Freescale Semiconductor
Application Note

Document Number: AN3978
Rev. 0, 04/2010

Different Display Configurations on the i.MX35 WinCE PDK

by *Multimedia Application Division*
Freescale Semiconductor, Inc.
Austin, TX

This application note provides information, considerations and steps in order to add or adapt a new panel to the Board Support Package (BSP) distribution for the i.MX35 PDK. This document provide details about the general panel information, and generalities of the display controller from the module. This document also describes the development process to adapt a new panel into the BSP package, considering the framework driver structure—provided by the operating system. This applications note assumes that the reader knows the platform builder packages, and is familiar with the WinCE™ (Windows CE) device driver concepts.

Contents

1 Introduction

As a multimedia processor, the i.MX35 supports several types of displays. Display devices are handled by a special module called image processing unit (IPU). The IPU also handles other graphic interfaces such as cameras and 2D graphics accelerators. All IPU sub modules are connected by a private DMA interface (IDMA), used only by the IPU to transfer data among the sub modules and between the IPU and external memory.

freescale semiconductor

Figure 1 shows the functional diagram of the IPU.

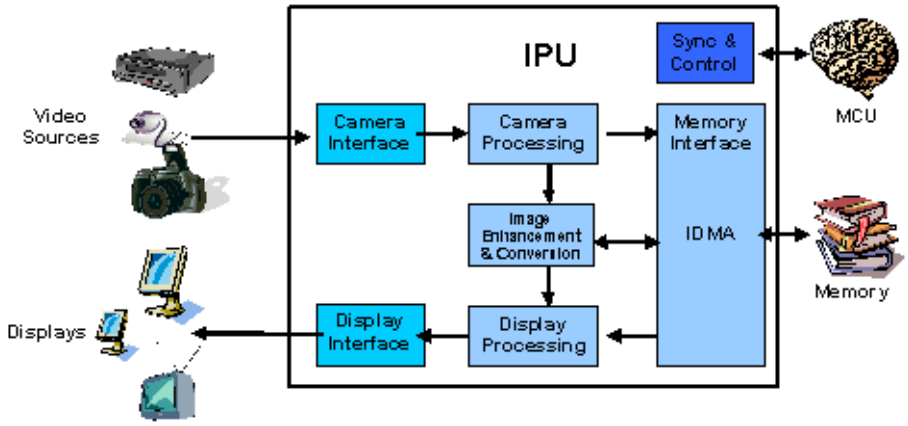

Figure 1. IPU Functional Diagram

The selection of a proper LCD in a mobile device is a challenge that often involves conflicts in the requirements. Some of the conflicting requirements 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 the LCD is demonstrated. But most developers 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, and it is hard to find the same LCD panel included in the reference design when a product is created.

While some of the information would be useful for Smart displays, this application note is intended only for dumb displays, especially the displays that do not have the Sharp synchronous interface.

NOTE

Do not confuse the Sharp LCDs with the Sharp interface, as there are plenty of Sharp panels that do not use the Sharp interface.

2 LCD Principles

This section describes the principles of LCD.

2.1 LCD Basics

Liquid crystal displays (LCD) are electronic devices, composed of array of pixels that are either color or monochrome. Every element in the array is created with a special material that allows them to change the

characteristics of the light passing through them. As these devices are not able to emit light, another element called backlight is shipped with the panel, to create a fully functional display device.

2.1.1 Resolution

For this application note, the term resolution is used as the number of pixels contained in the LCD array. It has two dimensions—horizontal and vertical. An ample amount of knowledge about video resolution standards is required, while creating a custom LCD array.

Some of the most common video resolution standards are shown in Table 1.

Table 1. Video Resolutions

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)	768	576	4:3
WVGA	Wide VGA	800	480	5:3
SVGA	Super VGA	800	600	4:3

There are many other standards resolutions, but since SVGA is the maximum standard resolution supported by the i.MX35, greater resolutions are not included in the table.

All resolutions mentioned in Table 1 refer to landscape orientation, in which the pixels in the horizontal axis is higher than that in the vertical axis. There are also portrait LCD panels, with the same standard resolution, but with the horizontal and vertical dimensions inverted.

Figure 2 shows portrait and landscape orientations.

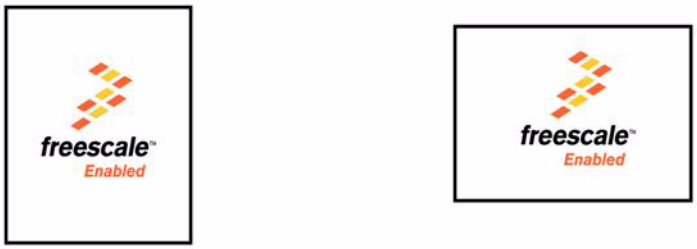

Figure 2. Portrait and Landscape Orientation

It is very important to select the orientation of the LCD, 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 (a chip) that draws the pixels from left to right and top to bottom. In scenarios, where images or videos are to be shown in the LCD, using a nonnative orientation, the display contents are pre-processed to create a buffer. The image is written on the buffer,

with the purpose of matching the way (order) the LCD controller expects the pixel information to be sent to it. This operation is called rotation, and the i.MX35 includes hardware to perform it.

Figure 3 shows both portrait and landscape LCD panels, displaying images in the nonnative orientation. The LCD rotation could be 90°, 180° or 270°. Note that it is needed to rotate every frame, before sending it to the display.

Figure 3. Rotated Frames on Portrait and Landscape Orientation

NOTE

It is recommendable to select an LCD panel, which works most of the time in its native orientation, to avoid extra image processing.

2.1.2 Size

The size of the LCD is measured diagonally, in inches, from corner to corner. When choosing between a VGA (640 × 480) panel and QVGA (320 × 240), it is expected for the VGA panel to have a larger size, since it has more pixels (four times as many). But this is not always the case. LCD manufacturing processes allow size and resolution to be independent variables. So it is hard to determine the size of a panel from its resolution alone. Screens that are lager in size tends to consume more power than the smaller ones, and they impact the size and weight of the final product. On the other hand, high resolutions on small LCD screens can complicate the visibility on screen objects. So, sometimes it is hard to imagine how well a particular LCD panel fits on the application, based only on the information in its datasheet.

2.1.3 Color Spaces

A color space is a way to represent colors. There are two main color spaces—RGB (that is RGB565, RGB888, RGBA8888) and YUV (that is YUV 4:4:4, YUV 4:2:2, YUV 4:2:0), and the i.MX35 supports both. But, the available display panels can receive data only using the RGB interface.

2.2 LCD Types

This section describes the various LCD types.

2.2.1 Synchronous Panel (Dumb Display)

Dumb display or synchronous display stands for panels that require the microprocessor to send all the pixels array image data, for each frame. For every single frame, the refresh is done by sending all the pixels that compose a full frame. In general, Smart displays are more expensive than dumb displays, and that is

one of the reasons why it is more common to use synchronous panels in a final product. This application note is focused on TFT (thin film transistor) liquid crystal display, which is a special group of synchronous panels.

2.2.2 Asynchronous Panel (Smart Displays)

The advantage of Smart displays is that the i.MX35 has to send only the display data, when the image is changed, and most of the times, it needs to send only the portion that has changed. Images can be sent at any time. The screen refresh is handled by the embedded Smart LCD display controller. The i.MX35 can handle up to three asynchronous displays simultaneously, and can handle the asynchronous interface at the same time. This means that for applications, which need two LCD panels, at least one of them must use an asynchronous interface.

3 Synchronous Display Interface

The i.MX35 synchronous display controller can be configured to handle the following devices:

- TFT monochrome
- TFT color
- YUV progressive
- YUV interlaced

This document focuses only on the synchronous TFT color interface. In this example, the i.MX35 provides a 28-line interface, which is described in Table 2.

Table 2. Display Interface Signals

Signal	IPU Signal	Description
HSYNC	DISPB_D3_HSYNC	Horizontal synchronization
VSYNC	DISPB_D3_VSYNC	Vertical synchronization
DRDY	DISPB_D3_DRDY	Data enable or Data ready
PIXCLK	DISPB_D3_CLK	Pixel clock
Red Data[7:0]	DISPB_DATA[23:16]	Pixel red component
Green Data[7:0]	DISPB_DATA[15:8]	Pixel green component
Blue Data[7:0]	DISPB_DATA[7:0]	Pixel blue component

The signals and interface in this display system are as follows:

HSYNC — Horizontal synchronization signal, also known as FPLINE or LP, indicates the LCD that a line has ended, and the following valid pixels are a part of the next line.

VSYNC — Vertical synchronization signal is also known as FPFRAME, FLM, SPS or TV. When this signal is active, it 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 into the first line of the panel.

DRDY	When data ready (DRDY) or data enable (DE) is active, it indicates the LCD that the data in the RGB bus is valid, and must be latched (latching happens using the PIXCLK signal). While data enable is active, every PIXCLK pulse makes the LCD draw a pixel using the color described in the RGB bus. The width of this signal should store all the pixels (that is, as long as the sum of all pixel clock cycles are in a single line).
PIXCLK	Pixel clock signal specifies when the RGB data has to be placed on the bus. There are two possibilities. The first option is when data is written by the i.MX35 to RGB bus on the falling edges. In this case, data is stable, and ready to be latched by the LCD panel on the rising edges, as long as data enable is active. This behavior corresponds to a high PIXCLK polarity. The second option is called low PIXCLK polarity, and it means that the i.MX35 writes RGB data during the rising edges, and the data is latched by the LCD panel during the falling edges.
RGB Data	The i.MX35 can internally use different bit depths per pixel, such as RGB565, RGB666, RGB888, RGBA8888, and so on. In the same manner, the display interface could be configured to support more than one color depth. The i.MX35 processors can use up to 24 data lines (RGB888) as display interface bus. If the color depth used internally is bigger than the display interface, then an RGB to RGB conversion is performed, where the least significant bits are removed from the pixel, and the remaining bits are send directly to the display interface. Dithering or filtering actions are not performed during this process. The remaining RGB data lines can be used for other purposes, including GPIO (general purpose input/output).
Extra signals	There are also some other lines that are included in the panel interface. These signals are not part of the 28-line display interface, but they are required for a full functional module. For example, it is common for some panels to have a reset signal, as well as initialization commands, and these commands are usually sent by a serial interface such as I2C or SPI (serial peripheral interface). Similarly, display panels sometimes have touch panels and backlight unit embedded in them, that require additional signals.
SPI Interface	Some LCD displays require an initialization routine through a serial interface—3-wire, 4-wire, or 5-wire. Even when the i.MX35 IPU has a serial interface (SD_D_CLK, LCS1, SD_D_IO and SD_D_I), it should not be used to send serial commands to the LCDs. This interface is not intended for general purpose usage, instead, it is used only by the IPU, when the asynchronous display1 or display2 are configured to use the serial interface.

3.1 Examples of Synchronous Display Interfaces

This section explains some of the synchronous display interfaces.

3.1.1 i.MX35 PDK Chunghwa 5.7" VGA LCD Interface

Figure 4 shows the LCD interface between the i.MX35 and Chunghwa CLAA057VA01CTVGA panel.

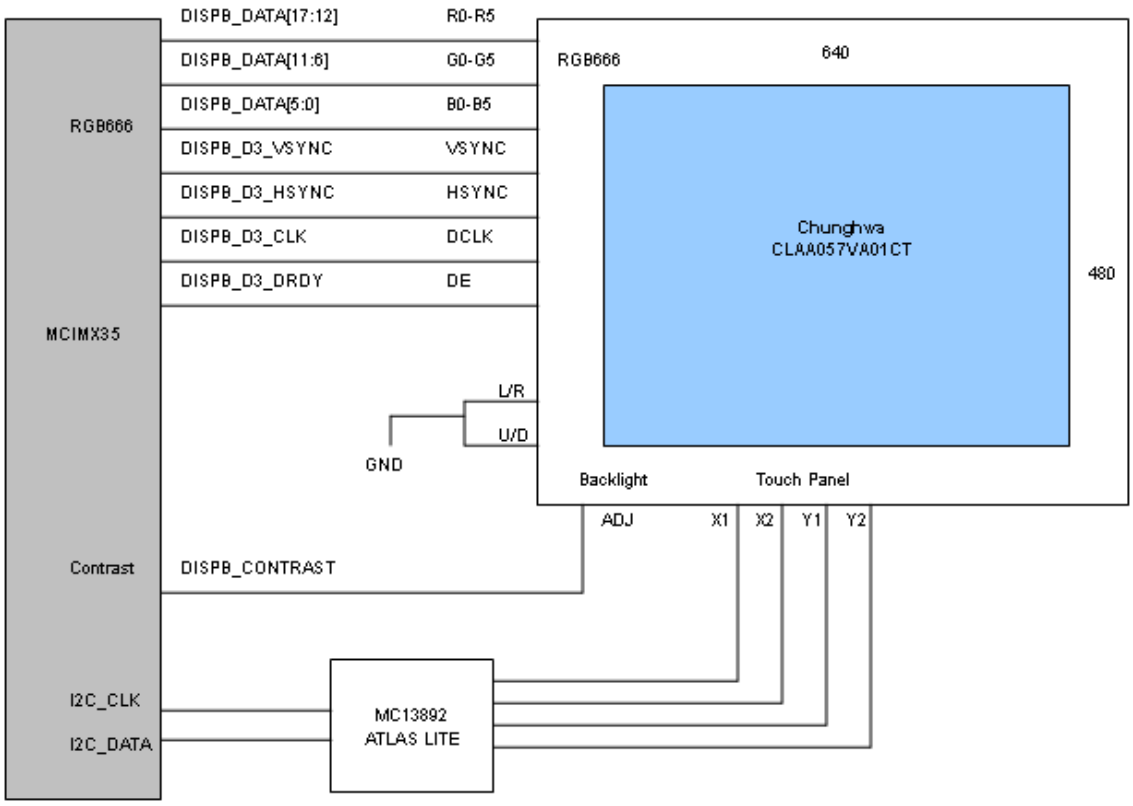

Figure 4. LCD Interface between the i.MX35 and Chunghwa CLAA057VA01CTVGA Panel

As shown in Figure 4, the LCD panel requires HSYNC, VSYNC, DE, PIXCLK, and the complete RGB data interface (DISPB_DATA[17:0]). But no additional signals such as reset signal or serial interface initialization routine commands (SPI or I2C) are required. The backlight unit is controlled by a PWM (pulse width modulation) signal, generated by the i.MX35 (contrast). Also, the touch panel interface is handled by the MC13892 ATLAS LITE chip.

In general, Figure 4 illustrates a typical synchronous panel interface. For this application note, it is taken as the base for the panel interface. The idea of the base interface is useful when there are many panels that do not use the complete interface. For example, some of the panels do not require HSYNC or VSYNC signal. Only DRDY, PIXCLK, and RGB data may be used. Also many others panels do not need a reset or a serial initialization routine, to handle display signals. So, when there is no serial interface, timing, signals, porches, and polarities are specified, the panels expect the microprocessors to comply with these waveforms, as there is no way for such panels to handle different interfaces.

3.1.2 i.MX35 PDK Chunghwa CLAA070VC01 7" WVGA LCD Interface

Figure 5 shows the LCD interface between the i.MX35 and Chunghwa CLAA070VC01 WVGA panel.

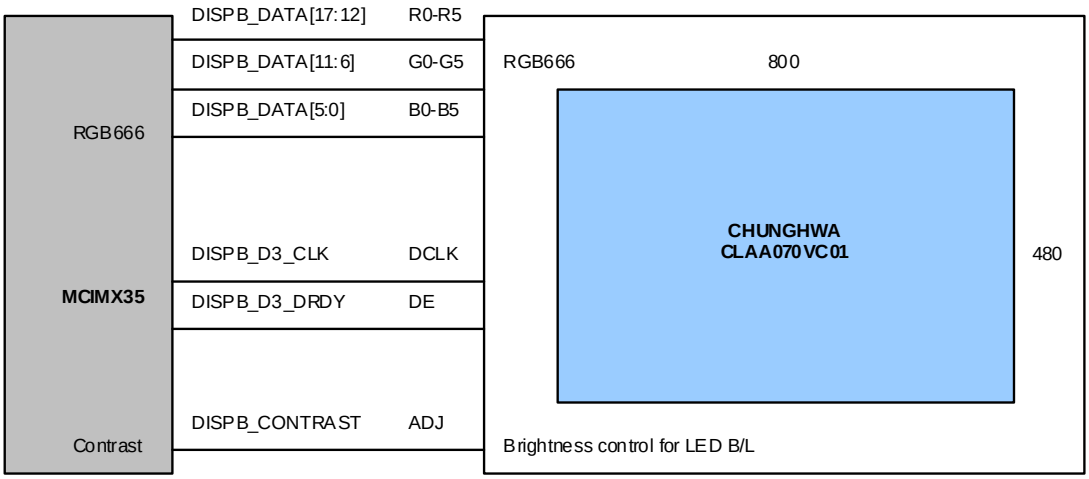

Figure 5. LCD Interface between the i.MX35 and Chunghwa CLAA070VC01 WVGA Panel

This LCD is shipped with the i.MX35 PDK, and Figure 5 shows a very simple display interface, where HSYNC and VSYNC signals are not used. Since they are not used, the pins DISPB_D3_VSYNC and DISPB_D3_HSYNC can be used as GPIO or be used for any other alternate functions. Also the SPI interface is not required, and so the chip select (CSPI1_SS2 pin) is available for other devices. Additionally, the power booster for the backlight unit is included in the module, which means that the CONTRAST signal is connected directly to the display connector.

In this example and also in the previous one (Section 3.1.1, “i.MX35 PDK Chunghwa 5.7" VGA LCD Interface,”) only 18 RGB data is required, that is DISPB_DATA[17:0]. Hence, all the remaining RGB data lines, that is DISPB_DATA[23:18], can be used for other purposes, such as a GPIO or for any other alternate functions.

Apart from the advantages mentioned above, it has certain disadvantages, and are listed as follows:

- This display module does not include a touch panel. So it necessitate an external touch screen for the LCD panel.
- It does not have a reset signal, or SPI interface. So there is no way to turn off the display—This feature is important for mobile devices, where power consumption is an issue. If the energy used by the LCD need to be controlled, external circuits are required for the same. In the case of the PDK, power ON/OFF settings are handled by an external 8-bit microprocessor (MC9S08DZ60), driven by the i.MX35, through an I2C bus.

Based on the above observations, the complete LCD circuit for the i.MX35 PDK system is shown in Figure 6.

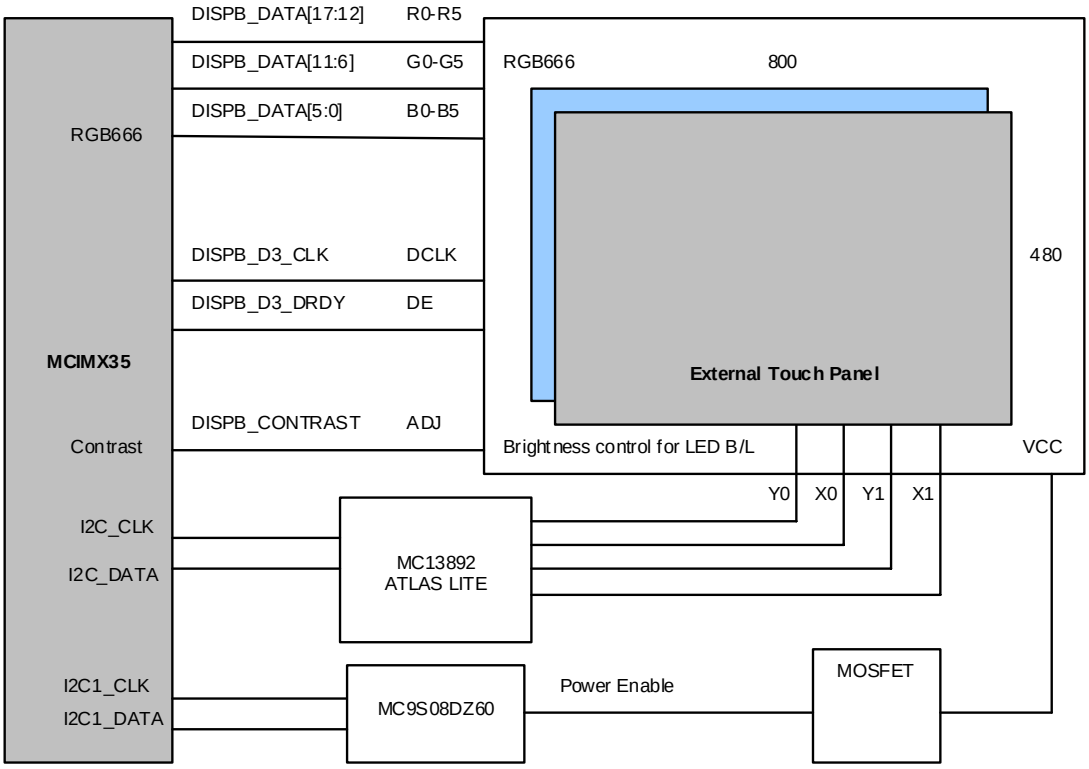

Figure 6. LCD Interface between the i.MX35 and Chunghwa CLAA070VC01VWGA Panel with Touch Panel

The previous examples might be helpful, when selecting an LCD display, to determine the advantages and disadvantages of each different types of panel.

3.2 Synchronous Display Timing and Signals

This section focusses on the timing and signal waveforms, and how to configure these in the LCD panel and i.MX35 display interface. The first step in selecting an LCD module is to refer to its datasheet. The datasheet must show the pin interface, the initialization routine, and the timing charts for the RGB interface, and also the serial interface. Sometimes, a shorter version of the datasheet is also received, which may not contain all the information. In such situations, request the 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

The timing concepts that form the basis for LCD timing are explained below:

Horizontal Back Porch (HBP) Number of PIXCLK between the HSYNC signal, when asserted, and the first valid pixel data.

Horizontal Front Porch (HFP)	Number of PIXCLK between the last valid pixel data in the line, and the next HSYNC pulse.
Vertical Back Porch (VBP)	Number of lines (HSYNC pulses) between the VSYNC signal, since asserted, and the first valid line.
Vertical Front Porch (VFP)	Number of lines (HSYNC pulses) between the last valid line of the frame, and the next VSYNC pulse.
VSYNC pulse width	Number of HSYNC pulses that the VSYNC signal is active.
HSYNC pulse width	Number of PIXCLK pulses that the HSYNC signal is active.
Active Frame Width	This value is equal to the horizontal resolution, which means the number of pixels in one line. For example, for a WVGA display (800H × 480V), the 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 (800H × 480V), the value of the frame height is equal to 480 lines.
Screen Width	Number of pixel clock periods between the last HSYNC and new HSYNC. So, this value not only includes the valid pixels, but also the horizontal back and front porches.

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

Screen Height	Number of rows between the last VSYNC pulse and new VSYNC pulse. It includes all the valid lines, and also the vertical back and front porch.

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

VSYNC polarity	It is the value that VSYNC takes, to indicate the starting of a new frame. It is active low when its value is 0, and active high when it is 1.
HSYNC polarity	It is the value that the HSYNC takes to indicate the starting of a new line. It is active low when its value is 0, and active high when it is 1.

3.2.2 Timing Charts

The following charts are reviewed to clarify the timing issues in an LCD interface. All datasheets include this information. In general, three different charts are available as given below:

- The first one covers the vertical timing
- The second one specifies the horizontal timing
- The third one covers the pixel clock characteristics

Additionally, if the display uses a serial interface, then a chart describing the serial interface and reset is available. This information is extracted from the datasheet when support for a new LCD panel is added to a BSP. For example, when a VGA (640H × 480V) LCD panel is used instead of a Chunghwa CLAA057VA01CT (see Figure 5), then the display uses the complete RGB interface—RGB666, VSYNC, HSYNC, data enable and pixel clock.

3.2.2.1 Vertical Timing

Vertical timing can be categorized as VGA vertical timing and WVGA vertical timing. These vertical timings are discussed as follows.

VGA Vertical Timing

Figure 7 shows the vertical timing for a hypothetical synchronous display VGA (640H × 480V).

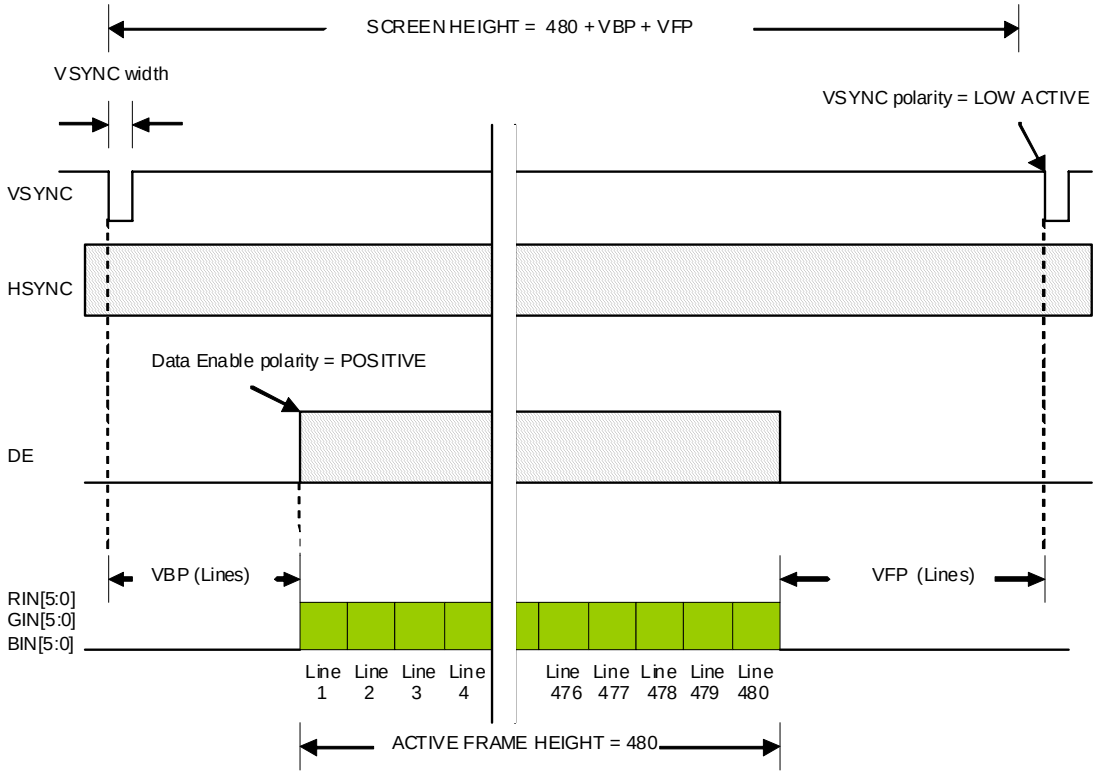

Figure 7. VGA Vertical Timing Example

It is important to mention how signals appear during the VSYNC period. VSYNC period involves a complete frame cycle. During this cycle, every pixel, and line in the frame are sent to the panel. The beginning of the frame is asserted 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 few lines are designated for the vertical back porch (VBP). During VBP, data enable (DE) signal is not present, and the pixel data on the bus during these lines are ignored by the panel. After VBP, DE 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 vertical front porch (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 the VSYNC signal is asserted again (goes low).

Along with this chart, a table showing the range of the timing parameters are can be seen, as shown in Table 3.

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 in the Table 3, following timing features are verified:

- In Figure 7, it is shown that the VSYNC polarity is active low, which means that vertical synchronization is normally high, but goes low to indicate the beginning of the new frame.
- As shown in Figure 7, VSYNC width (VSW) timing has certain flexibility. So the timing can be set using more than one value. It is highly recommended to use the typical values, or any values close to them. So, 1 line is used as VSYNC width.
- VSYNC width is included into the VBP stage. This means that VBP (vertical back porch) starts when VSYNC is asserted, and not when the VSYNC returns to normal state.

VBP and VFP are measured in lines or which translates into HSYNC pulses. In this example, VBP is 34 lines, and VFP is 11 lines width. Using these values, the screen height or vertical cycle is 525. In some cases, the values of the VBP and VFP are not given in lines, instead in nanoseconds or milliseconds. In that case, additional calculations are performed to find the number of lines needed to meet those timings.

WVGA Vertical Timing

If an LCD panel like the hypothetical WVGA (800H × 480V), described in Figure 5 and Figure 6 is used, where the HSYNC and VSYNC signals are not used, the waveforms are analyzed using another perspective. The chart so formed is shown in Figure 8.

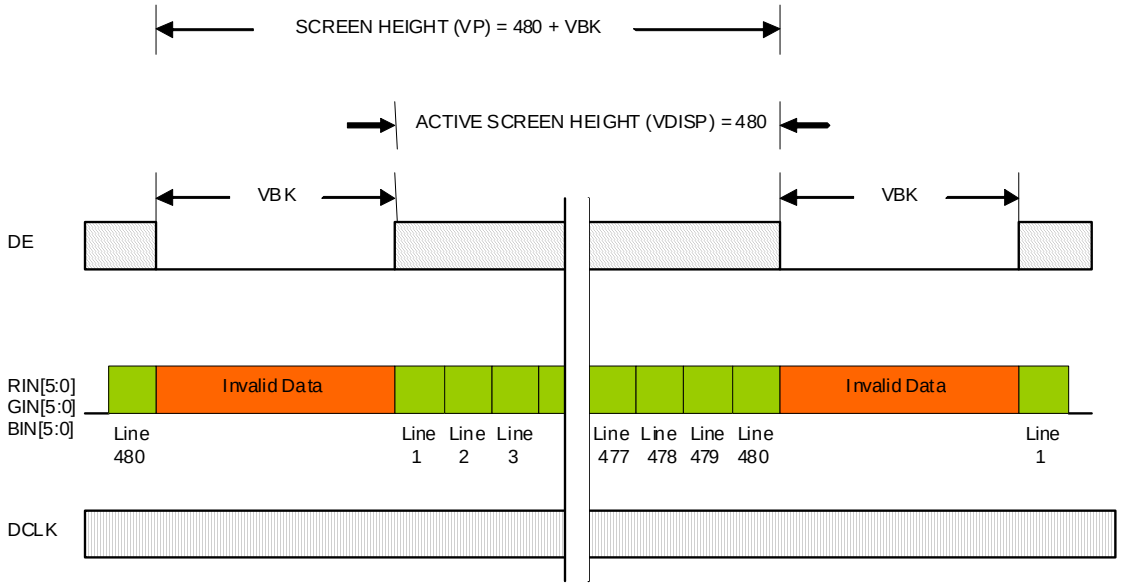

Figure 8. WVGA Vertical Timing Example

Table 4 shows the range of timing parameters.

Table 4. WVGA Vertical Timing

Parameter	Symbol	Min	Typ	Max	Unit
Screen height or Vertical cycle	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, VBP, and VFP are not given in the chart. Even when VSYNC is not used, these values are required, while configuring the i.MX35 display interface. This waveform can be used to understand the vertical cycle behavior. For the i.MX35, the sequence remains the same—vertical cycle starts with the VSYNC signal, and then the rest of the VBP follows, then the active frame area, and finally the VFP appears, until the next VSYNC is asserted. The VSYNC width, VBP, and VFP can be found out, based on the events occurring during the VBP.

Figure 9 shows an example of WVGA vertical timing.

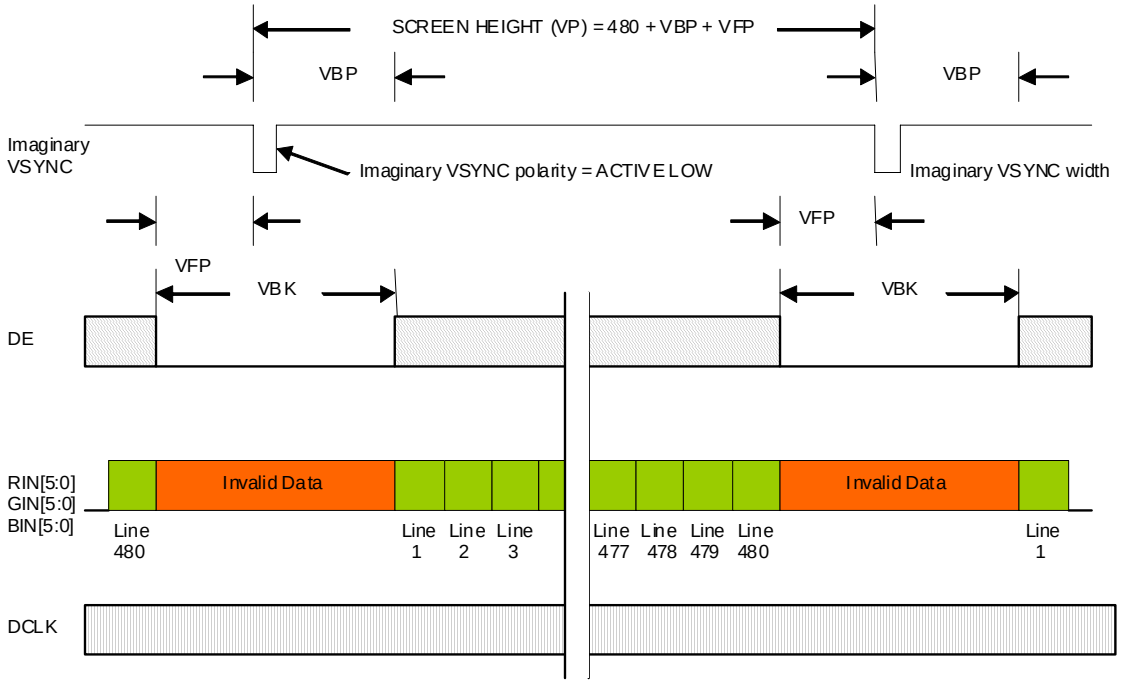

Figure 9. WVGA Vertical Timing Example with Imaginary VSYNC Signal

The VSYNC signal can be used as a base to calculate the VFP and VBP. Since VSYNC is not used, any polarity can be set. However, setting VSYNC as active low is recommended. VSYNC is usually one line long, and thus the value of VSW is 1. Now, to find VBP and VFP, it is important to note that the goal is to meet the vertical blank period. So the VBK period can be split into two parts—the first part for the VFP, before VSYNC is asserted, and the other part for the VBP, including VSYNC. The sum of these values must be equal to the VBK period. It is recommended to leave an imaginary VSYNC in the middle of the blank period. This means that both VBP and VFP should be equal or almost equal.

Consider that the VBK is 20 lines (typical). Then VBP should be 10 lines, which is equal to VFP. Based on these information, vertical timing table is created as shown in Table 5.

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

Horizontal timing can be categorized as VGA horizontal timing and WVGA horizontal timing. These horizontal timing are discussed as follows.

VGA Horizontal Timing

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

Figure 10 shows an example of line period datasheet chart.

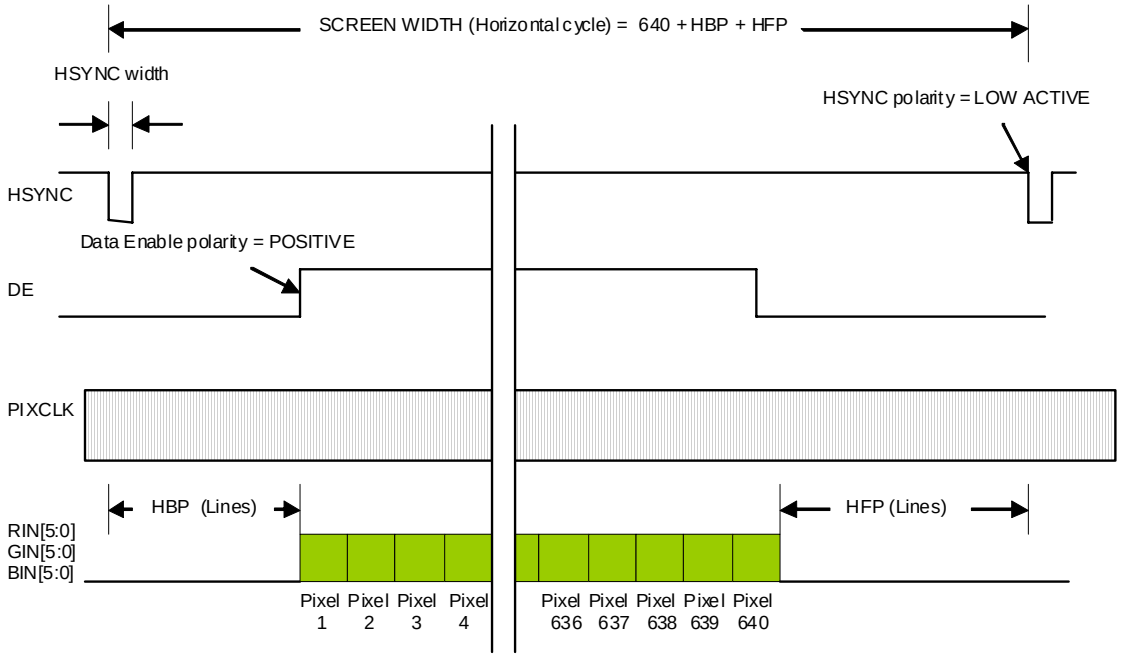

Figure 10. VGA Horizontal Timing Example

The line cycle begins when the HSYNC is asserted. Here, when the signal goes low, the HBP stage continues. During this time, the data enable signal remains inactive. When data enable is asserted (since DE is active high, it is activated, when the DE signal goes high), the horizontal active area (ACTIVE FRAME WIDTH) begins. Now, for every pixel clock pulse, the panel latches the RGB data placed in the bus, and draws a new pixel on the screen in the current line. The time for which DE remains active depends on the horizontal resolution of the panel, as DE width is always equal to the panel horizontal resolution. In this example, DE is 640 pixels long. When DE is inactive, the HFP occurs, and all the pixels in the line are drawn. The line cycle ends when the new HSYNC pulse is asserted.

Similar to the vertical timing, Table 6 shows the horizontal timing characteristics (PIXCLK refers to pixel clock pulses)

Table 6. VGA Horizontal Timing

Parameter	Symbol	Min	Typ	Max	Unit
Screen width or Horizontal cycle	HP	750	800	900	PIXCLK
HSYNC pulse width	HSW	1	1	1	PIXCLK
Horizontal back porch	HBP	46	46	46	PIXCLK
Horizontal front porch	HFP	64	114	214	PIXCLK
Active frame width	HDISP	—	640	—	PIXCLK

WVGA Horizontal Timing

The chart and table available for the WVGA (800H × 480V) datasheet is similar to the one shown in Figure 11 and Table 7 respectively.

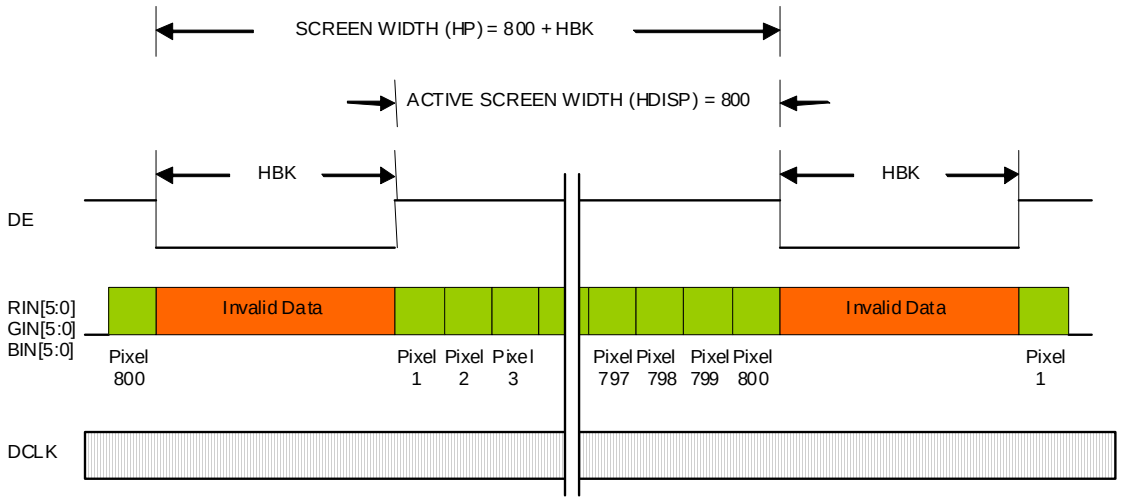

Figure 11. WVGA Horizontal Timing Example

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 approach followed in the WVGA vertical timing section is followed here to calculate HYNC width, HBP, and HFP. See Section , “WVGA Vertical Timing,” for information on WVGA vertical timing.

The horizontal timing diagram and the table using an imaginary HSYNC signal are shown in Figure 12 and Table 8 respectively.

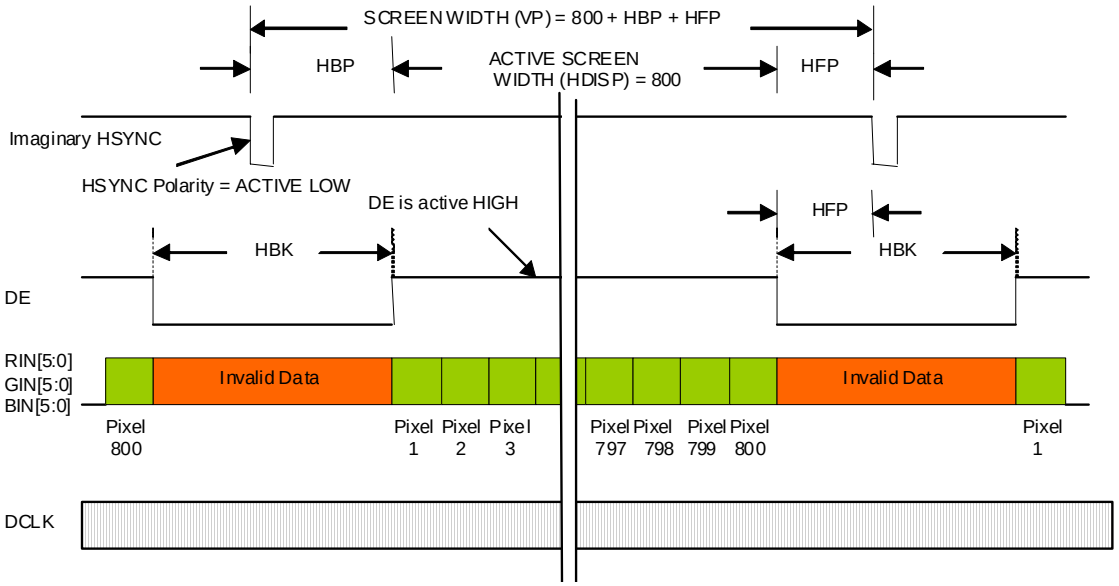

Figure 12. WVGA Vertical Timing Example with Imaginary HSYNC Signals

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 VGA, WVGA pixel clock timings, and the data polarity features are explained as follows.

VGA Pixel Clock Timing

The chart and table of the pixel clock timing characteristics in the datasheet is similar to the one shown in Figure 13 and Table 9 respectively.

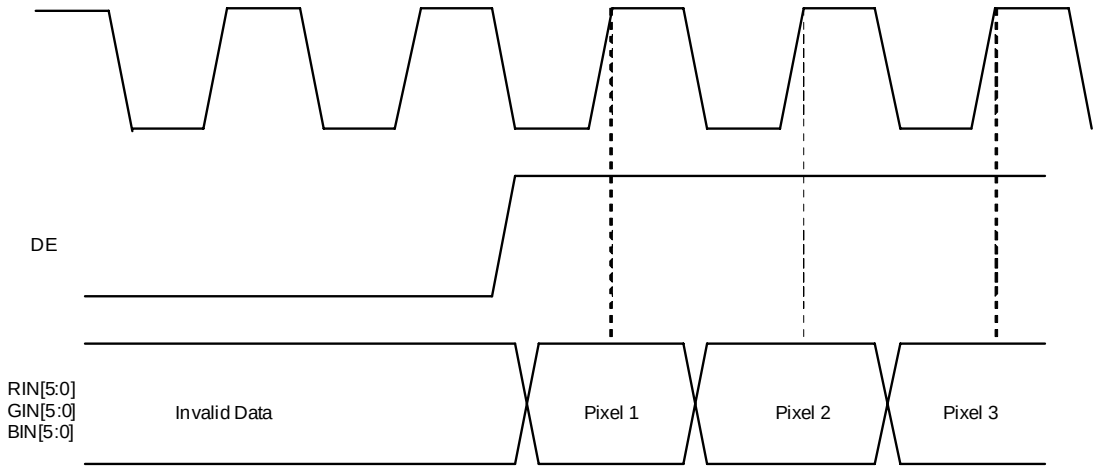

Figure 13. VGA Pixel Clock Timing Example

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 the RGB data is latched by the panel. This characteristic is very important as the i.MX35 has to prepare the data, one edge before the LCD latches the data in the bus. In this example, data is latched by the LCD panel on DCLK rising edges. So the i.MX35 must be configured to write the RGB data to the bus, on the falling edge. In this manner, the data is made ready and stable when the panel reads it. The waveform in Figure 13 shows the typical inverse clock polarity. Clock polarity is set in DI_DISP_SIG_POL i.MX35 register, under the D3_CLK_POL bit field.

NOTE

The maximum display clock rate cannot be greater than a quarter of the high speed processing clock rate. Remember that HSP_CLK in the i.MX35 PDK BSP is 133 MHz. So the maximum pixel clock is 133 MHz / 4 = 33.25 MHz. However, most LCD displays are able to work with lower frequencies than the typical values.

WVGA Pixel Clock Timing

The waveform characteristics chart and table for a WVGA pixel clock is shown in Figure 14 and Table 10 respectively.

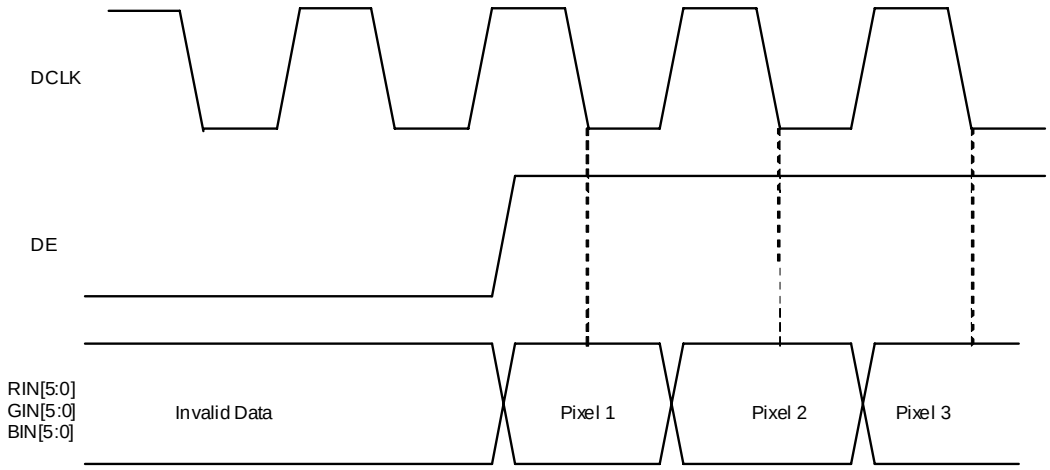

Figure 14. WVGA Pixel Clock Timing Example

Table 10. WVGA Pixel Clock Timing

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

In contrast to the VGA panel, the WVGA display latches RGB data on the falling edges of DCLK. So the i.MX35 should be configured to write the RGB data to the bus, on the rising edge. In this manner, the data is ready and stable, when the panel reads it. The waveform in Figure 14 shows the straight clock polarity.

Data Polarity

Pixel polarity is the polarity of the signals in the RGB bus that an LCD recognizes as active. For example, consider that the i.MX35 is trying to draw a red pixel (only red component), using an RGB565 interface, and the LCD using straight polarity. Then the value in the bus would be 0xF800, which means that all the RGB bits are high and the rest of the bits are low. However, if the LCD utilizes inverse data polarity, the value would be 0x07FF, which means that all the red bits are low, and the other bits are high. Both the values represent red color, and the difference in the value is caused by the data polarity on the LCD panel. This feature is configured using the D3_DATA_POL bit field in the DI_DISP_SIG_POL i.MX35 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, then the following charts (Figure 15 and Figure 16) are found.

3.2.3.1 Reset

The reset and serial command interface are explained in the following sections.

Reset

Many LCD panels include an LCD controller, which requires an external system reset. If the LCD mentions the signal usage, then finding the timing required for this pulse is useful.

Reset signal and its timing is shown in Figure 15 and Table 11 respectively.

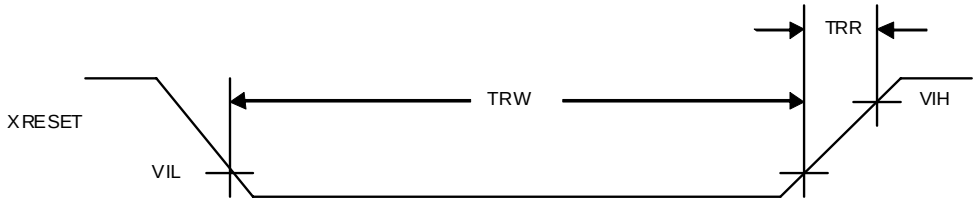

Figure 15. Reset Signal

Table 11. Reset Timing

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

Based on the chart, the following important facts are observed:

- Reset signal is active low. It means that the reset signal must be high during normal operation, and low when activated.
- Reset must be low for at least 15 ns to ensure a valid reset.
- The rise time of the signal should be 10 ns. For this reason it is not recommended to use an RC circuit to provide this signal. Usually this pin is controlled by the i.MX35 GPIO.

Serial Command Interface

In case the LCD panel has a serial command interface, then a chart similar to the one shown in Figure 16, is included in the datasheet.

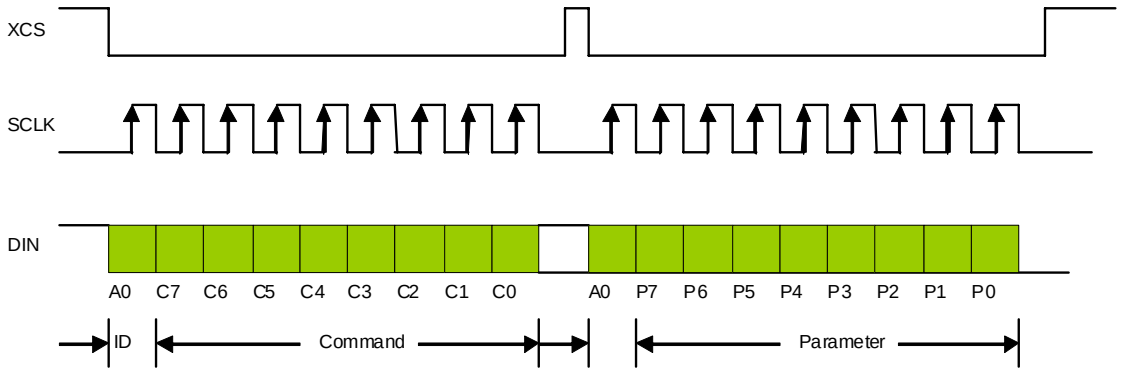

Figure 16. SPI Command Interface Signals

NOTE

Though it is important to understand how to initialize the LCD, this application note does not review all the serial interfaces that an LCD has. The protocols and data format are described in the datasheet. The user should have prior knowledge in synchronous serial interfaces, to program these settings. See the chapter, Configurable Serial Peripheral Interface (CSPI) in *i.MX35 (MCIMX35) Multimedia Applications Processor Reference Manual* (IMX35RM), for more information.

3.3 LCD Panels Supported by the i.MX35

The i.MX35 processor supports up to four displays simultaneously, that are handled by the display controllers DISP0, DISP1, DISP2, and DISP3.

Table 12 gives the various types of display interfaces.

Table 12. i.MX35 Display Interfaces

Display	Display Type	Interface
DISP0	Asynchronous	Parallel interface only
DISP1	Asynchronous	Serial and parallel interface
DISP2	Asynchronous	Serial and parallel interface
DISP3	Synchronous	RGB interface (HSYNC, VSYNC, PIXCLK, up to RGB888)

As shown in Table 12, only one of the LCD display controllers on the i.MX35 is synchronous (dumb display). So, this application note is focused on the DISP3 controller. Note that the DISP3 RGB interface is multiplexed with all the other asynchronous parallel interfaces. It means that the data cannot be sent to the synchronous display (DISP3) and another parallel display device, at the same time. Instead, the

i.MX35 has to send data to the asynchronous panel (Smart display), while the synchronous interface is inactive (during horizontal and vertical back and front porches). For this reason, the frame rate of Smart displays could be affected, when multiple displays are attached to the i.MX35.

The synchronous LCD interface on the i.MX35 is very flexible. It can handle many types of LCD devices, as long as these devices have the following characteristics:

- Synchronous display (dumb display)
- RGB interface (RGB888 maximum)
- Resolution not larger than SVGA
- Utilize at least DE and pixel clock to latch RGB data (some LCDs need HSYNC and VSYNC signals as well, which are also supported by the i.MX35)
- Pixel clock frequency lower than 33.25 MHz

In addition, the i.MX35 can handle dumb displays with a Sharp interface (as this application note is only intended for non-sharp dumb displays, Smart displays and Sharp displays interfaces are not covered). But, its support is limited to certain models.

4 Display Configuration in WinCE 6.0

One of the most important features, for any multimedia device is display support. It enables the device to have a graphical user interface (GUI), and increases its possibility to become an entertainment artifact.

The graphic context of a device is composed of several layers, where the i.MX35 display interface is the final part in the abstraction. All the SDC (synchronous display controller) and display interface characteristics that were reviewed in the previous sections, describes only the way how the i.MX35 sends the frame buffer to the panel. However, it is very important to know who is going to create the frames that need to be sent to the panel. If the screen is refreshed 60 times per second (60 Hz), every line and every single pixel has to be created, to maintain the coherence of the graphic context.

For all the WINCE600 devices, the i.MX35 PDK BSP bases its display driver on the display driver interface (DDI), defined by the Microsoft™. By implementing a driver using this model, it ensures the compatibility of the hardware with the operating system. In other words, once WinCE is loaded, and if the driver is created using the Microsoft model, the OS handles the graphic context, providing all the frames.

4.1 WINCE600 Display Driver Development Concepts

Display drivers are loaded, and called directly by the graphics, windowing, and event subsystems, called Gwes.exe. Generally, the drivers are written using a layered architecture, because of the number of hardware independent operations. The graphics primitive engine (GPE) library handles the default drawing, thus 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, like a Hello World application, or complex, like a three-dimensional engineering application. The application calls the GDI functions. Coredll.dll exposes the GDI functions.
Coredll.dll	The major set of functions are exposed through a single DLL (dynamic link library), 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 an another process. The specific process depends on the function call. All drawing and windowing calls are send to Gwes.exe.
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 the 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 the rest 24 for the display driver developer.
Hardware	The graphic pipeline ends at the hardware. The display driver communicates to the hardware through the mechanism used by the hardware. This process typically involves a combination of memory-mapped video buffers and I/O registers.

Figure 17 shows the Windows CE graphics architecture.

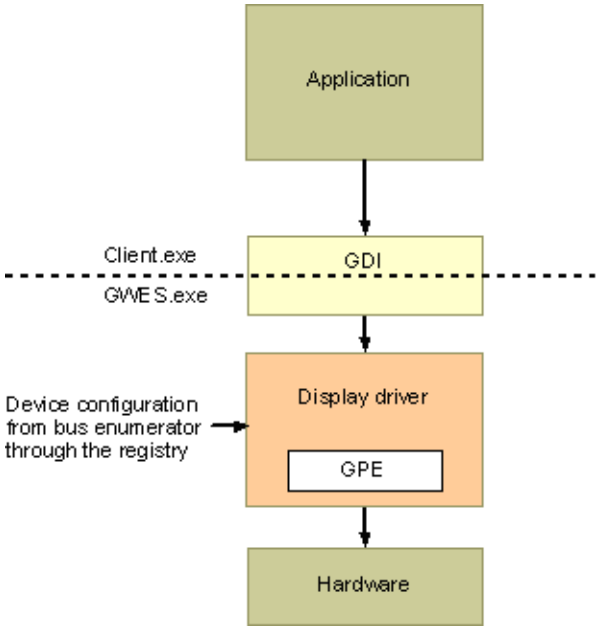

Figure 17. WinCE Graphics Architecture

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.

Developing a WinCE display driver from the scratch implies a considerable effort and knowledge. Hence, Freescale provides the i.MX35 display driver for synchronous displays, in the WINCE600 BSP. The i.MX35 Windows CE 6.0 BSP display driver is based on the Microsoft DirectDraw Graphics Primitive Engine (DDGPE) classes, and supports the Microsoft DirectDraw interface. This driver combines the functionality of a standard LCD display with DirectDraw support. The display driver interfaces with the IPU. This driver supports more than one panel, which could be selected, by using the windows register. The support for a new SDC panel can be easily added, using the procedure described in this 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 Identification of LCD Characteristics and Timing

To add the support for a new synchronous LCD display for the Freescale i.MX35 BSP, ensure that the panel is compatible with the i.MX35. 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 the data (some LCDs need HSYNC and VSYNC signals that are also supported by the i.MX51)
- Pixel clock frequency lower than 33.25 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	HSP	
VSYNC polarity	VSP	
DRDY (data enable) polarity	DEP	
Pixel clock polarity	CLKPOL	
Data polarity	DP	

More details to deduce these values from the LCD datasheet can be found in Section 1, “Introduction,” Section 2, “LCD Principles,” and Section 3, “Synchronous Display Interface.”

4.2.2 i.MX35 WINCE600 PDK LCD Driver Initialization Flow

The following snippet shows the i.MX35 WINCE600 PDK LCD driver initialization flow.

```
DDIPU::DDIPU()
+ DDIPU::Init
          + GetPanelTypeFromRegistry()
          + Set GPEMode
          + GetPixelDepthFromRegistry()
          + GetVideoPixelDepthFromRegistry()
          + GetTVModeSupportFromRegistry()
          + PPOpenHandle()
          + Create Dsiplay Events
          + Create Threads
+ DDIPU::SetupVideoMemory()
```

```
        + Allocate Video Memory Size comes from constant in image_cfg.h
+ DDIPUSurf::SetRotation()
+ DDIPU::InitHardware()
        + BSPInitializeLCD(eIPU_SDC)
                + BSPLCDPower(TRUE)
        + InitializeSDC()
                + DDKClockSetGatingMode()
                + Configure SDC and DI (IPU_CONF, SDC_COM_CONF, SDC_HOR_CONF,
                 SDC_VER_CONF, DI_DISP_IF_CONF, DI_DISP_SIG_POL, DI_HSP_CLK_PER,
                 DI_DISP3_TIME_CONF, SDC_BG_POS, SDC_FG_POS)
                + _init_dma(..,SDC_DMA_CHANNEL)
        + BackgroundSetSrcBuffer()
        + BSPEnableLCD(eIPU_SDC)
                + Turn on the LCD - Not included
                + Set up power voltages  - Not included
                + Reset the Module  - Not included
                + Configure CSPI registers and enable it.   - Not included
                + LCD_CSPI_Write() - SPI LCD Initialization - Not included
                + Disable CSPI interface                    - Not included
        + EnableSDC()
                + BSPDisplayIOMUXEnable(eIPU_SDC)
                        + Configure LCD pins :LD0-LD17, VSYNC, HSYNC, DRDY, CONTRAST, etc
                + DeviceIoControl(..,IPU_IOCTL_ENABLE_SDC,..)
                + DeviceIoControl(..,IPU_IOCTL_ENABLE_DI,..)
                + Enable DMA SDC Channel 1
```

LCD driver initialization begins with the `DDIPU::DDIPU()` function. Video memory size is not taken from the registry. The value is a constant (`IMAGE_WINCE_IPU_RAM_SIZE`), declared in `image_cfg.h`. Panel type, pixel depth, and TV modes, supported by the platform are extracted from the registry, `platform.reg`. Based on the selected panel, the driver notifies WinCE, the properties of the display, such as type (synchronous or asynchronous), width, height, bits per pixel, and so on. `GPEMode` data is automatically set, by using the `PANEL_INFO` structure of the current panel. With this information, WinCE graphic context creates the frame buffers in the width, height, and format required by the LCD, in order to display on the screen. The information regarding `GPEMode` is modified, to complete the support for the new LCD. The width and height is provided in the natural orientation of the LCD.

The next step is the hardware initialization. Here the IPU registers are configured to enable the SDC and display interface (DI), for working with the selected panel. LCD timing features are not stored in the register, but in the array `PANEL_INFO g_PanelArray[numPanel]`, placed in the `sdc.c` file. So, if support for a new panel has to be added, this array is updated by adding another `PANEL_INFO` structure, with the panel timing information. Later, the IDMAC SDC channels and IOMUX are configured, to enable the LCD pin interface (VSYNC, HSYNC, LD0-LD17, PIXCLK, DRDY). Then, using `bspdisplay.cpp`, the driver configures the specific LCD panel pins for initialization, such as SPI interface, reset, and other enable pins. And at the end of the process, the power levels of the LCD are enabled, the panel is reset, and all the serial initialization commands are sent to the LCD panel, in case it is needed.

4.2.3 i.MX35 WINCE600 PDK LCD Display Interface Related Files

The i.MX35 WINCE600 PDK LCD display interface related files are as follows:

- `WINCE600\PLATFORM\iMX35-3DS-PDK1_6\SRC\INC\sdc.h`
- `WINCE600\PLATFORM\iMX35-3DS-PDK1_6\SRC\DRIVERS\IPU\SDC\sdc.c`
- `WINCE600\PLATFORM\iMX35-3DS-PDK1_6\SRC\DRIVERS\IPU\DISPLAY\COMMON\ddipu.cpp`

- WINCE600\PLATFORM\COMMON\SRC\SOC\COMMON_FSL_V2_PDK1_6\IPU\INC\ipu.h
- WINCE600\PLATFORM\iMX35-3DS-PDK1_6\SRC\DRIVERS\IPU\DISPLAY\DLL\bspdisplay.cpp
- WINCE600\PLATFORM\iMX35-3DS-PDK1_6\FILES\platform.reg
- WINCE600\PLATFORM\iMX35-3DS-PDK1_6\FILES\platform.bib

4.2.4 i.MX35 PDK LCD Structures

It is very important to understand, where data related to the LCD panels settings (see Section 4.2.1, "Identification of LCD Characteristics and Timing,") are stored, as the communication channel is used to inform the synchronous LCD driver, about the settings for the new LCD display. The PANEL_INFO stores the information of the synchronous, asynchronous, and TV displays. This structure contains ADC_IPU_DI_SIGNAL_CFG and SDC_IPU_DI_SIGNAL_CFG., and is modified in order to add the support for a new panel.

NOTE

The g_PanelArray[] in sdc.c file is the global array that stores the PANEL_INFO, for all supported displays (LCD, NTSC TV and PAL TV). Even if there are many ways to modify the driver, it is recommended to add a new PANEL_INFO entry, with the new PANEL_INFO structure.

4.2.4.1 PANEL_INFO

PANEL_INFO is the main structure for the LCD timing and features. It contains two structures, related to the polarity of the signal—ADC_IPU_DI_SIGNAL_CFG (a data type used in PANEL_INFO) is used for asynchronous displays, and SDC_IPU_DI_SIGNAL_CFG (a data type used in PANEL_INFO), on the other hand describes the polarities and characteristics of the RGB interface, which are not described in this application note for practical reasons.

The PANEL_INFO structure is given as follows.

```
struct PANEL_INFO_ST {
        PUCHAR NAME;
        IPU_PANEL_TYPE TYPE;
        IPU_PIXEL_FORMAT PIXEL_FMT;
        INT MODEID;
        INT WIDTH;
        INT HEIGHT;
        INT FREQUENCY;
        INT VSYNCWIDTH;
        INT VSTARTWIDTH;
        INT VENDWIDTH;
        INT HSYNCWIDTH;
        INT HSTARTWIDTH;
        INT HENDWIDTH;
        INT RD_CYCLE_PER; // in ns
        INT RD_UP_POS; // in ns
        INT RD_DOWN_POS; // in ns
        INT WR_CYCLE_PER; // in ns
        INT WR_UP_POS; // in ns
        INT WR_DOWN_POS; // in ns
        INT PIX_CLK_FREQ; // in Hz
        INT PIX_DATA_POS; // in ns
```

```
        ADC_IPU_DI_SIGNAL_CFG ADC_SIG_POL;
        SDC_IPU_DI_SIGNAL_CFG SDC_SIG_POL;
};
typedef struct PANEL_INFO_ST PANEL_INFO;
```

Table 16 gives the members of the `PANEL_INFO` structure.

Table 16. PANEL_INFO Member Features

Data type	Var. name	Description	Symbol	Unit
PUCHAR	NAME	Name of the panel	—	—
IPU_PANEL_TYPE	TYPE	Index of the panel type[1]	—	—
IPU_PIXEL_FORMAT	PIXEL_FMT	Pixel format[2]	—	—
INT	MODEID	Mode ID[3]	—	—
INT	WIDTH	Active frame width	VDISP	Pixels
INT	HEIGHT	Active frame height	HDISP	Lines
INT	FREQUENCY	Refresh rate	FV	Hz
INT	VSYNCWIDTH	VSYNC pulse width	VSW	Lines
INT	VSTARTWIDTH	Vertical back porch	VBP	Lines
INT	VENDWIDTH	Vertical front porch	VFP	Lines
INT	HSYNCWIDTH	HSYNC pulse width	HSW	Pixels
INT	HSTARTWIDTH	Horizontal back porch	HBP	Pixels
INT	HENDWIDTH	Horizontal front porch	HFP	Pixels
INT	RD_CYCLE_PER	Not used for dumb displays	—	—
INT	RD_UP_POS	Not used for dumb displays	—	—
INT	RD_DOWN_POS	Not used for dumb displays	—	—
INT	WR_CYCLE_PER	Not used for dumb displays	—	—
INT	WR_UP_POS	Not used for dumb displays	—	—
INT	WR_DOWN_POS	Not used for dumb displays	—	—
INT	PIX_CLK_FREQ	Not used for dumb displays	—	—
INT	PIX_DATA_POS	Not used for dumb displays	—	—

1 The enum on this data field is used by the `sdc.c` file, to distinguish the proper time settings between the supported displays (LCD, NTSC TV and PAL TV), when the selected display is being loaded. `IPU_PANEL_TYPE` is found in the `ipu.h` header file. The entry must be changed or added, before the `ADCPanelOffset`, in the `IPU_PANEL_TYPE`, with the new panel type of the user. enum's value must be identical with the value of the `g_PanelArray[]`, of the new `PANEL_INFO`.

2 There are three different RGB pixel formats—RGB565, RGB666, and RGB888. Selecting one of these formats specifies how the SDC interprets the frame buffer data.

3 For LCD panels, use DISPLAY_MODE_DEVICE as MODEID. There are two other supported modes. But those values belong to TV out functionality.

4.2.4.2 ADC_IPU_DI_SIGNAL_CFG

The ADC_IPU_DI_SIGNAL_CFG register structure is given as follows.

```
// Bit field of ADC Display Interface signal polarities.

typedef struct {
         UINT32 DISP_NUM  :2;
         UINT32 DISP_IF_MODE:2;
         UINT32 DISP_PAR_BURST_MODE:2;
         UINT32 DATA_POL :1;            // true = inverted
         UINT32 CS_POL  :1;             // true = active high
         UINT32 PAR_RS_POL   :1;        // true = inverse
         UINT32 WR_POL    :1;           // true = active high
         UINT32 RD_POL  :1;             // true = active high
         UINT32 VSYNC_POL  :1;          // true = active high
         UINT32 SD_D_POL :1;            // true = inverse
         UINT32 SD_CLK_POL :1;          // true = inverse
         UINT32 SER_RS_POL :1;          // true = inverse
         UINT32 BCLK_POL :1;            // true = inverted
         UINT32 Dummy      :16;         // Dummy variable for alignment
} ADC_IPU_DI_SIGNAL_CFG;
```

NOTE

This structure is not used for synchronous display panels. Here, all the bit fields can be set to zero.

4.2.4.3 SDC_IPU_DI_SIGNAL_CFG

The SDC_IPU_DI_SIGNAL_CFG register structure is given as follows.

```
// Bit field of SDC Display Interface signal polarities.
typedef struct {
         UINT32 DATAMASK_EN:1;
         UINT32 CLKIDLE_EN :1;
         UINT32 CLKSEL_EN  :1;
         UINT32 VSYNC_POL  :1;
         UINT32 ENABLE_POL :1;
         UINT32 DATA_POL   :1;       // true = inverted
         UINT32 CLK_POL    :1;       // true = rising edge
         UINT32 HSYNC_POL  :1;       // true = active high
         UINT32 Dummy      :24;      // Dummy variable for alignment.
} SDC_IPU_DI_SIGNAL_CFG;
```

Table 17 gives the members of the SDC_IPU_DI_SIGNAL_CFG register structure.

Table 17. SDC_IPU_DI_SIGNAL_CFG Member Features

Offset	Bit field name	Description	Symbol
0	DATAMASK_EN	Data mask for the display 3 (not used)[1]	
1	CLKIDLE_EN	Display 3 interface clock idle enable[2]	
2	CLKSEL_EN	Select Display 3 interface clock[3]	
3	VSYNC_POL	VSYNC signal polarity[4]	VSP
4	ENABLE_POL	Data enable polarity[5]	DEP

Table 17. SDC_IPU_DI_SIGNAL_CFG Member Features (continued)

Offset	Bit field name	Description	Symbol
5	DATA_POL	Data polarity[6]	DP
6	CLK_POL	Clock polarity[7]	CLKPOL
7	HSYNC_POL	HSYNC signal polarity[8]	HSP
8	Dummy	Not used	

1 Data mask is used to mask the data output to zero, for the Sharp TFT power off sequence. Since it is not a Sharp display, this bit field is not relevant. Because of this reason, the bit field should be set to zero (`DATAMASK_EN = FALSE`).

2 Setting `CLKIDLE_EN` to `FALSE` enables the pixel clock when VSYNC is active. Using logic 1 disables the pixel clock during this time. Most of the LCD panels show that the pixel clock signal should be present, while VSYNC is active (`CLKIDLE_EN = FALSE`).

3 Selects whether to enable or disable display clock when there is no data output. `CLKSEL_EN` sets to zero when pixel clock need to be enabled, even in the absence of data output (Data Enable active). Setting `CLKSEL_EN` to logic 1 disables pixel clock when there is no data output. Most of the LCD panels show that the pixel clock signal present is always active (`CLKSEL_EN = FALSE`).

4 Set `VSYNC_POL` to `FALSE` when VSYNC signal is active low, and `TRUE` when this signal is active high. See Section 3.2.2.1, “Vertical Timing,” for more information.

5 `Set ENABLE_POL` to `FALSE` when DRDY signal is active low, and `TRUE` when this signal is active high. See Section 3.2.2.2, “Horizontal Timing,” to review the details.

6 Set `DATA_POL` to `FALSE` when straight RGB data polarity is used, and `TRUE` when data polarity is inverted. See Section 3.2.2.2, “Horizontal Timing,” for more information.

7 For straight clock polarity, set `CLK_POL` bit field as `FALSE`, and for inverse clock polarity, set `CLK_POL` as `TRUE`. See Section 3.2.2.3, “Pixel Clock Timing,” for more information.

8 Set `HSYNC_POL` to `FALSE` when HSYNC signal is active low, and `TRUE` when HSYNC signal is active high. See Section 3.2.2.2, “Horizontal Timing,” to review the details.

4.2.4.4 GPEMode

The WinCE graphics engine uses the `GPEMode` structure to determine the 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 as follows.

```
// STRUCT GPEMode
// This structure describes a display mode
struct GPEMode
{
        int          modeId;
        int          width;
        int          height;
        int          Bpp;
        int          frequency;
        EGPEFormat   format;
};
```

Table 18 gives the members of the GPEMode structure.

Table 18. GPEMode Member Features

Data type	Var. name	Description	Symbol	Unit
int	modeId	Display or TV modes[1]	—	—
int	Width	Active frame width	VDISP	Pixels
int	Height	Active frame height	HDISP	Lines
int	Bpp	Bits per pixel[2]	BPP	bits
int	Frequency	Screen refresh rate	FV	Hz
int	Format	EGPEFormat enum[3]	—	—

1 Display mode can be either DISPLAY_MODE_DEVICE or DISPLAY_MODE_NTSC.

2 This value should be set to DISP_BPP, which has a value of 16.

3 This field represents the bits per pixel of the GPE frames, and it takes any of the following values.

```
enum EGPEFormat{
 gpe1Bpp,
 gpe2Bpp,
 gpe4Bpp,
 gpe8Bpp,
 gpe16Bpp,
 gpe24Bpp,
 gpe32Bpp,
 gpe16YCrCb,
 gpeDeviceCompatible,
 gpeUndefined
};
```

The i.MX35 supports a maximum of only 24 bits, as display interface width. In WINCE600, GPEMode is automatically set by the asynchronous display controller driver.

5 Modifying the BSP

To add support for the i.MX35 PDK WINCE600 and other synchronous panels, there are some common steps that are followed:

1. Modify the catalog
2. Modify platform.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 PANEL_INFO entry and cases for the new LCD
9. Write initialization command routine
10. Rebuild the project

In order to explain the BSP changes, consider using an LCD panel similar to Chunghwa CLAA057VA01CT. The panel is an LCD module, composed of a synchronous panel, internal backlight

unit, and an external touch screen. RGB interface is 6 bits per color, and it uses HSYNC, VSYNC, DE and PIXCLK. See Figure 4, to see the connections between the i.MX35 and this type of LCD.

5.1 Modifying the 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\iMX35-3DS-PDK1_6\CATALOG folder as shown in Figure 18.

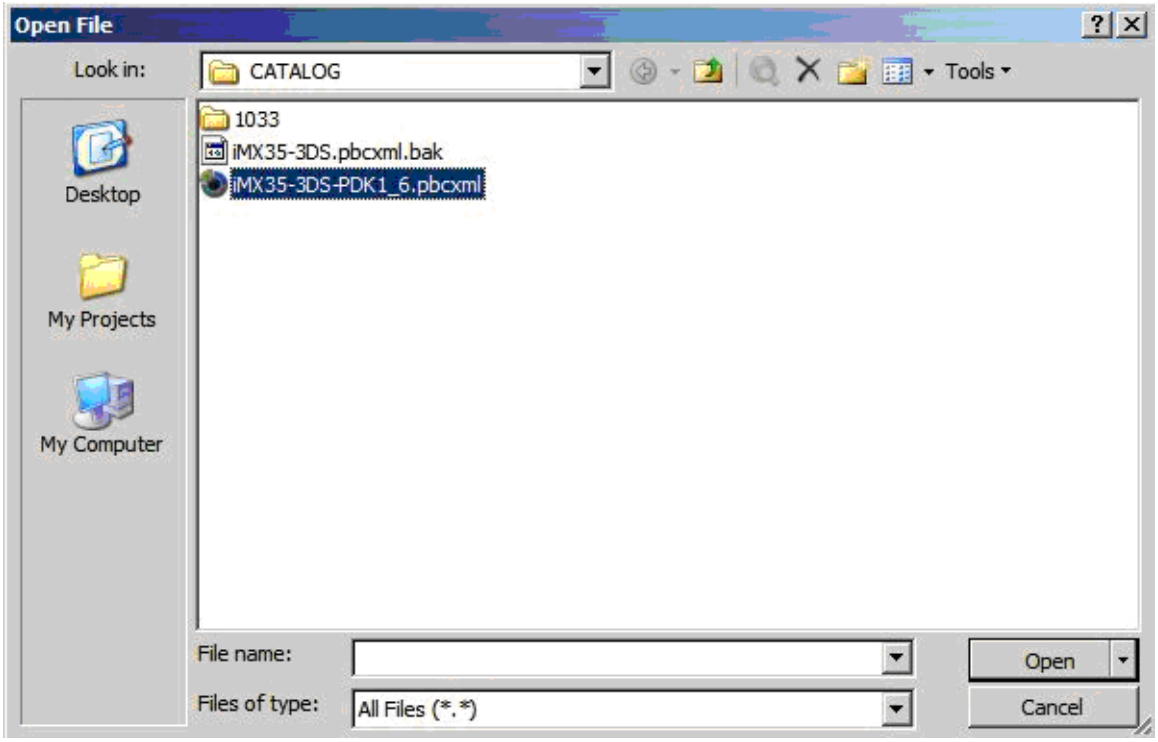

Figure 18. Opening Catalog File

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.

For this, right click the Display folder and select Add Catalog Item as shown in Figure 19.

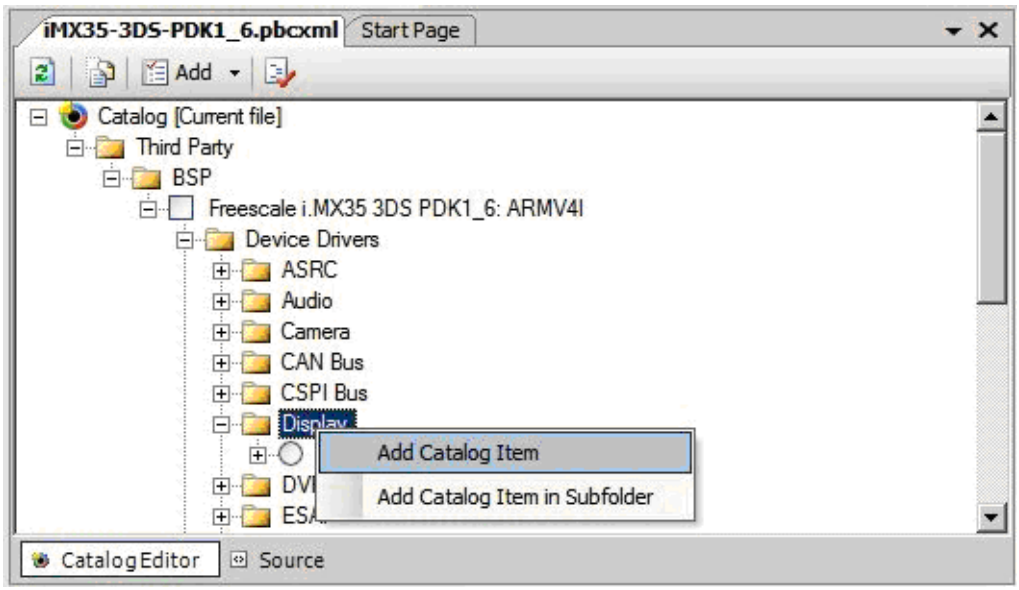

Figure 19. Add Catalog Item

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

Table 19. Display Driver Catalog Item Properties

Properties	Value
Description	IPU SDC display driver CHUNGHWA CLAA057VA01CT
Title	CHUNGHWA CLAA057VA01CT(VGA)
Additional variables	BSP_DISPLAY_CHUNGHWA_CLAA057VA01CT BSP_I2CBUS1 BSP_MCU
Modules	ddraw_ipu.dll
Choose one group	True

As shown in Figure 20, all the other properties must have the default value.

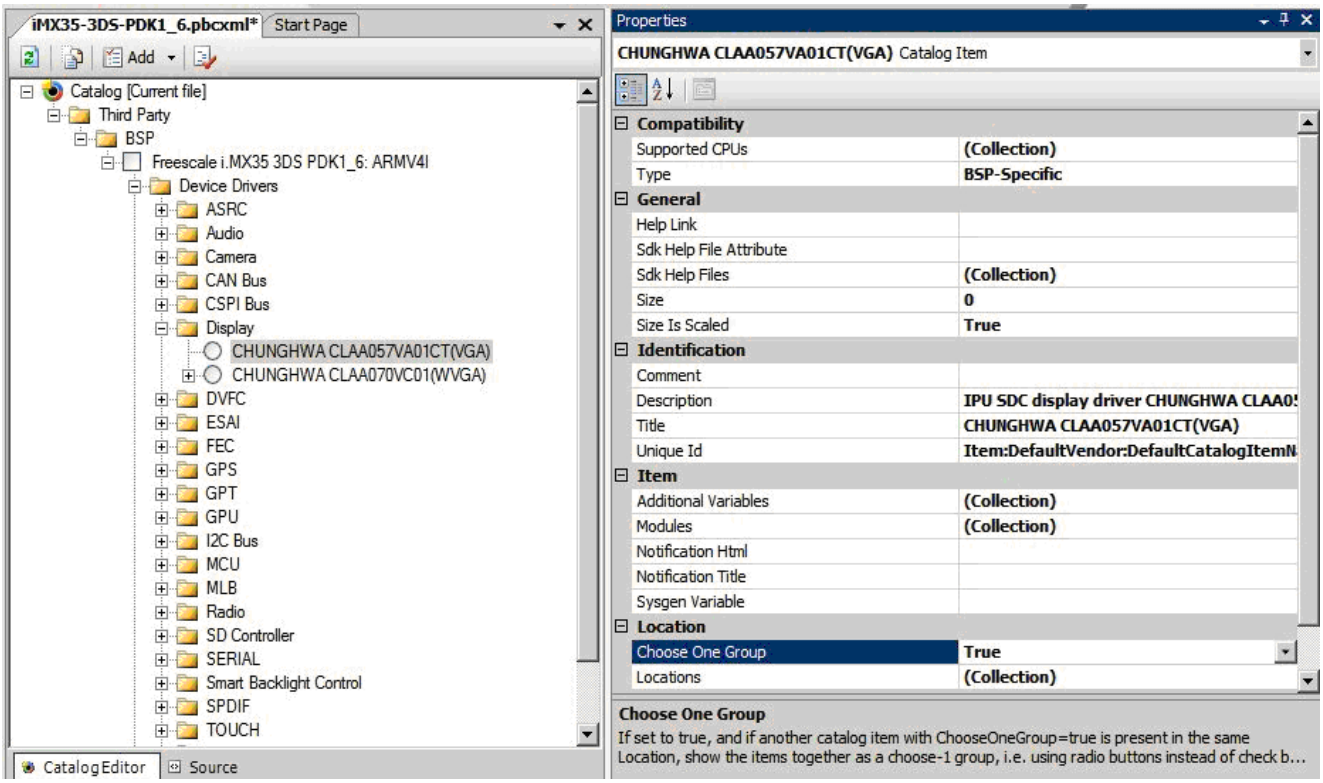

Figure 20. Catalog Item Properties

5. Click OK on variables collection message box. Save the file and close the file and then again open the iMX35-3DS-PDK1_6-Mobility project. Then, refresh the catalog by using the button on the top of the catalog items view.
6. Verify that CUNGHWA CLAA070VC01(WVGA) catalog item, has the properties in Table 20.

Table 20. CLAA070VC01(WVGA) Catalog Item Properties

Properties	Value
Description	#FSL:iMX35-3DS-PDK1_6:CLAA070VC01:Title
Title	#FSL:iMX35-3DS-PDK1_6:CLAA070VC01:Title
Additional variables	BSP_DISPLAY_CHUNGHWA_CLAA070VC01 BSP_I2CBUS1 BSP_MCU
Modules	ddraw_ipu.dll
Choose one group	True

When the iMX35-3DS-PDK1_6-Mobility project is opened again in Visual Studio, the CHUNGHWA CLAA057VA01CT entry is also available in the catalog. Now, the CHUNGHWA CLAA057VA01CT(VGA) item can be selected. But some files need to be modified, before the new LCD support is completed. CHUNGHWA CLAA070VC01(WVGA) is included in the project until the

platform.bib file is modified.

5.2 Platform.reg

Open platform.reg file, and add the register configuration settings, for the new panel. Basically conditions to distinguish display registry settings for the WVGA panel—CLAA070VC01, and for the new display— CLAA057VA01CT, need to be added.

The code for Platform.reg is given as follows.

```
IF BSP_DISPLAY_CHUNGHWA_CLAA070VC01
[HKEY_LOCAL_MACHINE\Drivers\Display\DDIPU]
"Bpp"=dword:10                  ; 16bpp
"VideoBpp"=dword:10             ; RGB565
"PanelType"=dword:3             ; CHUNGHWA WVGA Panel
"VideoMemSize"=dword:600000     ; 6.0MB
ENDIF ; BSP_DISPLAY_CHUNGHWA_CLAA070VC01
IF BSP_DISPLAY_CHUNGHWA_CLAA057VA01CT
[HKEY_LOCAL_MACHINE\Drivers\Display\DDIPU]
"Bpp"=dword:10                  ; 16bpp
"VideoBpp"=dword:10             ; RGB565
"PanelType"=dword:4             ; CHUNGHWA VGA Panel
"VideoMemSize"=dword:600000     ; 6.0MB
ENDIF ; BSP_DISPLAY_CHUNGHWA_CLAA057VA01CT
```

VGA PanelType is determined by the IPU_PANEL_TYPE enum created for this panel. It must have the same value of that of the offset of the new VGA PANEL_INFO, in the g_PanelArray[] array, which is located in the sdc.c file. See Section 5.7, “Adding a Panel Type enum for the New LCD,” and Section 5.8, “Creating PANEL_INFO Entry and Cases for New LCD,” for more information.

5.3 Platform.bib

Platform.bib determines the files that are to be included in the image. For display driver, ddraw_ipu.dll and ipu_base.dll files have to be included. So, platform.bib entries are updated by adding the inclusion of these modules, when BSP_DISPLAY_CHUNGHWA_CLAA057VA01CT variable is set.

The code for Platform.bib is given as follows.

```
; @CESYSGEN IF CE_MODULES_DISPLAY
IF BSP_NODISPLAY !
IF BSP_MCU
IF BSP_I2CBUS1
#if (defined BSP_DISPLAY_CHUNGHWA_CLAA070VC01 || defined BSP_DISPLAY_CHUNGHWA_CLAA057VA01CT)
; ------------------------------------------------------------------------------
; IPU Common Driver
;
ipu_base.dll    $(_FLATRELEASEDIR)\ipu_base.dll                 NK  SHK
; ------------------------------------------------------------------------------

; ------------------------------------------------------------------------------
; Post-processor Driver
;
pp.dll          $(_FLATRELEASEDIR)\pp.dll                       NK  SHK
;
; ------------------------------------------------------------------------------
```

```
; ------------------------------------------------------------------------------
; Postfilter Driver
;
pf.dll          $(_FLATRELEASEDIR)\pf.dll                       NK  SHK
; ------------------------------------------------------------------------------

; ------------------------------------------------------------------------------
; Display Driver
;
ddraw_ipu.dll $(_FLATRELEASEDIR)\ddraw_ipu.dll                  NK  SHK
; ------------------------------------------------------------------------------
#endif
ENDIF BSP_I2CBUS1
ENDIF BSP_MCU
ENDIF BSP_NODISPLAY !
; @CESYSGEN ENDIF CE_MODULES_DISPLAY
```

5.4 Setting Up Power LCD Voltages

Before connecting any LCD panel to the i.MX35 PDK, it is important to verify whether the LCD can be powered with the proper supply voltages, and the display data interface has the correct value of VIO (voltage input/output). Power settings are handled by the ATLAS PMIC and some other voltage regulators. During PDK initialization, the PMIC (power management IC) driver must configure the PMIC, in order to set up the power voltage configuration.

Chunghwa CLAA070VC01 WVGA panel requires two different power supply levels—VDD (5.0 V) and VCC (3.3 V). In the i.MX35 PDK, these two voltages are directly connected to a voltage regulator that cannot be modified by the software.

The most common power supply levels that are included in the i.MX35 PDK—1.8V, 2.8, 3.3V and 5.0 V, could be found at UI generic I/F connector (J4). So, only the LCD need to be connected with the proper power supply lines in this connector.

In case, any extra software consideration is needed for the power supply settings, the code in the `BSPInitializeLCD()` function, which is in `bspdisplay.cpp`, has to be added to the `eIPU_SDC` case.

The code for `bspdisplay.cpp` is as follows.

```
BOOL BSPInitializeLCD(IPU_DRIVE_TYPE dispType)
{
        ...
        BSPLCDPower(TRUE);
        switch (dispType)
        {
                case eIPU_ADC:
                break
                case eIPU_SDC:
                break;
        }
}
```

5.5 Setting Up Reset Signal

Many synchronous display modules need a reset signal. It is recommended for the signal to be controlled by the microprocessor. Neither WVGA nor VGA LCDs mentioned in this application note, need a reset signal. But it need not mean that the BSP does not support it. The C9S08DZ60 8-bit microprocessor is assigned a pin, in case the LCD should need to perform the reset function (`PTG3 - LCD_LCS1_RST`). An enum is defined for all reset signals handled by the MC9S08DZ60 in `mcu_reset.h` header file, and the LCD reset enumeration is found there.

The code for the `mcu_reset.h` header file is given as follows.

```
typedef enum _MCU_MC9S08DZ60_RESERT_ID {
. . .
        MCU_MC9S08DZ60_RESERT_AUDIO  = 9,
        MCU_MC9S08DZ60_RESERT_LCD    = 10,
. . .
} MCU_RESERT_ID;
```

Additionally, the BSP contains the `McuGpioReset()` function, which is used to generate signal waveform for an LCD. It needs, either an active low or active high reset signal. Sometimes, and depending on the LCD requirements, it becomes necessary to add a delay between the power enable signal `(BSPLCDPower)` and reset signal `(McuGpioReset)`. Reset signal code should be included in `BSPEnableLCD()` function.

The following shows the code for an active low reset, which is most commonly used in LCDs.

```
BOOL BSPEnableLCD(IPU_DRIVE_TYPE dispType)
{
        ....

        switch (dispType)
        {
        case eIPU_SDC:
                // Delay between LCD power-on and LCD reset (20 ms)
                Sleep(20);
                // LCD pin goes low
                McuGpioReset(MCU_MC9S08DZ60_RESERT_LCD, TRUE);
                // More than 150 ns delay
                Sleep(0);
                // LCD pin goes high
                McuGpioReset(MCU_MC9S08DZ60_RESERT_LCD, FALSE);
                break;


        case eIPU_ADC:
                break;
        }
}
```

5.6 Setting Up the Backlight

Ensure that the backlight is turned ON, as the contents on the LCD panel are 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.MX35 PDK, this signal is generated by an IPU pin called CONTRAST. So, if the LCD backlight control signal is connected to this pin, additional change need not be performed, to adapt the backlight control.

5.7 Adding a Panel Type enum for the New LCD

A new IPU_PANEL_TYPE enumeration is created for this new panel when an another LCD panel is added for support. IPU_PANEL_TYPE enumeration can be found in the ipu.h header file. Here, all synchronous panel entries are placed before the TV entries and ADCPanelOffset. The offset after this entry represents the position of the new synchronous PANEL_INFO structure, in the g_PanelArray[] array, which is in the sdc.c file, and also the PanelType value of this panel in the platform.reg file.

The code for ipu.h header file is given as follows.

```
typedef enum {
        // g_PanelArray [0]...
        IPU_PANEL_SHARP_QVGA_LQ035Q7DB02,    // Registry value is PanelType 0
        IPU_PANEL_NEC_VGA_NL6448BC33_46,     // Registry value is PanelType 1
        IPU_PANEL_EPSON_VGA_L4F00242T03,     // Registry value is PanelType 2
        IPU_PANEL_CHUNGHWA_WVGA_CLAA070VC01, // Registry value is PanelType 3
        IPU_PANEL_CHUNGHWA_VGA_CLAA057VA01CT,// Registry value is PanelType 4
        // New SDC panel goes here           // Registry value ++
        IPU_TV_VGA_NTSC,                     // Registry value is TVSupported
        IPU_TV_VGA_PAL,                      // Registry value is TVSupported + 1
        IPU_TV_D1_NTSC,                      // Registry value is TVSupported + 2
        IPU_TV_D1_PAL,                       // Registry value is TVSupported + 3
        // New TV panel goes here            // Registry value ++
        ADCPanelOffset,                      // Below is for ADC
        // g_ADCPanelArray [0]...
        IPU_PANEL_TOSHIBA_QVGA_ADC,   // Registry value is PanelType ADCPanelOffset + 1
        IPU_PANEL_EPSON_QVGA_ADC,     // Registry value is PanelType ADCPanelOffset + 2
        // New ADC panel goes here    // Registry value ++
        numPanel,
} IPU_PANEL_TYPE;
```

5.8 Creating PANEL_INFO Entry and Cases for New LCD

Based on the information described in Section 4.2.4.1, “PANEL_INFO,” fill the PANEL_INFO structure for the new synchronous panel, and include it in g_PanelArray[] array on sdc.c file. The position in the array of the new LCD is determined by the IPU_PANEL_TYPE enumeration.

See the following code for the CHUNGHWA VGA CLAA057VA01CT dumb panel, which is located in sdc.c.

```
PANEL_INFO g_PanelArray[numPanel] =
{
        // Sharp Definitions
        {
                (PUCHAR) "Sharp QVGA Panel",
                IPU_PANEL_SHARP_QVGA_LQ035Q7DB02,
                ...
        },

        //NEC VGA Panel definitions
        {
                (PUCHAR) "NEC VGA Panel",
                IPU_PANEL_NEC_VGA_NL6448BC33_46,
                ...
        },
```

```
// Epson VGA dumb Panel definitions
{
        (PUCHAR) "EPSON VGA Panel",
        IPU_PANEL_EPSON_VGA_L4F00242T03,
        ...
},

// CHUNGHWA WVGA Definitions
{
        (PUCHAR) "CHUNGHWA WVGA Panel",
        IPU_PANEL_CHUNGHWA_WVGA_CLAA070VC01,

},

// CHUNGHWA VGA Definitions
{
        (PUCHAR) "CHUNGHWA VGA Panel",
        IPU_PANEL_CHUNGHWA_VGA_CLAA057VA01CT,
        IPU_PIX_FMT_RGB565,          // Pixel Format
        DISPLAY_MODE_DEVICE,         // Mode ID
        640,                         // width
        480,                         // height
        60,                          // frequency
        1,                           // Vertical Sync width
        34,                          // Vertical Start Width
        11,                          // Vertical End Width
        1,                           // Horizontal Sync Width
        46,                          // Horizontal Start Width
        114,                         // Horizontal End Width
        0,                           // Write Cycle Period
        0,                           // Write Up Position
        0,                           // Write Down Position
        0,                           // Read Cycle Period
        0,                           // Read Up Position
        0,                           // Read Down Position
        0,                           // Pixel Clock Cyle Frequency
        0,                           // Pixel Data Offset Position
        // ADC Display Interface signal polarities
        {
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
                0,
        },
        // Display Interface signal polarities
        {
                FALSE,                       // Data mask enable
                TRUE,                        // Clock Idle enable
```

```
                        FALSE,                          // Clock Select Enable
                        FALSE,                          // Vertical Sync Polarity
                        TRUE,                           // Output enable polarity
                        FALSE,                          // Data Pol
                        TRUE,                           // Clock Pol
                        FALSE,                          // HSync Pol
                }
        },

        // New SDC panel goes here

        // NTSC TV VGA mode definitions
        {
                (PUCHAR) "NTSC VGA",
                IPU_TV_VGA_NTSC,
                ...
        },

        // PAL TV VGA mode definitions
        {
        (PUCHAR) "PAL VGA",
        IPU_TV_VGA_PAL,

        },

        // NTSC TV D1 mode definitions
        {
                (PUCHAR) "NTSC D1",
                IPU_TV_D1_NTSC,
        },

        // PAL TV D1 mode definitions
        {
                (PUCHAR) "PAL D1",
                IPU_TV_D1_PAL,
        },

        // New Panel definitions go here
        // Put other Panel info here in future expasion
};
```

Two new cases are added during the SDC initialization, in order to support the new LCD SDC module initialization. All `IPU_PANEL_CHUNGHWA_WVGA_CLAA070VC01` case occurrences are to be found, and a new case is added for the `IPU_PANEL_CHUNGHWA_VGA_CLAA057VA01CT` panel type. The code for the same is as follows.

```
UINT32 InitializeSDC(PANEL_INFO *currentPanel, int bpp)
{
        ...
        //----- Display interface configuration
        switch(currentPanel -> TYPE)
        {
                case IPU_PANEL_SHARP_QVGA_LQ035Q7DB02:
                case IPU_PANEL_CHUNGHWA_WVGA_CLAA070VC01:
                case IPU_PANEL_CHUNGHWA_VGA_CLAA057VA01CT:
                ...
        }

        switch (currentPanel -> TYPE)
        {
                case IPU_PANEL_SHARP_QVGA_LQ035Q7DB02:
```

```
                case IPU_PANEL_NEC_VGA_NL6448BC33_46:
                case IPU_PANEL_EPSON_VGA_L4F00242T03:
                case IPU_PANEL_CHUNGHWA_WVGA_CLAA070VC01:
                case IPU_PANEL_CHUNGHWA_VGA_CLAA057VA01CT:
                ...
        }

...
}
```

5.9 Writing Initialization Command Routine

The panels described in this application note do not need any serial initialization command routine. But in case it is needed, the code is placed in the `BSPEnableLCD()` function, which is located in `bhspdisplay.cpp`, just after the reset signal code. At that point, the CSPI (configurable serial peripheral interface) interface has already been configured. The initialization routine is provided by the LCD vendor.

The code for `bhspdisplay.cpp` is as follows.

```
BOOL BSPEnableLCD(IPU_DRIVE_TYPE dispType)
{
        ....

        switch (dispType)
        {
                case eIPU_SDC:
                        ...
                        break;
                case eIPU_ADC:
                        ...
                        break;

        }
}
```

5.10 Rebuilding the Project

At the end, project can be rebuilt, by using Rebuild Current BSP and Subprojects—Menu Build > Advanced Build Commands > Rebuild Current BSP and Subprojects.

6 Summary, Tips and Tricks

6.1 Support for VGA and Bigger Panels

The i.MX35 supports several LCD resolutions. The range goes from the small panels, to the SVGA (800 × 600), which has the maximum display resolution. However, there are some considerations and limitations for this resolution that should be aware of. This section explains the possible scenarios that are experienced when using LCDs of resolution above VGA in the i.MX35 platform.

6.1.1 Frame Rate

Depending on the pixel clock, back porch, front porch, and memory interface, the frame rate might decrease to less than 60 fps. For example, on WinCE, the microprocessor creates the frames related with the WinCE desktop, and place the images in the memory. In this use case, incoming frames do not come from IC, rather they come from the memory display buffer. An 800 × 600 frame is periodically created, and after that, the display interface (DI) sends this information to the VGA, WVGA or SVGA LCD. DI sends the complete buffer in a little more time than the HORIZONTAL_RESOLUTION × VERTICAL_RESOLUTION × PIXCLK period. For example, for an SVGA panel, this time is the result of 800 × 600 × PIXCLK period (usually 30 ns). This implies one frame in every 14.43 ms or 69 frames per second (fps). But considering the front and back porch for the horizontal and vertical adjustment, it is probable for the frame rate to decrease less than 60 fps.

6.1.2 Image Converter for WVGA and SVGA

In contrast with the i.MX31, the i.MX35 IPU image converter (IC) buffer is larger, as seen in the IPU features table of the *i.MX35 (MCIMX35) Multimedia Applications Processor Reference Manual* (IMX35RM). The maximum frame size after resizing, either downsizing or upsizing is 800 × 1024. The IC performs color space conversion, combining (alpha blending), downsizing, upsizing, and other operations on the image or video frames. The IC is also the IPU entry point for the camera interface. The IPU IC output cannot be greater than 800.

6.1.3 Memory Access

The i.MX35 can create WinCE desktop frames that require a lot of memory transfers. For example, in a normal SVGA application in the i.MX35 PDK, where the memory interface is a double data rate (DDR), running at 133 MHz, the maximum memory bandwidth is 266,000,000 × 32 bits (266, because DDR latches data on rising and falling edge). Even when there is a large bandwidth, DI shares the memory with the processor and all the peripherals, included in the i.MX35.

For WinCE desktop, the average percentage of memory that the DI uses to maintain the SVGA interface is calculated as follows.

$$\textit{Memory Bandwidth} = 266{,}000{,}000 \times 32 \text{ bits} = 8{,}512{,}000{,}000 \text{ bits per second} \qquad \textbf{Eqn. 3}$$

$$\textit{SVGA frame write access} = 800 \times 600 \text{ pixels} \times 16 \text{ bits (RGB565)} \times 60 \text{ fps} = 460{,}800{,}000 \qquad \textbf{Eqn. 4}$$

$$\textit{SVGA frame read access} = 800 \times 600 \text{ pixels} \times 16 \text{ bits (RGB565)} \times 60 \text{ fps} = 460{,}800{,}000 \qquad \textbf{Eqn. 5}$$

$$\textit{Percentage memory used by DI} = (460{,}800{,}000 \times 2) \div 8{,}512{,}000{,}000 = 10.83\ \% \qquad \textbf{Eqn. 6}$$

460,800,000 is multiplied by 2, as it is needed first to create the frame (write) first and then, read the contents from the buffer, and send them to the DI.

The number calculated above is only for the DI. 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 21.66% of the memory bandwidth. This number varies, depending on the

output of the resized frame after post-processing. Also, if Background (WinCE desktop) and Foreground (camera, for example) planes are worked at the same time, the usage increases as well. The memory usage depends on the number of planes used, and the number of times those frames are accessed.

7 References

The references for this application note are as follows:

- *i.MX35 (MCIMX35) Multimedia Applications Processor Reference Manual* (IMX35RM)
- *i.MX35 3-Stack Windows Embedded CE 6.0* (IMX35_PDK_WINCE_REFERENCE_M)
- *i.MX35 Windows Embedded CE 6.0 User's Guide* (IMX35_PDK_WINCE_USER_GUIDE)
- i.MX35 PDK schematic files (available in the IMX35_PDK_DESIGN_FILES installation):
  - MAD Modular Platform : RINGO Personality Board A
  - MAD 3DS Platform : i.MX35 CPU Engine Board A2

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	04/2010	Initial release.

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: AN3978
Rev. 0
04/2010