 19.

Table 19. Display Driver Catalog Item Properties

Properties	Value
Description	LCDC display driver CHUNGHWA CLAA070VC01
Title	CHUNGHWA CLAA070VC01(WVGA)
Additional Variables	BSP_DISPLAY_CHUNGHWA_CLAA070VC01 BSP_LCDC
Modules	lcdc.dll
Choose One Group	True

As shown in Figure 19, all the other properties must have the default value.



iMX25-3DS-PDK1_6.pbcxml | Start Page

- Catalog [Current file]
  - Third Party
    - BSP
      - Freescale i.MX25 3DS PDK1_6: ARMV4I
        - Device Drivers
          - ADC
          - Audio
          - Backlight
          - Battery
          - Camera
          - CAN Bus
          - CSPI Bus
          - Display
            - CHUNGHWA CLAA057VA01CT(VGA)
            - CHUNGHWA CLAA070VC01(WVGA)
          - DVFC
          - ESAI
          - FEC
          - GPT
          - I2C Bus
          - Input Devices
          - Power Button
          - PWM
          - SD Controller
          - Serial
          - SIM

Properties

CHUNGHWA CLAA070VC01(WVGA) Catalog Item

Supported CPUs	(Collection)
Type	BSP-Specific
General	
Help Link	
Sdk Help File Attribute	
Sdk Help Files	(Collection)
Size	0
Size Is Scaled	True
Identification	
Comment	
Description	LCDC display driver CHUNGHWA CLA
Title	CHUNGHWA CLAA070VC01(WVGA)
Unique Id	Item:DefaultVendor:DefaultCatalog
Item	
Additional Variables	(Collection)
Modules	(Collection)
Notification Html	
Notification Title	
Sysgen Variable	
Location	
Choose One Group	True
Locations	(Collection)

Choose One Group
If set to true, and if another catalog item with ChooseOneGroup=true is present in t…

Figure 19. Catalog Item Properties

5. Modify the properties of the CUNGHWA CLAA057VA01CT(VGA) catalog item as listed in Table 20.

Table 20. CLAA057VA01CT(VGA) Catalog Item Properties

Properties	Value
Description	#FSL:iMX25-3DS-PDK1_6:CLAA057VA01CT:Description
Title	#FSL:iMX25-3DS-PDK1_6:CLAA057VA01CT:Title
Additional Variables	BSP_DISPLAY_CHUNGHWA_CLAA057VA01CT BSP_LCDC
Modules	lcdc.dll
Choose One Group	True

6. Click OK on variables collection message box. Save and close the file and again open the iMX25-3DS-PDK1_6-Mobility project.

   Then, refresh the catalog by using the button on the top of the catalog items view.

When the iMX25-3DS-PDK1_6-Mobility project is opened again in Visual Studio, the CHUNGHWA CLAA070VC01(WVGA) entry is also available in the catalog. For the new LCD support to be completed has been completed some files need to be modified. CHUNGHWA CLAA057VA01CT(VGA) is included in the project until the platform.bib file is modified.

5.2 Modifying the mx25_lcdc.reg File

Open `mx25_lcdc.reg` file and add the register configuration settings for the new panel. Basically add a new IF condition for the BSP_DISPLAY_CHUNGHWA_CLAA070VC01 entry, where `PanelType` index identifier must be set. Also it is recommended to use BSP_LCDC variable to ensure that display registry entries are included only when a display panel has been selected.

```
IF BSP_LCDC
[HKEY_LOCAL_MACHINE\Drivers\Display\LCDC]
; Panel type: This value must match with bsplcdc.h PanelType.
; 0: CHUNGHWA CLAA057VA01CT
        IF BSP_DISPLAY_CHUNGHWA_CLAA057VA01CT
                "PanelType"=dword:0;LCD Panel identifier,0 for CHUNGHWA CLAA057VA01CT 16BPP
                "VideoMemSize"=dword:900000  ; 9MB
        ENDIF
; Panel type: This value must match with bsplcdc.h PanelType.
; 1: CHUNGHWA CLAA070VC01
        IF BSP_DISPLAY_CHUNGHWA_CLAA070VC01
                "PanelType"=dword:1;LCD Panel identifier, 1 for CHUNGHWA CLAA070VC01(WVGA)
                "VideoMemSize"=dword:900000  ; 9MB
        ENDIF
ENDIF BSP_LCDC
```

`PanelType` is determined by the enum created for this panel in bsplcdc.h header file. Also must be the same value than the offset of the CLAA070VC01 WVGA LCD_MODE_DESCRIPTOR in the `ModeArray[]` array located in `bsplcdc.cpp` file (see Section 5.7, “Adding a Panel Type enum for the New LCD” and for more information, see Section 5.8, “Creating the LCDC_MODE_DESCRIPTOR Entry for the New LCD”).

5.3 Modifying the platform.bib File

Add a conditional variable for include display driver DLL (lcdc.dll) in the image when any of the displays has been selected.

```
; Display Driver
; @CESYSGEN IF CE_MODULES_DISPLAY
IF BSP_NODISPLAY !
#if (defined BSP_DISPLAY_CHUNGHWA_CLAA057VA01CT || defined BSP_DISPLAY_CHUNGHWA_CLAA070VC01)
lcdc.dll             $(_FLATRELEASEDIR)\lcdc.dll                 NK   SHK
#endif
ENDIF ; BSP_NODISPLAY !
; @CESYSGEN ENDIF CE_MODULES_DISPLAY
```

5.4 Setting up Power LCD Voltages

Before connecting any LCD panel to the i.MX25 PDK, it is important to verify if the LCD can be powered with the proper supply voltages and if the display data interface has the correct value of VIO. Power settings are handled by the MC34704 PMIC and some other voltage regulators. During PDK initialization, the PMIC driver must configure the voltage settings in order to set up the power voltage configuration.

Chunghwa CLAA070VC01 WVGA Panel requires two different power supply levels, VDD (5.0V) and VCC (3.3V). In the i.MX25 PDK these two voltages are directly connected to a voltage regulator which can not be modified by software.

The most common power supply levels are included in the i.MX25 PDK. Voltages 1.8V, 2.8V, 3.3V and 5.0 V are found at UI generic I/F connector (J4). Always connect the LCD to the proper power supply lines with this connector.

These two panels does not have a turn ON/OFF method. Therefore, it is necessary to add a circuit to control the power management in those devices. The circuit is controlled by a GPIO pin (LCD_EN), which is connected to the i.MX25 GPIO PORT3 GPIO [17]. This pin needs to be configured in `BSPInitLCDC()` function which is used as GPIO. i.MX25 WINCE600 BSP provides a function where specific power settings and initialization commands must be placed. This function is located in bsplcdc.cpp and its name is `BSPTurnOnDisplay()`. The signal is LOW active so DDKGpioWriteDataPin function is used to set the i.MX25 GPIO PORT3 GPIO [17] pin to 0 in order to enable the LCD power.

```
bsplcdc.cpp

BOOL BSPInitLCDC(VOID)

{
. . .
// LCD_EN
         DDKIomuxSetPinMux(DDK_IOMUX_PIN_VSTBY_REQ,
                           DDK_IOMUX_PIN_MUXMODE_ALT5,
                           DDK_IOMUX_PIN_SION_REGULAR);
         DDKIomuxSetPadConfig(DDK_IOMUX_PAD_VSTBY_REQ,
                                    DDK_IOMUX_PAD_SLEW_SLOW,
                                    DDK_IOMUX_PAD_DRIVE_NORMAL,
                                    DDK_IOMUX_PAD_OPENDRAIN_DISABLE,
                                    DDK_IOMUX_PAD_PULL_UP_100K,
                                    DDK_IOMUX_PAD_HYSTERESIS_DISABLE,
                                    DDK_IOMUX_PAD_VOLTAGE_3V3);
. . .
         return TRUE;
}

void BSPTurnOnDisplay(PLCDC_MODE_DESCRIPTOR pModeDesc)
{
         TurnOnLCD(pModeDesc);
}

BOOL TurnOnLCD(PLCDC_MODE_DESCRIPTOR pModeDesc)
{
         UNREFERENCED_PARAMETER(pModeDesc);

         DDKGpioSetConfig(DDK_GPIO_PORT3,17,DDK_GPIO_DIR_OUT, DDK_GPIO_INTR_NONE);
         DDKGpioWriteDataPin(DDK_GPIO_PORT3,17,0);

         return TRUE;
}
```

5.5 Setting up the Reset Signal

Many synchronous display modules need a reset signal. It is recommended that this signal be controlled by the microprocessor. Neither WVGA nor VGA LCDs mentioned in this application note need a reset signal. If the LCD requires a reset (LCD_LCS1_RST), the GPIO PORT4 GPIO[5] pin is used. This pin needs to be configured using `DDKIomuxSetPinMux()` and `DDKIomuxSetPadConfig()` functions when `BSPInitLCDC()` is executed. LCD_LCS1_RST can be accessed through connector J4. And the reset signal assertion must be called during `MXDDLcdc::SetModeHardware()` execution after `BSPTurnOnDisplay()` call and prior the initialization command routine or in the order the LCD requires it.

```
DDLcdc.cpp
//------------------------------------------------------------------------------
// External Functions
//------------------------------------------------------------------------------
extern void   BSPResetDisplay(PLCDC_MODE_DESCRIPTOR pModeDesc);

BOOL MXDDLcdc::SetModeHardware(int modeNo)
{
....
         LCDCEnableClock(TRUE);
         BSPResetDisplay(&m_ModeDesc);
         BSPTurnOnDisplay(&m_ModeDesc);
....
}
bsplcdc.cpp
BOOL BSPInitLCDC(VOID)
{
. . .
// LCD_LCS1_RST
         DDKIomuxSetPinMux(DDK_IOMUX_PIN_D15,
                           DDK_IOMUX_PIN_MUXMODE_ALT5,
                           DDK_IOMUX_PIN_SION_REGULAR);
         DDKIomuxSetPadConfig(DDK_IOMUX_PAD_D15,
                           DDK_IOMUX_PAD_SLEW_SLOW,
                           DDK_IOMUX_PAD_DRIVE_NORMAL,
                           DDK_IOMUX_PAD_OPENDRAIN_DISABLE,
                           DDK_IOMUX_PAD_PULL_UP_100K,
                           DDK_IOMUX_PAD_HYSTERESIS_DISABLE,
                           DDK_IOMUX_PAD_VOLTAGE_3V3);
         return TRUE;
}
void BSPResetDisplay(PLCDC_MODE_DESCRIPTOR pModeDesc)
{
         UNREFERENCED_PARAMETER(pModeDesc);

         DDKGpioSetConfig(DDK_GPIO_PORT4,5,DDK_GPIO_DIR_OUT, DDK_GPIO_INTR_NONE);
         DDKGpioWriteDataPin(DDK_GPIO_PORT4,5,1);
         Sleep(200);
         DDKGpioWriteDataPin(DDK_GPIO_PORT4,5,0);
         Sleep(200);
         DDKGpioWriteDataPin(DDK_GPIO_PORT4,5,1);
         return;
}
```

5.6 Setting up the Backlight

Ensure that the backlight is turned ON, as the contents on the LCD panel can be seen only when the backlight is on. There are many ways to control the backlight level through a PWM signal.In the case of the i.MX25 PDK, this signal is generated by an LCDC pin called CONTRAST. If the LCD backlight control signal is connected to this pin (LCD_CONTRAST), perform any additional change to adapt the backlight control.

5.7 Adding a Panel Type enum for the New LCD

A new `PanelType` enumeration must be created for this new panel if support to another LCD panel is to be added. `PanelType` enumeration can be found in `bsplcdc.h` header file. The offset after this entry represents the position of the new synchronous `LCDC_MODE_DESCRIPTOR` structure in the `ModeArray[]` in bsplcdc.cpp file and also the `PanelType` value of this panel in the `mx25_lcdc.reg` file.

```
bsplcdc.h
typedef enum PanelType_c
{
         DISPLAY_CHUNGHWA_CLAA057VA01CT = 0, // VGA display
         DISPLAY_CHUNGHWA_CLAA070VC01,      // WVGA display
// Add your display panel here
// ...
         numPanels  // Define the numPanel so value automatically the number of panel
}PanelType;
```

5.8 Creating the LCDC_MODE_DESCRIPTOR Entry for the New LCD

Based on the information in LCDC_MODE_DESCRIPTOR, fill the `LCDC_MODE_DESCRIPTOR` structure for the new synchronous panel and add it to the `ModeArray[]` on `bsplcdc.cpp` file. The position in the array of the new LCD is determined by its `PanelType` enumeration. See the code below for the CHUNGHWA VGA CLAA070VC01T dumb panel.

```
bsplcdc.cpp
// All mode supported description.
LCDC_MODE_DESCRIPTOR ModeArray[] =
{
// DISPLAY_CHUNGHWA_CLAA057VA01CT 16 BPP
{
. . . .
},
// DISPLAY_CHUNGHWA_CLAA070VC01 (WVGA) 16 BPP
{
         DISPLAY_CHUNGHWA_CLAA070VC01,
         {                // - - GPEMode - -
                  modeid_1,
                  800,         // Active Frame Width
                  480,         // Active Frame Height
                  16,          // BPP (RGB565)
                  60,          // Refresh rate 60 Hz
                  gpe16Bpp
},
         1,               // HSYNC Width
         50,              // Horizontal Front porch
```

```
        49,             // Horizontal Back Porch - HSYNC Width
        1,              // VSYNC Width
        10,             // Vertical Front Porch
        9,              // Vertical Back Porch - VSYNC Width
        {                                   // - - PCR_CFG bitfield - -
                0,                               // PCD (set by driver)
                LCDC_PCR_SHARP_DISABLE,          // SHARP
                LCDC_PCR_SCLKSEL_ENABLE,         // SCLKSEL
                0,                               // Not used for TFT panel
                LCDC_PCR_ACDSEL_USE_LPHSYNC,     // ACDSEL
                LCDC_PCR_REV_VS_NORMAL,          // REV_VS
                LCDC_PCR_SWAP_SEL_16BPP,         // SWAP_SEL
                LCDC_PCR_END_SEL_LITTLE_ENDIAN,  // END_SEL
                LCDC_PCR_SCLKIDLE_ENABLE,        // SCLKIDLE
                LCDC_PCR_OEPOL_ACTIVE_HIGH,      // OEPOL
                LCDC_PCR_CLKPOL_NEG_EDGE,        // CLKPOL
                LCDC_PCR_LPPOL_ACTIVE_LOW,       // LPPOL (HSYNC POLARITY)
                LCDC_PCR_FLMPOL_ACTIVE_LOW,      // FLMPOL (VSYNC POLARITY)
                LCDC_PCR_PIXPOL_ACTIVE_HIGH,     // PIXPOL
                0,
                LCDC_PCR_PBSIZ_1BIT,             //LCDC_PCR_PBSIZ_8BIT
                LCDC_PCR_COLOR_COLOR,            //LCDC_PCR_COLOR_COLOR
                LCDC_PCR_TFT_ACTIVE              //LCDC_PCR_TFT_ACTIVE
        }
},

. . . .
}
```

5.9 Writing Initialization Command Routine

The panels described in this application note do not require any serial initialization command routine. If required, the code must be called during `BSPTurnOnDisplay()` execution. Ensure that the CSPI interface is already configured. The initialization routine is provided by the LCD vendor.

```
bsplcdc.cpp
void BSPTurnOnDisplay(PLCDC_MODE_DESCRIPTOR pModeDesc)
{
// LCD serial initialization routine goes here...
. . . .
TurnOnLCD(pModeDesc);
}
```

5.10 Rebuilding the Project

In the end, rebuild the project by using Build current BSP and subprojects. Menu Build > Advanced Build Commands > Build Current BSP and Subprojects.

6 Summary and Tips

6.1 Using LVDS with the i.MX25

i.MX25 is designed to support TFT panels providing the synchronization signals and data for the RGB interface (HSYNC, VSYNC, DE, PIXCLK, LD[17:0]). LVDS requires a serialized stream where the RGB pixel data and LCD timing are encapsulated. Using an external circuit for which PLL base would be the PIXCLK, the RGB data and timing could be serialized. This works for 12bpp and 16bpp but for 2bpp, 4bpp, 8bpp, 18bpp and 24bpp, LSCLK skips one cycle of every DE rising, every VSYNC rising and every VSYNC falling. These glitches do not affect TFT panels because during this time all signals are ignored for the TFT LCD panel but causes LVDS serialization to fail. Some customers have reported that they have got positive results by using CLKO instead of PIXCLK as a PLL. It is important to mention that this scenario is not supported by the i.MX25, the customer must not expect optimum results, and this is not recommended. The chip is not designed considering this environment and Freescale does not warranty the LCDC behavior under these circumstances or provide/ensure the maximum timing difference between CLKO and LSCLK.

6.2 Frame Rate

Depending on the pixel clock, back porch, front porch, and memory interface, the frame rate may decrease to less than 60 fps. For example, on WINCE, the microprocessor creates the frames related with the WINCE desktop and places the images on memory. In this use case, incoming frames come from memory display buffer. An 800 x 600 frame is periodically created and the LCDC sends this information to VGA, WVGA or SVGA LCD. LCDC sends the complete buffer in a time greater than the HORIZONTAL_RESOLUTION x VERTICAL_RESOLUTION x PIXCLK period. For example, for an SVGA panel, the result is 800 x 600 x PIXCLK period (usually 30 ns). This implies one frame for every 14.43 ms, or 69 fps, but considering the front and back porch for horizontal and vertical adjustment, it is probable that frame rate decreases to less than 60 fps.

6.3 Memory Access

i.MX25 can create WINCE desktop frames, which requires a lot of memory transfers. For example, in a normal SVGA application in the i.MX25 PDK, where the memory interface is a Double Date Rate (DDR) operating at 133 MHz, the maximum memory bandwidth is 266,000,000 x 16 bits (266, because DDR latches data on rising and falling edges). Even when there is large bandwidth, LCDC shares the memory with the processor and all the peripherals included in the i.MX25.

For WINCE desktop, calculate the average of percentage of memory that display interface uses to maintain the SVGA interface.

Memory Bandwidth = 266,000,000 x 16 bits = 4,256,000,000 bits per second **Eqn. 3**

SVGA frame write access = 800 x 600 pixels x 16 bits (RGB565) x 60 fps= 460,800,000 **Eqn. 4**

SVGA frame read access = 800 x 600 pixels x 16 bits (RGB565) x 60 fps= 460,800,000 **Eqn. 5**

Percentage of memory used by the LCDC = (460,800,000 x 2) / 4,256,000,000 = 21.66 % **Eqn. 6**

460,800,000 is multiplied by two as it is needed to create the frame (write) first and then read the contents from the buffer and send them to the LCDC.

The number calculated above is only for LCDC. If there is a post-processing stage (CSC, resize, alpha blending, and so on) for the frames, another read/write related operation is performed with each frame. As a result, the display uses 43.31% of the memory bandwidth. Also, if Background (WINCE desktop) and Foreground (For example, camera) planes are used working at the same time, the usage will increase as well. The memory usage depends on the number of planes used and the number of times those frames are accessed.

7 References

For more information on WINCE600 Display Driver Development Concepts refer to the link below:

- http://msdn.microsoft.com/en-us/library/ee485877.aspx

8 Revision History

Table 21 provides a revision history for this application note.

Table 21. Document Revision History

Rev. Number	Date	Substantive Change(s)
0	01/2010	Initial release.

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: AN3977
Rev. 0
01/2010