AN10880

Using the LCD interface in the LPC313x/LPC314x/LPC315x devices

Rev. 01 — 21 October 2009 Application note

Document information

Info	Content
Keywords	LPC3130, LPC3131, LPC3141, LPC3143, LPC3152, LPC3154 LCD interface, LCD controller, LCD display.
Abstract	This document provides a guide on how to configure the LPC313x LCD interface.

Revision history

Rev	Date	Description
01	20091021	Initial version.

Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

The LPC313x/4x/5x devices provide an LCD interface which can be used to interface with 6800 (Motorola) and 8080 (Intel) compatible LCD controllers with 4/8/16-bit modes. A serial interface is also supported, and the output clock can be shifted by 25 %, 50 %, or 75 % in order to comply with the specifications of the LCD controller.

Using the LCD interface is mutually exclusive with using SDRAM, i.e., users may use just one, or the other, but not both in a system. However, bus-based LCD panels which have an embedded LCD controller and frame buffer can be connected to the memory bus so that both SDRAM and an LCD panel can be used in such a configuration. This configuration does not use the LCD interface IP at all in the LPC31xx.

The speed of the interface can be adjusted in software to match the speed of the connected LCD display. In addition, a 16 byte FIFO is implemented for sending control and data information to the LCD controller. Support for maskable interrupts and DMA transfers is provided.

This application note provides a guide to program the LCD interface for both serial and parallel modes. Embedded Artists' EALPC3131 evaluation board is used (for serial mode) and full source code is included which requires the Keil tool (evaluation version can be used). The NXP LPC313x Common Driver Library (CDL) is required and is available for download from:

http://www.standardics.nxp.com/support/documents/microcontrollers/zip/lpc313x.cdl.drivers.zip?1

The provided folder should be copied into the following directory (where the CDL is installed):

C:\nxpmcu\Software\csps\lpc313x\bsps\ea3131\examples\

2. Configuring the EA3131 board (only for Serial mode)

On the EALPC3131 evaluation board, although the LCD interface is not supported, some lines are available through connector J8 (expansion connector). See Fig 1. The following signals are available from J8 connector:

Table 1. J8 connector signals

Signal (for serial mode)	J8 – pin #
mLCD_DB_15 (serial data out)	15
mLCD_DB_14 (serial data in)	16
mLCD_DM_13 (serial clock)	17
mLCD_CSB	9

For debug purposes, an Oscilloscope can be connected here to probe these lines in order to monitor the LCD interface signals for Serial mode.

The lines required for Parallel mode are not available on this board, so the user should monitor these lines using his or her own board.

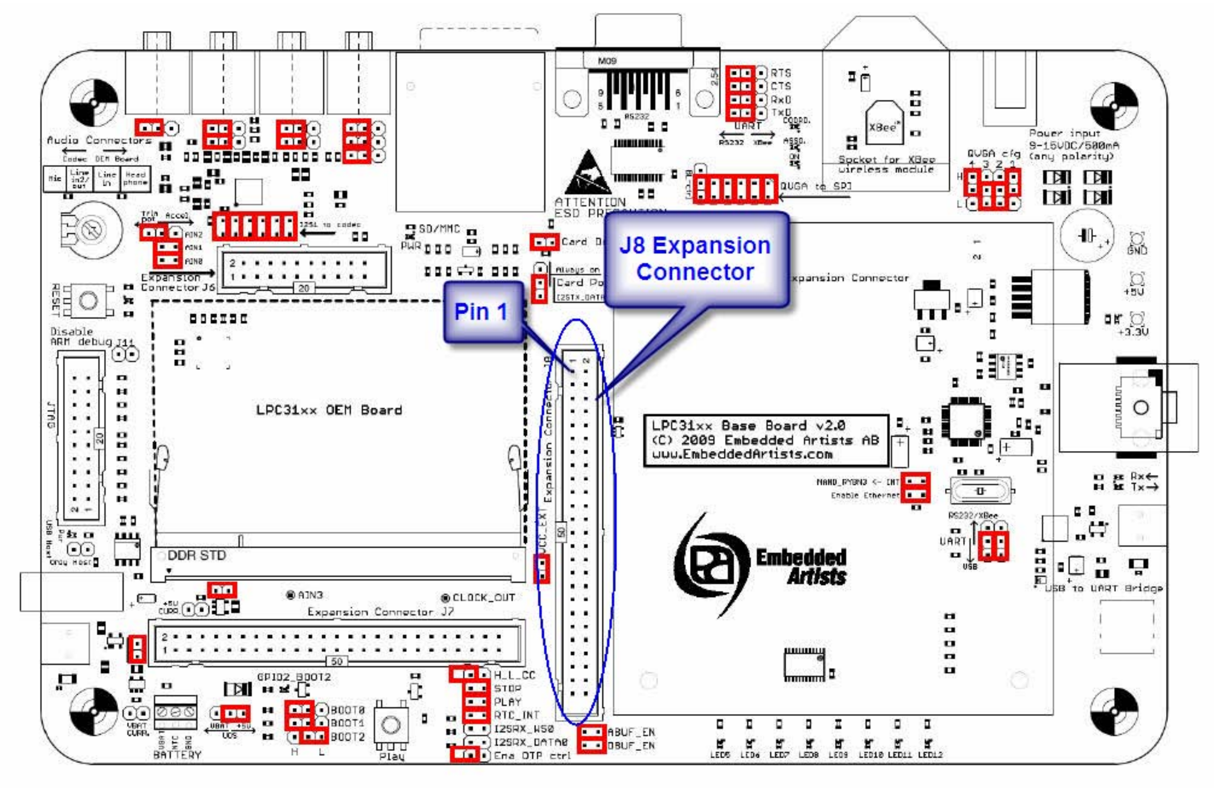


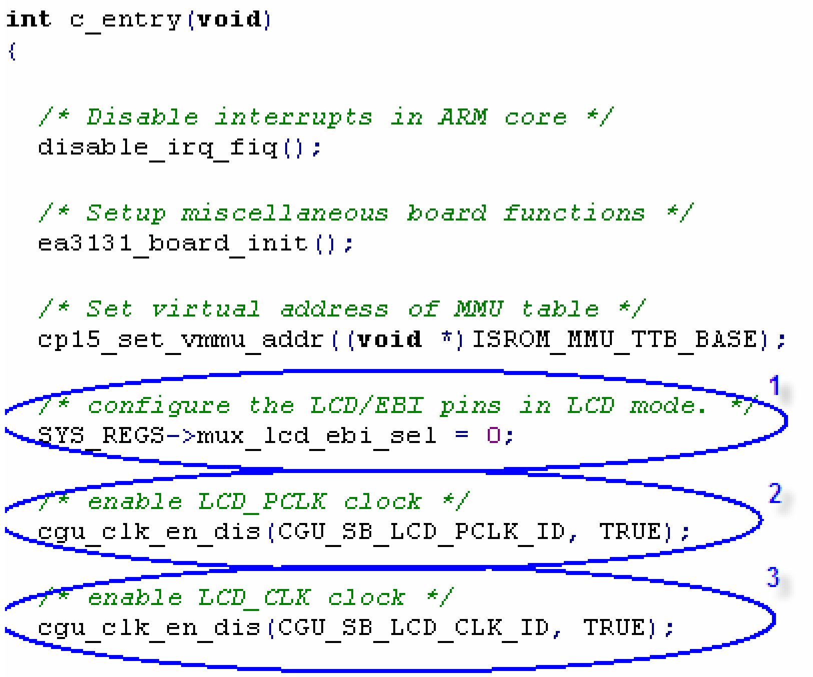
Fig 1. The EA3131 board and the J8 expansion connector

3. Programming the LCD interface

Before using the LCD Interface to send commands and/or data to the LCD display, some initialization and configuration is required. Initialization consists of configuring the LCD/EBI pins in LCD mode (the LCD interface lines are multiplexed with the External Bus Interface lines), enabling the LCD’s clock signals, configuring the required operation mode (serial or parallel, 4, 8, or 16 bits, 8080 or 6800 mode, etc) and the interrupt system (if interrupts will be used).

Once the LCD interface is properly initialized and configured, it can be used to send commands and data to the LCD display. This is accomplished by writing to the appropriate register. For example, for 8-bit and 16-bit instructions, the INST_BYTE register should be written. For 32-bit instructions, the INST_WORD register should be used instead. When sending data to the LCD display, the DATA_BYTE register should be used with 8-bit or 16-bit data, while the DATA_WORD register should be used for 32-bit data.

Fig 2 shows the code used to initialize the LCD interface.

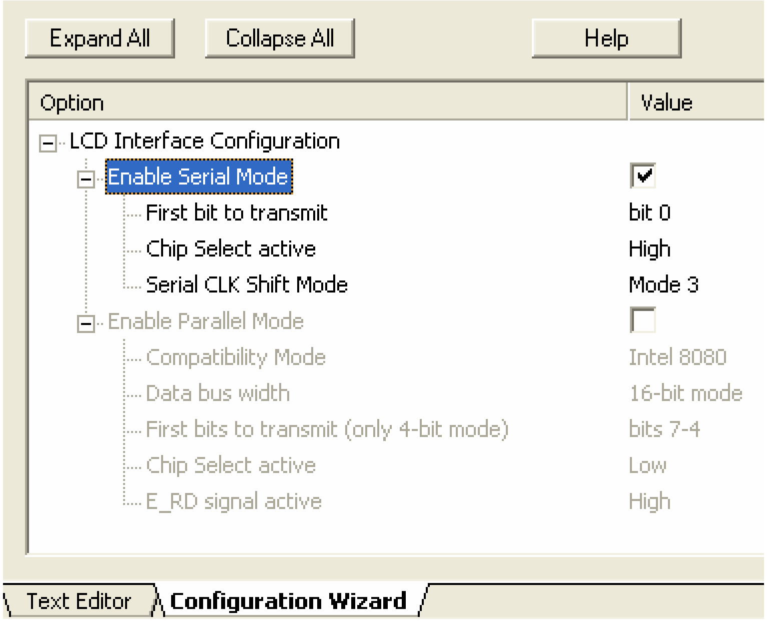


```
int c_entry(void)
{

  /* Disable interrupts in ARM core */
  disable_irq_fiq();

  /* Setup miscellaneous board functions */
  ea3131_board_init();

  /* Set virtual address of MMU table */
  cp15_set_vmmu_addr((void *)ISROM_MMU_TTB_BASE);

  /* configure the LCD/EBI pins in LCD mode. */
  SYS_REGS->mux_lcd_ebi_sel = 0;

  /* enable LCD_PCLK clock */
  cgu_clk_en_dis(CGU_SB_LCD_PCLK_ID, TRUE);

  /* enable LCD_CLK clock */
  cgu_clk_en_dis(CGU_SB_LCD_CLK_ID, TRUE);
```

Fig 2. Detecting the LPC313x device

In the above code, line (1) configures the LCD/EBI pins in LCD mode, while lines (2) and (3) enable the LCD_PCLK and LCD_CLK signals respectively. In case the speed of the interface needs to be adjusted to match the speed of the connected LCD display, the clock default values can be modified in the ***lpc313x_cgu_default.c*** file (see domain 3 – AHB0_APB2 ClockDomain).

In order to configure the LCD_CONTROL register, the lcd_config.h file presents a graphical menu where different configuration options can be set (the Configuration Wizard tab should be selected). This is shown in Fig 3.

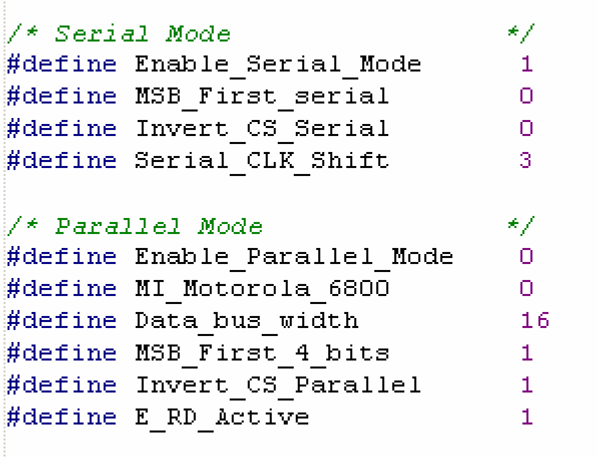


Option	Value
⊟ LCD Interface Configuration	
⊟ Enable Serial Mode	☑
First bit to transmit	bit 0
Chip Select active	High
Serial CLK Shift Mode	Mode 3
⊟ Enable Parallel Mode	□
Compatibility Mode	Intel 8080
Data bus width	16-bit mode
First bits to transmit (only 4-bit mode)	bits 7-4
Chip Select active	Low
E_RD signal active	High

Fig 3. LCD interface configuration options

In the above configuration menu, the desired Serial / Parallel mode should be selected (don't select both at the same time). Once this mode is selected, the corresponding options are enabled and available for modification.

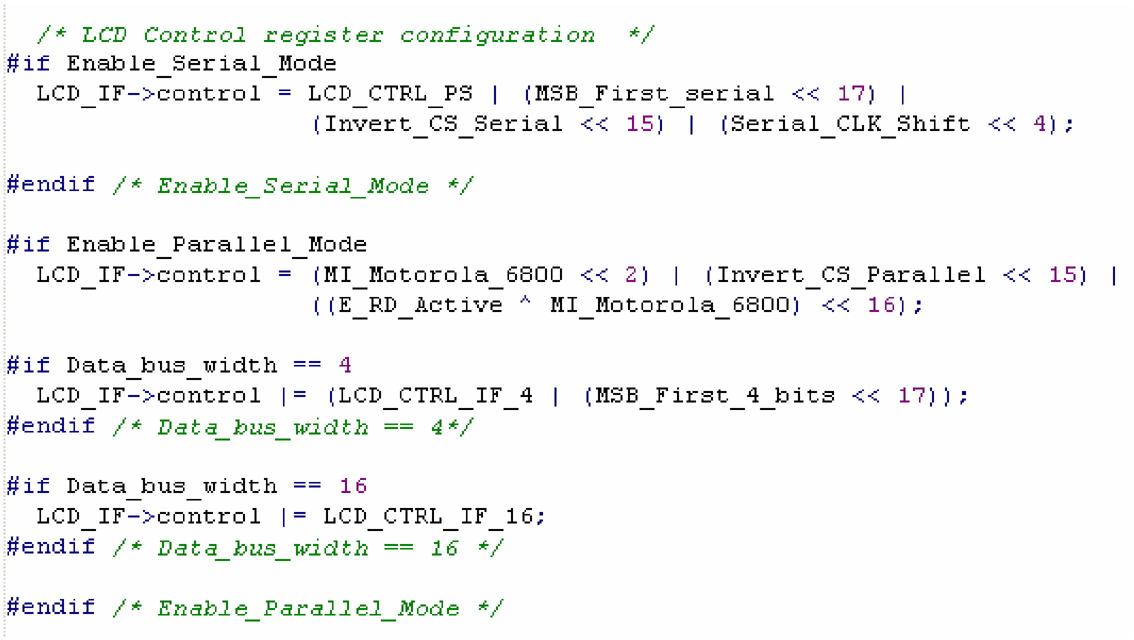
The #define statements, with their selected values for the Serial Mode, are shown in Fig 4.



```
/* Serial Mode                 */
#define Enable_Serial_Mode      1
#define MSB_First_serial        0
#define Invert_CS_Serial        0
#define Serial_CLK_Shift        3

/* Parallel Mode               */
#define Enable_Parallel_Mode    0
#define MI_Motorola_6800        0
#define Data_bus_width          16
#define MSB_First_4_bits        1
#define Invert_CS_Parallel      1
#define E_RD_Active             1
```

Fig 4. #define values containing the selected configuration

The next step is to configure the LCD_CONTROL register with the above configuration. This is shown in Fig 5.

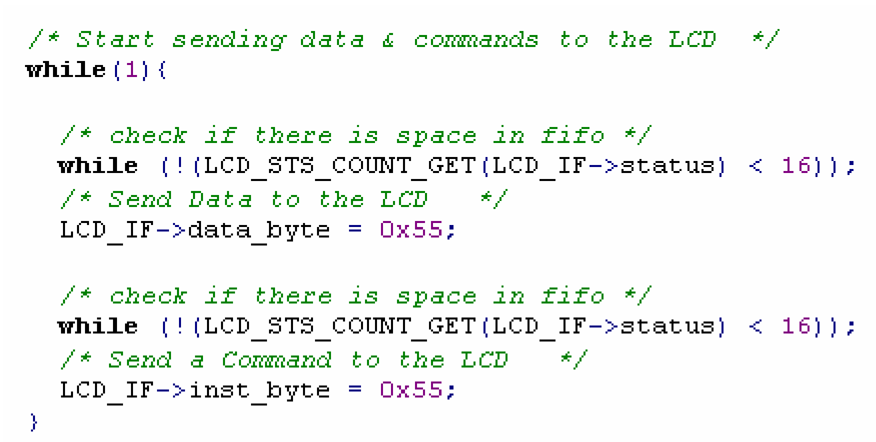


```
  /* LCD Control register configuration  */
#if Enable_Serial_Mode
  LCD_IF->control = LCD_CTRL_PS | (MSB_First_serial << 17) |
                    (Invert_CS_Serial << 15) | (Serial_CLK_Shift << 4);

#endif /* Enable_Serial_Mode */

#if Enable_Parallel_Mode
  LCD_IF->control = (MI_Motorola_6800 << 2) | (Invert_CS_Parallel << 15) |
                    ((E_RD_Active ^ MI_Motorola_6800) << 16);

#if Data_bus_width == 4
  LCD_IF->control |= (LCD_CTRL_IF_4 | (MSB_First_4_bits << 17));
#endif /* Data_bus_width == 4*/

#if Data_bus_width == 16
  LCD_IF->control |= LCD_CTRL_IF_16;
#endif /* Data_bus_width == 16 */

#endif /* Enable_Parallel_Mode */
```

Fig 5. LCD_CONTROL register configuration

The last part of the LCD interface initialization is configuring the appropriate LCD Interrupt Mask register (LCD_INT_MASK). The default value for this register is 0x0F, which means interrupts are masked and not generated. When interrupts need to be enabled just write a 0 in the corresponding bit of the Interrupt Mask register.

Once the LCD interface is properly initialized, commands and data can be sent to the LCD display. An example of this is shown in Fig 6.



```
/* Start sending data & commands to the LCD  */
while(1){

  /* check if there is space in fifo */
  while (!(LCD_STS_COUNT_GET(LCD_IF->status) < 16));
  /* Send Data to the LCD   */
  LCD_IF->data_byte = 0x55;

  /* check if there is space in fifo */
  while (!(LCD_STS_COUNT_GET(LCD_IF->status) < 16));
  /* Send a Command to the LCD   */
  LCD_IF->inst_byte = 0x55;
}
```

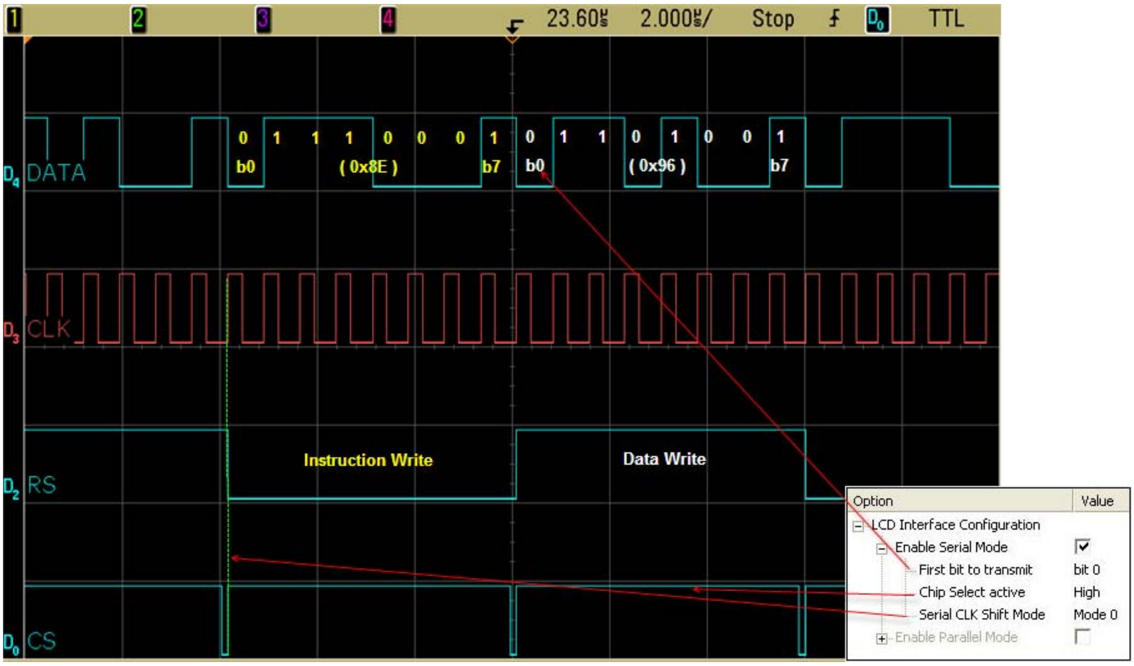
Fig 6. Sending data and commands to the LCD display

In the above code, the data is written into the DATA_BYTE register (using the pointer LCD->data_byte), while commands are written into INST_BYTE (using LCD->inst_byte). These same registers can be used to write 16-bit data. For 32-bit, we use DATA_WORD and INST_WORD registers.

The ***while (! (LCD_STS_COUNT_GET(LCD_IF->status) < 16))*** line will check if the FIFO has space before writing to the register.

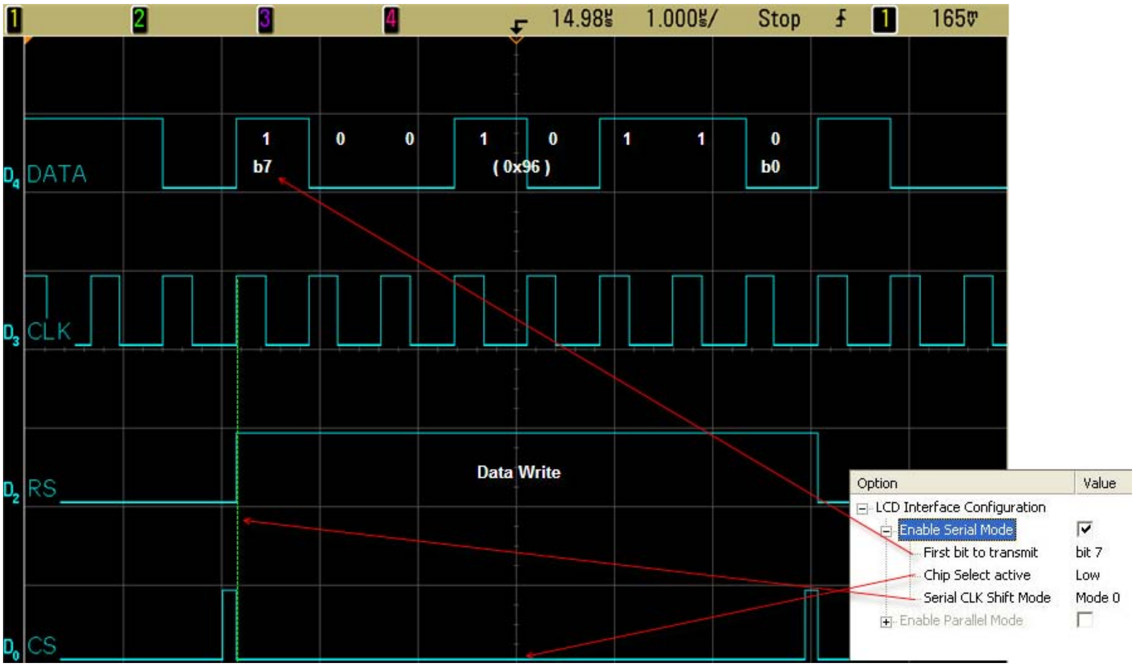
4. Testing the LCD interface

The following scopes show the LCD interface timing when configured in different modes. The attached screenshot shows the software's configuration used in each case.



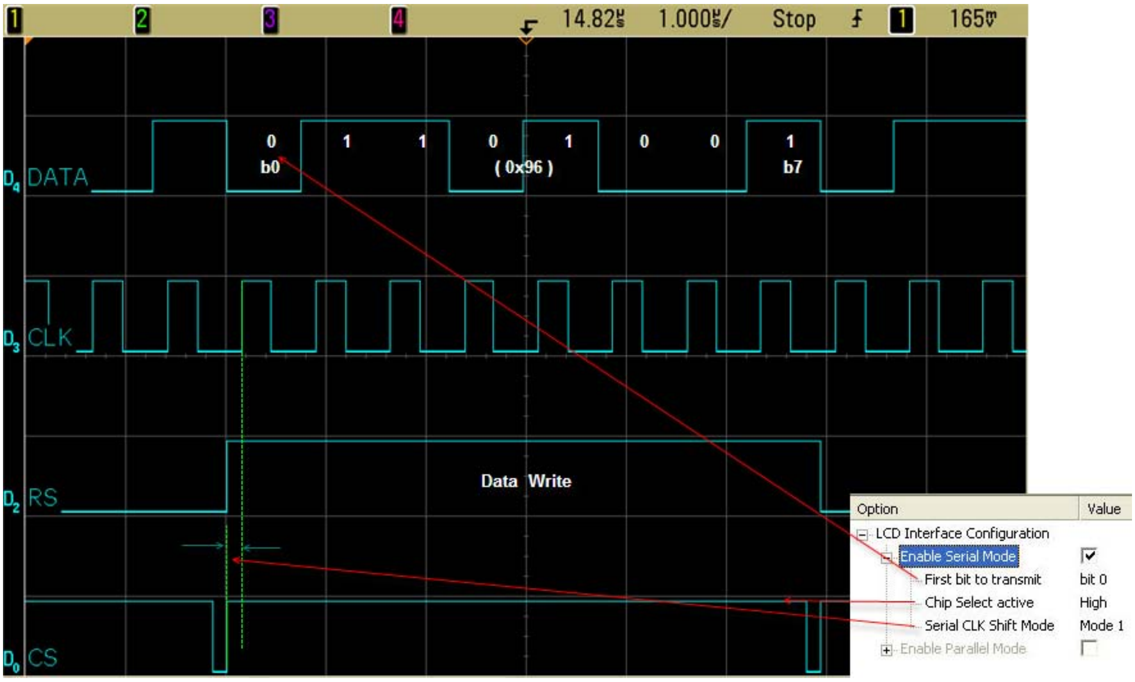
Option	Value
LCD Interface Configuration	
Enable Serial Mode	☑
First bit to transmit	bit 0
Chip Select active	High
Serial CLK Shift Mode	Mode 0
Enable Parallel Mode	□

Fig 7. LCD interface in Serial mode, showing both an Instruction and Data Write cycle



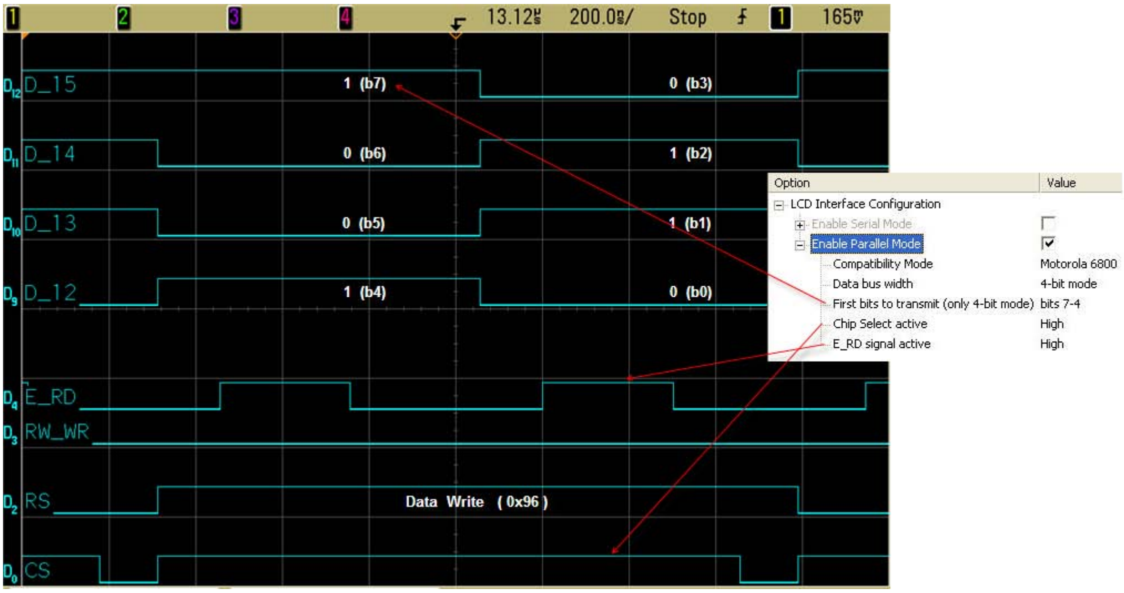
Option	Value
⊟ LCD Interface Configuration	
⊟ Enable Serial Mode	☑
First bit to transmit	bit 7
Chip Select active	Low
Serial CLK Shift Mode	Mode 0
⊞ Enable Parallel Mode	□

Fig 8. Serial mode with Chip Select active LOW and bit 7 transmitted first



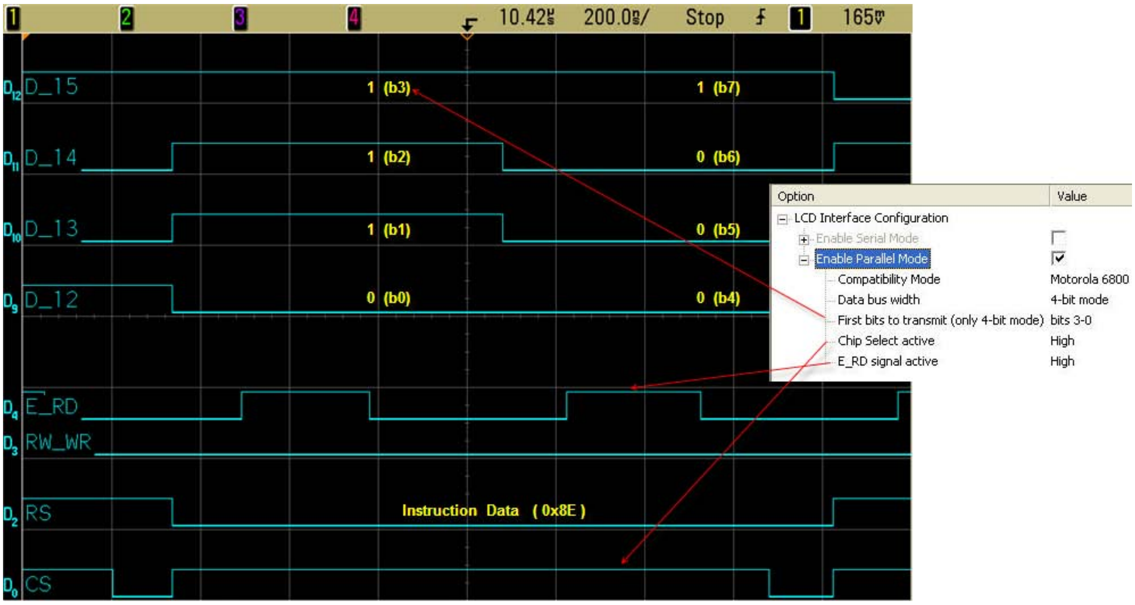
Option	Value
⊟ LCD Interface Configuration	
⊟ Enable Serial Mode	☑
First bit to transmit	bit 0
Chip Select active	High
Serial CLK Shift Mode	Mode 1
⊞ Enable Parallel Mode	□

Fig 9. Serial mode showing CLK shifted 25 % (Mode 1)



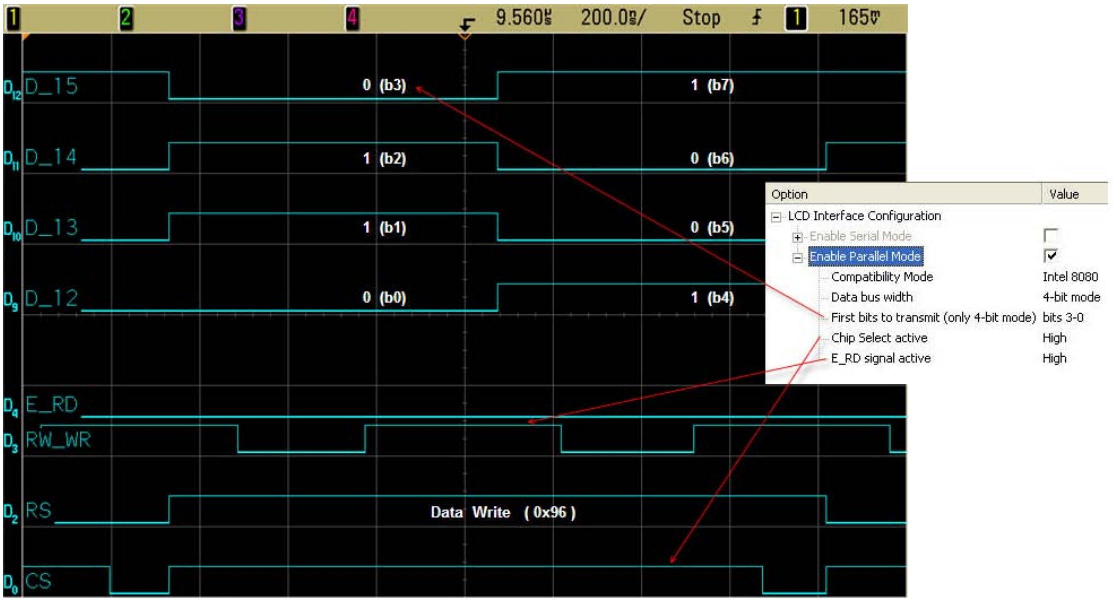
Option	Value
⊟ LCD Interface Configuration	
⊞ Enable Serial Mode	□
⊟ Enable Parallel Mode	☑
Compatibility Mode	Motorola 6800
Data bus width	4-bit mode
First bits to transmit (only 4-bit mode)	bits 7-4
Chip Select active	High
E_RD signal active	High

Fig 10. LCD interface in 4-bit Parallel mode, Motorola 6800 selected, showing a Data write cycle



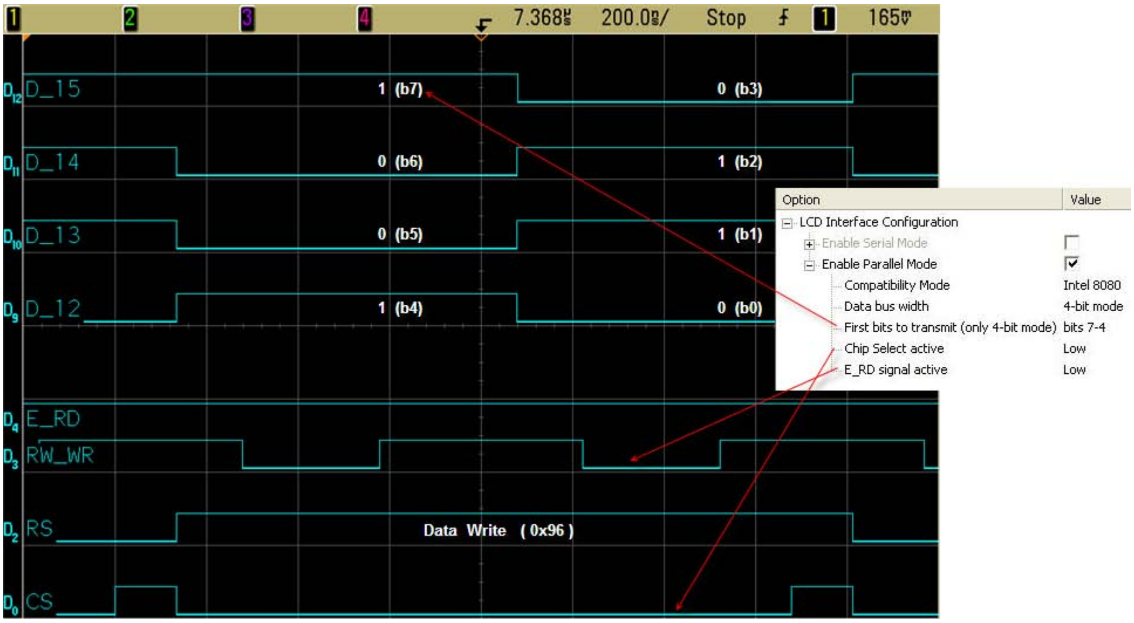
Option	Value
⊟ LCD Interface Configuration	
⊞ Enable Serial Mode	□
⊟ Enable Parallel Mode	☑
Compatibility Mode	Motorola 6800
Data bus width	4-bit mode
First bits to transmit (only 4-bit mode)	bits 3-0
Chip Select active	High
E_RD signal active	High

Fig 11. Same configuration as in Fig 10, but transmitting bits 3-0 first



Option	Value
LCD Interface Configuration	
Enable Serial Mode	□
Enable Parallel Mode	☑
Compatibility Mode	Intel 8080
Data bus width	4-bit mode
First bits to transmit (only 4-bit mode)	bits 3-0
Chip Select active	High
E_RD signal active	High

Fig 12. LCD interface in 4-bit Parallel mode, Intel 8080 mode selected, showing a Data Write cycle



Option	Value
LCD Interface Configuration	
Enable Serial Mode	□
Enable Parallel Mode	☑
Compatibility Mode	Intel 8080
Data bus width	4-bit mode
First bits to transmit (only 4-bit mode)	bits 7-4
Chip Select active	Low
E_RD signal active	Low

Fig 13. Same configuration as in Fig 12 but with Chip select active LOW, E_RD (and RW_WR) signal active LOW, and bits 7-4 transmitted first

5. Conclusion

The LCD interface of the LPC313x/4x/5x is very flexible as it allows connection with many different LCD display types, such as serial or parallel. A broad range of these displays is supported by adjusting the speed in software. In addition, DMA transfers, which help to reduce CPU loading from LCD refreshing tasks, can be used.

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer’s own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

6.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

7. Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

For more information, please visit: http://www.nxp.com
For sales office addresses, email to: salesaddresses@nxp.com

Date of release: 21 October 2009

Document identifier: AN10880_1