MPC56xxB LCD segment driver

Driving LCD displays with standard GPIO

by: Viktor Felinger

Table Of Contents

1 Introduction

MPC56xxB LCD driver is a collection of header and source files, in the C programming language, that enables driving of bare twisted nematic (TN) and super twisted nematic (STN) liquid crystal displays (LCDs) with multiple backplane electrodes. The driver is optimized for low cost display driving solutions and offers two waveform generation methods:

1. use of GPIO pins and 2 external resistors per backplane electrode
2. use of GPIO pins and external RC networks to generate the required waveforms. Based on the solutions presented in this application note, customers can deploy MPC56xxB devices as a cost effective replacement for a microcontroller with LCD interface or to completely replace an external LCD

controller.

For more reference on TN LCDs, refer to AN3219 from http://www.freescale.com

2 Driving TN and STN LCDs Using General Purpose MPC56xxB Pins

Driving LCDs with a set of general purpose input/output (GPIO) pins on a microcontroller can result in a very cost effective and flexible solution. The microcontroller load resulting from the LCD drive is usually relatively low, and the cost of the specialized on-chip or off-chip LCD controller can be saved. Waveforms for statically driven LCDs require only two discrete voltage levels (see Figure 1). General purpose outputs of a typical microcontroller can therefore be connected directly to the LCD. However, statically driven LCDs are not suitable in many applications because of the high number of connections required, and their associated cost. Dynamically driven LCDs require more than two voltage levels to be present in the driving waveforms. In general, these are not trivial to obtain with the simple CMOS I/O structure used in the general purpose pins of modern microcontrollers. The following two methods present the ways to obtain additional voltage levels by using several GPIO pins and external network of passive components.

2.1 Creating Waveforms with Arbitrary Duty and Bias Ratios

A simple algorithm exists for creating waveforms with any duty and bias ratios. The algorithm is outlined in a pseudo-C programming language as given below. The voltage levels used by the algorithm range from 0 (corresponding to the lowest voltage) up to *b* (corresponding to the highest voltage). The algorithm does not create any link between the duty factor *n* and the bias factor *b*. This means that it is possible to create waveforms with 1/4 duty ratio and only 1/2 bias ratio. On the other hand, it is also possible to drive an LCD with only two backplane electrodes (1/2 duty ratio) with waveforms that make use of many voltage levels (for example, 1/5 bias ratio).

```
/* Even refresh cycle */
for (phase=0;phase<n;phase++) {
Drive electrode BPphase to voltage level 0
Drive all other BP electrodes to voltage level b-1
Drive FP electrodes of visible segments to voltage level b
Drive FP electrodes of invisible segments to voltage level b-2
}
/* Odd refresh cycle */
for (phase=0;phase<n;phase++) {
Drive electrode BPphase to voltage level b
Drive all other BP electrodes to voltage level 1
Drive FP electrodes of visible segments to voltage level 0
Drive FP electrodes of invisible segments to voltage level 2
}
```

Algorithm for Creating Waveforms with Arbitrary Duty and Bias Ratios

2.2 Waveform generation based on DMA

Waveform generation on a microcontroller without a dedicated LCD interface needs to be driven either by the main core, a co-processor or a set of modules setup in a way to allow update of GPIO states based on predefined timing. The waveform generation methods in this application note are based on a setup comprising of MPC56xxB DMA, PIT, SIUL and eMIOS units (detailed setup description on following pages). The DMA driven approach has following advantages as compared to a processor or co-processor solution (refer to AN3219 available from http://www.freescale.com):

- no code has to be executed during waveform generation
    - Advantage: more computing power for the application
- no interrupts for waveform generation
    - Advantage: more flexibility for application - no attention has to be paid to interrupt handling related to LCD interface.
- Accurate Phase and cycle Timing by PIT triggering a linked DMA chain
    - Advantage of the above is all timings which need to be considered for waveform generation are fixed by chip design. This solves the problem of any signal delay for particular electrode and avoids DC voltage offset between backplanes and frontplanes.

2.3 Method 1 for waveforms with 1/2 Bias Ratio

In this method, the additional voltage levels can be obtained by using several GPIO pins and/or external resistor networks. Waveforms with 1/2 bias ratio (see Figure 1)use three voltage levels for the backplane electrodes, and only two voltage levels for the frontplane electrodes, however it is possible to extend the driver and use three voltage levels for frontplanes as well. This means that the frontplane electrodes of the LCD can be connected directly to GPIO pins of the microcontroller. To keep the duty ratio favorable, dynamically driven LCDs typically feature more frontplane electrodes than backplane electrodes. Therefore, most LCD electrodes can be driven directly by the GPIO pins, and only few backplane electrodes require some additional circuitry.

2.3.1 Voltage levels

The third voltage level required for driving the backplane electrodes with 1/2 bias ratio waveforms can be obtained conveniently by making use of the input capability of the GPIO pins. MPC56xxB GPIO pin represents a high impedance load when output and input buffers are switched off. In such a state, a simple resistor network can be used to determine the voltage level on the pin. Whenever the GPIO pin is configured as an output, it overdrives the resistor network and applies a high or low voltage level to the backplane electrode. The resistor arrangement between microcontroller GPIO pins and LCD backplane electrodes and GPIO pin states (PCR[OBE] – Pad Configuration Output Buffer enable/disable, GPDO – GPIO output register) is shown in Figure 1.

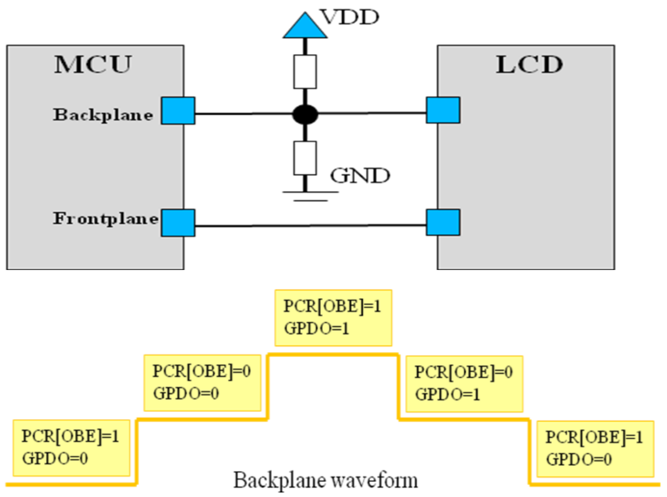

Figure 1. ½ bias waveform resistor arrangement and GPIO pin states

The values of the resistors should not be <3.3kΩ, because the current needs to be able to charge / discharge the display capacitance in an acceptable rise/ fall time to avoid contrast problems.

The voltage level imposed by the resistive divider will be influenced only by leakage currents of the microcontroller and the LCD, which are typically in the low µA range, or lower. Another factor influencing the choice of values for the resistors is the time constant of the circuit formed by the resistors and the stray capacitance of the connection between the microcontroller and the LCD. The stray capacitance can be significant in cases where the LCD is connected using long PCB tracks, a connector or a flat cable. For the waveforms to maintain their properties, the time constant of the RC circuit must be small compared to the length of one phase (*t*).

2.3.2 Generating 1/2 Bias Ratio Waveforms

MPC56xxB driver uses below modules for ½ bias ratio waveform generation:

- Periodic Interrupt Timer (PIT) - triggers DMA accesses
- Direct Memory Access (DMA) - DMA copy values from RAM locations to PIT load value register, pad configuration registers (PCR) and masked parallel data output register (MPGPDO). Switching between different memory locations is based on DMA channel linking capabilities.
- System Integration Unit (SIUL) – pad configuration registers (PCR) enables, disables output driver for backplanes. MPC56xxB offers masked parallel data output registers (MPGPDO) which allow the update of selected outputs (flexible assignment of GPIO frontplane pins).

Figure below shows the basic principle of bias 2 waveform generation.

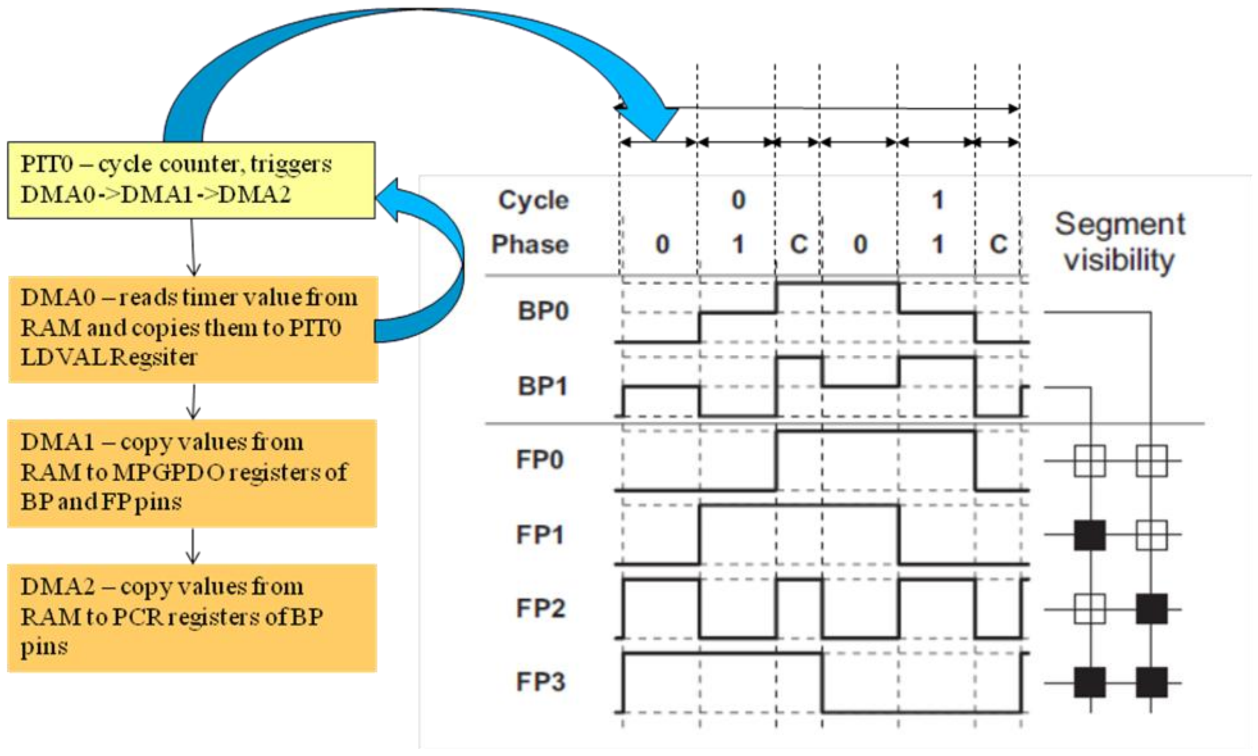
PIT0 – cycle counter, triggers DMA0->DMA1->DMA2
DMA0 – reads timer value from RAM and copies them to PIT0 LDVAL Regsiter
DMA1 – copy values from RAM to MPGPDO registers of BP and FP pins
DMA2 – copy values from RAM to PCR registers of BP pins
Cycle
Phase
0
1
0
1
C
0
1
C
Segment visibility
BP0
BP1
FP0
FP1
FP2
FP3

Figure 2. PIT trigger and DMA link flow for ½ bias waveform

2.4 Method 2 - PWM based

The LCD driver uses a second method for optimal VONRMS/VOFFRMS ratio for LCDs with up to 6 backplanes. As mentioned earlier in this application note, for best possible VONRMS/VOFFRMS ratio and as a result better contrast, a higher number of discrete voltage levels deliver better contrast results when driving LCD displays. For LCD segments from 3 up to 6 backplanes a waveform with 4 different voltage levels (1/3 bias ratio) deliver best results (see Table 1).

2.4.1 Voltage levels

Waveform generation with standard MPC56xxB GPIO for bias 3 is based on a combination of internal microcontroller modules which operate independent of CPU and external components. To generate different voltage levels an external RC circuit converts a digital PWM signal to an analog voltage level needed for backplane and frontplane waveform generation. The RC values have to be carefully selected to allow:

- Selection of typical values available on the market.
  Advantage: low cost components
- Small footprint (down to form factor 0402).
  Advantage: better match possible limited space on the PCB
- Good quality waveform.
  Advantage: lower EMC noise due to smooth rise/fall times

Figure below shows the MCU to LCD connection, PWM signal at MCU output and corresponding voltage levels at LCD input.

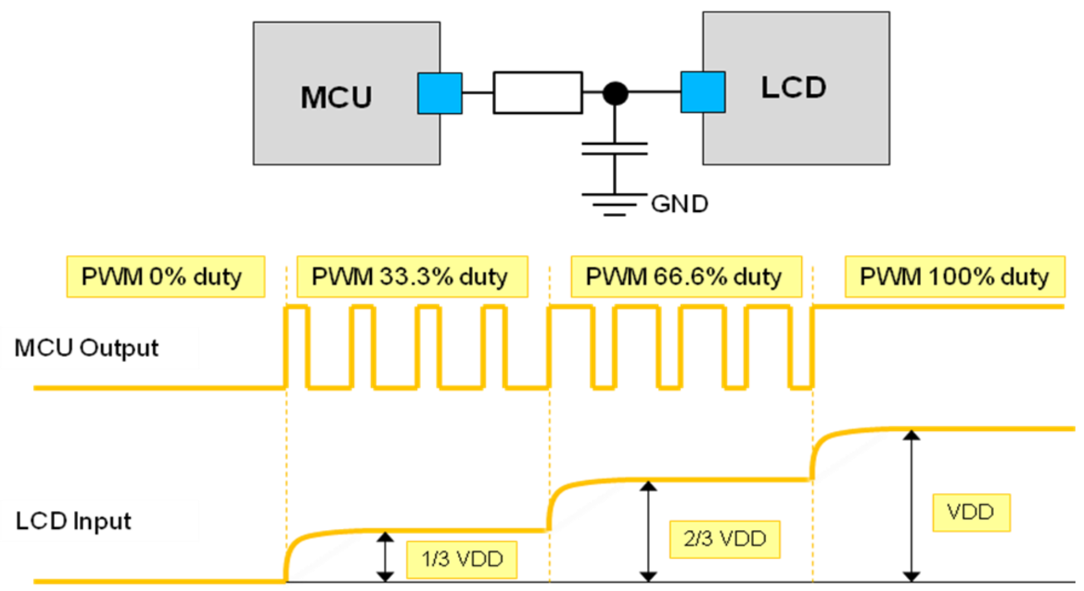

Figure 3. PIT trigger and DMA link flow for ½ bias waveform

2.4.2 Generating Bias Ratio Waveforms

MPC56xxB driver uses below modules for waveforms with 3 to 6 voltage levels:

- Periodic Interrupt Timer (PIT) - one PIT channel is used for time partitioning of the waveform which includes phase and optional contrast adjustment cycles. PIT channel triggers a DMA channel.
- Direct Memory Access (DMA) - DMA copies values from RAM locations to PIT load value register and eMIOS register for PWM duty cycle. Switching between different memory locations is based on DMA channel linking capabilities.
- System Integration Unit (SIUL) – pad configuration registers (PCR) enables, disables output driver for backplanes. Offers masked parallel data output registers (MPGPDO) which allow the update of selected outputs (flexible asisgnment of GPIO frontplane pins).

Figure below shows basic principle of PIT triggering and DMA channel linking flow for 1/3 bias waveform generation.

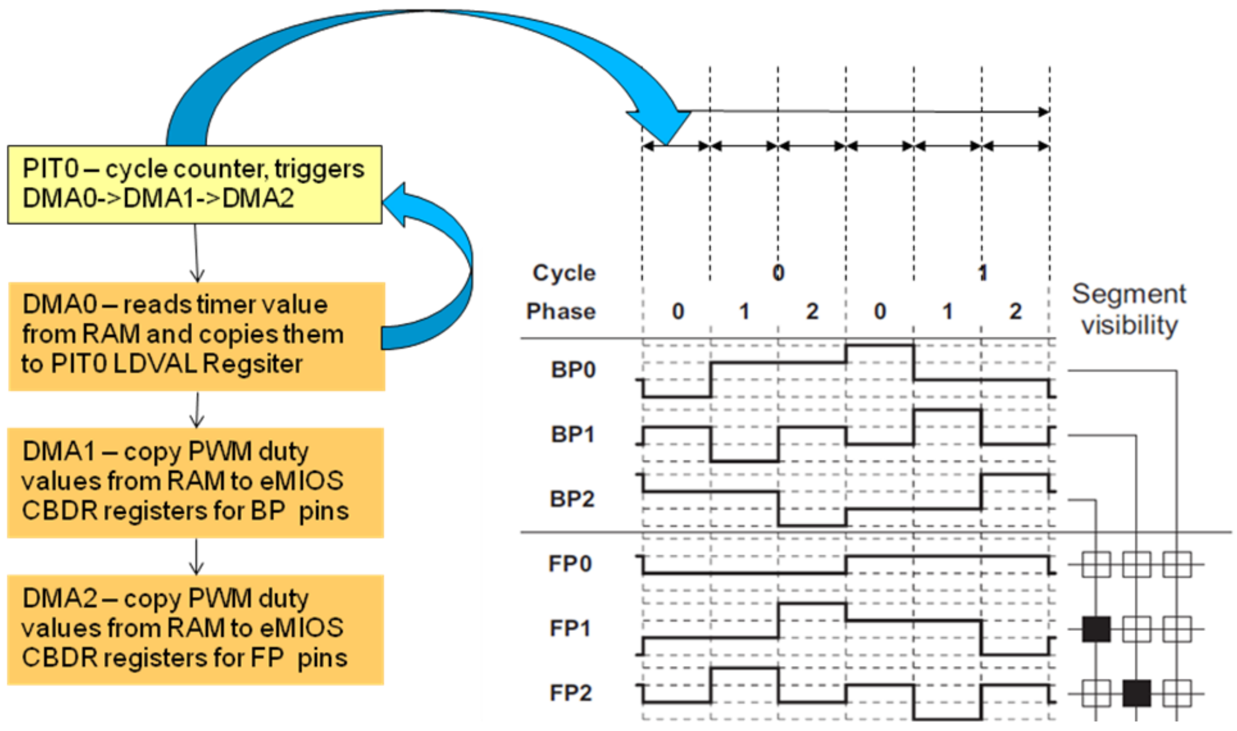
PIT0 – cycle counter, triggers DMA0->DMA1->DMA2
DMA0 – reads timer value from RAM and copies them to PIT0 LDVAL Regsiter
DMA1 – copy PWM duty values from RAM to eMIOS CBDR registers for BP pins
DMA2 – copy PWM duty values from RAM to eMIOS CBDR registers for FP pins
Cycle
Phase
0
1
0
1
2
0
1
2
Segment visibility
BP0
BP1
BP2
FP0
FP1
FP2

Figure 4. PIT trigger and DMA link flow for 1/3 bias waveform

After each DMA read/write transaction, the source and destination address can be adjusted by a fixed offset only. For this reason, it is recommended to assign one group of eMIOS channels adjacent to each other in the memory map to backplane pins and the other group of eMIOS channels adjacent to each other to frontplane pins. If this is not possible and the eMIOS channel in between is operating in a mode which uses CBDR register, than a new DMA channel should added to the DMA chain. Figure below explains the concept based on a simple use case:

Use case:

- Group1: DMA1 for Backplanes, DMA1 is linked by DMA for PIT
  - eMIOS ch 28 / 29 / 30 / 31, OPWMT mode
- Group2: DMA2 for Frontplanes, DMA2 is linked by DMA1
  - eMIOS ch 2-7 / 9, OPWMT mode
- Group3: DMA2 for Frontplanes, DMA3 is linked by DMA2
  - eMIOS ch 11-13, OPWMT mode
- Group4: non-LCD channel which requires CBDR register -> DMA write would corrupt the content
  - eMIOS ch 0 / 1 / 10, i.e. OPWMB mode
- Group5: non-LCD channel which do not require CBDR register -> DMA write is don‘t care
  - eMIOS ch 8, i.e. MCB mode

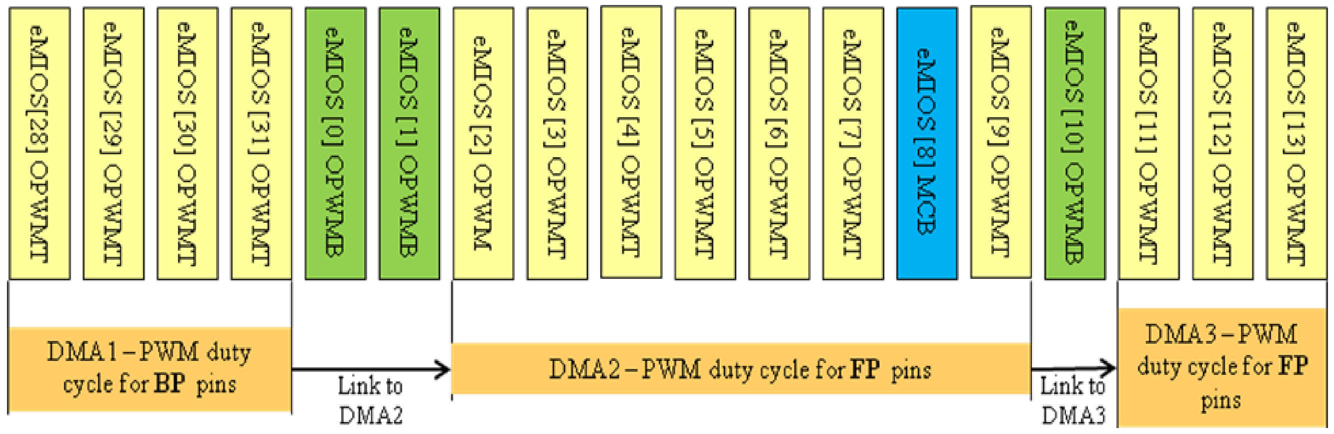

Figure 5. Example of PWM channel assignment and corresponding DMA flow

2.5 Suitability of Waveforms

The above sections explain two different methods of generating waveforms when driving an LCD using GPIO pins of a microcontroller. Using the 1/2 bias waveforms based on method 1 are optimal when driving an LCD with two backplane electrodes (1/2 duty ratio). When the 1/2 bias ratio waveforms are used for LCDs with three and more backplane electrodes, the VONRMS/VOFFRMS ratio is lower, compared to the optimum waveforms. For Bias 2 the voltage band between VONRMS and VOFFRMS is reduced by 20% in the case of a 1/3 duty ratio, by 28% in the case of a 1/4 duty ratio, and by 32% in case of a 1/5 duty ratio.
Whether method 1 waveforms are suitable to drive LCDs with 3 or more backplanes depends on the display specification. Narrower band between VONRMS and VOFFRMS may require a more accurate temperature compensation to maintain the best possible optical properties of the display over a wide temperature range. This depends on properties of the particular LCD — variation of VS and VT with temperature can usually be found in data sheets provided by LCD manufacturers. Advantage of method 1 is that only a few external resistor are required (2*number of backplanes).

For the best possible VONRMS/VOFFRMS ratio, a higher number of discrete voltage levels must be used when driving LCDs with three and more backplane electrodes.
Method 2 based on PWM approach and external RC components delivers optimal VONRMS/VOFFRMS for bias 3 to 6 waveforms and as a result offer optimum contrast for backplane LCD displays from 3 up to 10 backplanes.

Figure below shows the number backplanes (n) and bias (b) configurations supported by this driver:

- n/b configurations covered by Method 1: shaded fields =2, n=2,3,4
- n/b configurations covered by Method 2: bold area, all fields
- Best VONRMS/VOFFRMS ratio : bold text

n / b	2	3	4	5	6	7	8
2	**2.236**	**2.236**	1.844	1.612	1.475	1.387	1.325
3	1.732	**1.915**	1.732	1.567	1.453	1.374	1.318
4	1.528	**1.732**	1.648	1.528	1.433	1.363	1.311
5	1.414	**1.612**	1.581	1.494	1.414	1.352	1.304
6	1.342	**1.528**	1.528	1.464	1.397	1.342	1.297
7	1.291	1.464	**1.483**	1.438	1.382	1.332	1.291
8	1.254	1.414	**1.446**	1.414	1.367	1.323	1.285
9	1.225	1.374	**1.414**	1.393	1.354	1.314	1.279
10	1.202	1.342	**1.387**	1.374	1.342	1.306	1.274

Table 1. VONRMS/VOFFRMS Ratio for Different Combinations of Duty and Bias

Some MPC56xxB microcontrollers have 2 voltage domains for GPIO supply. Please note that if the pads for LCD waveform generation belong to different voltage domains they should be at the same voltage level.

3 Driver Implementation Using MPC56xxB GPIO, PIT, DMA and PWM

The flow of data, between an LCD driver and the application that uses the driver, is in one direction only. The application generates the display data (i.e., tells the driver which segments should be visible and invisible), and the driver generates the appropriate waveforms. All data is written by the application and read by the driver. The concept ensures that the application is not interrupted by the waveform generation task and the application itself has only to update the display data in the system memory, as appropriate. From the application's point of view, the LCD driver behaves like a virtual LCD peripheral with a specific area of the system RAM used to share the display data.

3.1 Supported devices

The driver can run on all MPC56xxB devices with DMA unit. Table below summarizes the devices and packages the driver can run on as well as the maximum numbers of backplane and frontplane connections for each solution:

Device	Package	Solution 1 #GPIO	Solution 1 #backplanes+ #fronplanes	Solution2 eMIOS outputs	Solution 2 #backplanes+ #fronplanes
MPC5601D	LQFP64	45	45	12	12
MPC5602D	LQFP100	79	79	28	27*
MPC5605B	LQFP100	77	77	37	36*
MPC5605/6B	LQFP144	121	121	64	62*
MPC5605/6/7B	LQFP176	149	149	64	62*
MPC5644/5/6B	LQFP176	149	149	64	62*
MPC5644/5/6B	LQFP208	173	173	64	62*
MPC5644/5/6B	BGA256	199	199	64	62*

Table 2. MPC56xxB LCD driver support and maximum numbers of backplanes and frontplanes

NOTE

Please note that within a 32ch eMIOS block at least one channel is needed for timebase.

3.2 Structure of the driver

The driver is split into several files:
Main.c – main application function
LCD.c – contains all the functions to initialize and run LCD interface
init.c – includes functions for basic system initialization
dma.c – dmamux and dma initialization functions
pit.c – periodic interrupt timer initialization functions
emios.c – includes functions to setup the time base and PWM channels
LCD.h - user defined macros for mapping the appropriate resources to the driver

The driver functions use macros to access hardware resources of the microcontroller. All the application dependent macros are located in the "lcd.h" header file. It is possible, therefore, to customize the driver functions by editing the header file only.

3.3 Using the Driver

All hardware specific configurations of the driver are performed through modification of the macro definitions located in the "lcd.h" header file. The user is responsible for providing correct definitions for these hardware access macros. Descriptions of the individual macros and examples of their definitions are given in the following sections.

3.3.1 Define application specific parameters in LCD.h

Following code from LCD.h defines LCD related specific parameters:

```
#define PWM 1        // waveform generation method
// 0 –  method1 - 2 bias with 2 external resistors per backplane
// 1 –  method2 - all bias levels based on PWM with external RC components
```

```
#define PIT_CH_NR 0  // PIT channel number to be used for waveform timing
#define PIT_INIT_VALUE 0x1000  // this is the init value for PIT
//delay before starting the waveform genearion after init
#define DMA_CH_for_PIT 0 // DMA channel for copying waveform timing values
from Ram to PIT
#if EMIOS
#define DMA_CH_for_EMIOS_BP 1          // DMA channel for copying PWM duty
cycle values for backplanes from RAM to eMIOS
#define DMA_CH_for_EMIOS_FP 2 // DMA channel for copying PWM duty cycle
values for frotplanes from RAM to eMIOS

#else
#define DMA_CH_for_PCR 1      // DMA channel for copying pad configuration
data for backplanes from RAM to SIUL pad configuration registers
#define DMA_CH_for_MPGPDO 2   // DMA channel for copying pad configuration
data for backplanes from RAM to SIUL pad configuration registers
#endif

#define LCD_BP_CNT   4   // number of backplanes
#define LCD_FP_CNT   28  // number of frontplanes
#define LCD_IDLE_CNT 1   // number of contrast adjustment phases per refresh
cycle
#define CONTRAST_RMS_INCREASE 1        // '1' - voltage levels for BP and FP
will be set to opposite
// voltage levels during contrast adjustment phases to increase RMS
// '0' - voltage levels for BP and FP will be set to the same
// voltage levels during contrast adjustment phases to decrease RMS
                              // in this driver version this paremeter is
valid for method 2 only (EMIOS=1)
#define LCD_FRAME_FREQUENCY  60  // LCD refresh cycle frequency in Hz
#define MPGPDO_REG_CNT        4   // number of MPGPDO registers
#define BIAS_CNT                        3   // waveform type, voltage levels =
bias +1
#define emios_peripheral_prescaler      1   // peripheral set 3 prescaler
#define emios_predivider                1   // emios global prescaler value
#define PWM_period_cycles               99  // number of emios clock cycles
withon one PWM period
```

3.3.1.1 Application specific parameters for Method 1

Following steps are needed to adjust the driver to application specific connections between microcontroller and LCD pins:

1. Define the pad information for backplane ports

   `const vuint8_t BP_SIUL_INFO [LCD_FP_CNT][3]`

   this contant has following format:

[`PCR_NUMBER, MPGPDO_NUMBER, BIT_POSITION_IN_MPGPDO`]

PCR_NUMBER – as specified in datasheet, chapter package pinout and signal description

MPGPDO_NUMBER – number of masked parallel general perpuse data register (MPGPDO) the backplane port belongs to (please refer to corresponding reference manual)

BIT_POSITION_IN_MPGPDO – bit position within the MPGPDO register (please refer to corresponding reference manual)

2. Define the pad information for frontplane ports

```
const vuint8_t FP_SIUL_INFO [LCD_FP_CNT][3]
```

this contant has following format:

[`PCR_NUMBER, MPGPDO_NUMBER, BIT_POSITION_IN_MPGPDO`]

Same meaning as for backplanes (see above)

3. Define the data structure for masked parallel general perpuse data register (MPGPDO)

   Driver code contains some example templates for this structure. The mask bits for MCU ports connected to LCD should be enabled in all phases. All other mask bits should be disabled to avoid conflicts with application. The data output value for backplanes and frontplanes should be set to ‘0’ for all phases in cycle 0 and to ‘1’ for all phases in cycle 1. This makes sure all segments are invisible after initialization. If contrast adjustment phase is needed (LCD_IDLE_CNT=1) please consider:

   - To increase VONRMS and brighten the visible segments (please note that this will increase the VOFFRMS of invisible segments) :
     - Cycle0 contrast adjustment phase: set output to ‘1’ for backplanes and ‘0’ for frontplanes
     - Cycle1 contrast adjustment phase: set output to ‘0’ for backplanes and ‘1’ for frontplanes
   - To decrease the VOFFRMS and darken the invisible segments (please note that this will decrease the VONRMS of invisible segments)
     - Cycle0 contrast adjustment phase: set output to ‘1’ for backplanes and ‘1’ for frontplanes
     - Cycle1 contrast adjustment phase: set output to ‘0’ for backplanes and ‘0’ for frontplanes
   - Please note that duration of contrast adustment phase and thus the VONRMS/VOFFRMS can be changed by calling LCD_contrast function.

3.3.1.2 Application specific parameters for Method 2

Following steps are needed to adjust the driver to application specific connections between microcontroller and LCD pins:

1. Define the pad and PWM channel information for backplane ports

   `const vuint8_t BP_EMIOS_INFO [LCD_FP_CNT][3]`

   this contant has following format:

   [`PCR_NUMBER, EMIOS_CH_NUMBER, PCR_EMIOS_AF`]

   PCR_NUMBER – as specified in datasheet, chapter package pinout and signal description

   EMIOS_CH_NUMBER – eMIOS channel number

   PCR_EMIOS_AF – eMIOS alternate output function as specified in datasheet, chapter package pinout and signal description

2. Define the pad and PWM channel information for frontplane ports

   `const vuint8_t FP_EMIOS_INFO [LCD_FP_CNT][3]`

   this contant has following format:

   [`PCR_NUMBER, EMIOS_CH_NUMBER, PCR_EMIOS_AF`]

   Same meaning as for backplanes (see above)

3.3.2 Software Functions

void LCD_init(unsigned int bus_frequency, int contrast_adj)

- initializes the data, phase and contrast adjustment parameter in LCD_data structure. These parameters are dependent on the selected system frequency, the LCD frequency (defined in LCD.h) and optional duration of contrast adjustment phase which
  - ► *unsigned int bus_frequency* — This parameter specifies the frequency used by the on-chip timer, which is used to time the waveform generation process. The function uses this parameter to calculate the number of "ticks" the timer must count to time the normal and the contrast adjustment phases.
  - ► *int contrast_adj* — This parameter influences the length of the contrast adjustment phases. The contrast adjustment phases and the normal phases will be of the same length when contrast_adj is equal to zero. Values above zero extend the duration of the contrast adjustment phases; values below zero reduce it.

void LCD_contrast (unsigned int bus_frequency, int contrast_adj)

- this function can be called during run time to change the length of the contrast adjustment phases without the need to shutdown and restart the driver. It calculates and writes new values for period, phase time and contrast adjustment to LCD_data structure and updates the PIT DMA array by calling INIT_PIT_DATA_FOR_DMA().

void update_waveform (uint32_t BP_NUMBER, uint32_t FP_NUMBER, uint8_t visible)

- updates the corresponding DMA data structure for an LCD segment defined by BP and FP number depending on whether it should be visible or invisible

void INIT_WAVEFORM_FOR_DMA (void)

- Called by LCD_init() to create Data Structures for DMA
- Calls: INIT_PIT_DATA_FOR_DMA()
- Driver Option (#define PWM=0 in LCD.h) calls
  - ▶ INIT_BACKPLANE_PCR_DATA_FOR_DMA()
  - ▶ INIT_MPGPDO_DATA_FOR_DMA()
- DrDriver Option (#define PWM=1 in LCD.h) calls
  - ▶ INIT_EMIOS_DATA_FOR_DMA()

void INIT_MODULES_FOR_LCD(void)

- Called by LCD_init() to initialize all modules needed for LCD waveform generation
- Driver Option (#define PWM=0 in LCD.h)
  - ▶ Init Pad configuration registers for back- and frontplanes
- Driver Option (#define PWM=1 in LCD.h)
  - ▶ Init PWM timebase by calling EMIOS_0_BUS_B_init(2, PWM_period_cycles);
  - ▶ Init PWM channels for back- and frontplanes by calling EMIOS_0_OPWMT_CH_init(PCR_N, N, CADR, CBDR, ALTCADR)
- DMA_Init()
- PIT_CH_ENABLE(N, LDVAL)

void INIT_PIT_DATA_FOR_DMA (void)

- creates an array of timer values for waveform generation. PIT channel triggers the DMA which writes the new timer value from array into the LDVAL register to determine the duration of next waveform phase

void INIT_BACKPLANE_PCR_DATA_FOR_DMA(void)

- creates an array of pad configuration data to be copied to backplane PCR registers via DMA

void INIT_MPGPDO_DATA_FOR_DMA(void)

- creates an array based on PCR numbers for front and backplanes. Array is filled with values to be copied to Masked Parallel GPDO Data via DMA

void INIT_EMIOS_DATA_FOR_DMA(void)

- calculates the number of PWM cycles within one waveform phase and corresponding timer values for PWM period and duty cycles. Creates an array for

PWM duty cycle values based on the bias define parameter (#define BIAS_CNT in LCD.h). The duty cycle values stored in array structures are copied to eMIOS CBDR registers in each phase via DMA.

void LCD_ALL_PINS_LOW(void)

- drives all PB and FP pins to low level

void LCD_start(void)

- enables the LCD waveform generation

void LCD_stop(void)

- disables the LCD waveform generation

3.4 Driver Performance

The solutions offered here is DMA driven and does not require any code to be executed during waveform generation and as a result no attention has to be paid to interrupt handling related to LCD interface. This offloads the CPU from waveform generation tasks and leaves more performance for the application. DMA reads data from RAM and writes it through the AIPS bridge to corresponding PIT, SIUL or eMIOS register. Below calculation show the percentage of crossbar port bandwidth required by DMA for waveform generation:

Crossbar Bandwidth for Solution 1

Assumptions

- 120 Hz LCD display
- 3 backplanes, 30 frontplanes
- 1 idle cycle for contrast adjustment

Number of waveform phases

- (#BPs+#IDLE)*2 = (3+1)*2 = 8 Phases

Number of Read Writes

- PIT DMA – 1xRAM read, 1x AIPS write
- MPGPDO DMA – 4xRAM read, 4xAIPS writes
- PCR DMA – 2xRAM read, 2xAIPS writes
- Sum of RAM/AIPS reads/writes per Phase = 7
- Sum of RAM/AIPS reads/writes per second = 7reads * 8phases * 128Hz = 7,168 KHz

RAM port bandwidth calculations

- Crossbar Frequency = 64MHz
- Driver RAM bandwidth [%] = 7,168KHz / 64000KHz   = 0.11 %
- Driver AIPS bandwidth (1 wait state) [%] = (7,168KHz *2) / 64000KHz   = 0.22 %

Crossbar Bandwidth for Solution 2 (PWM)

Assumptions

- 120 Hz LCD display

- 4 backplanes, 40 frontplanes

Number of waveform phases

- (#BPs)*2 = (4)*2 = 8 Phases

Number of Read Writes

- PIT DMA – 1xRAM read, 1x AIPS write
- eMIOS DMA – 44xRAM read, 44xAIPS writes
- Sum of RAM/AIPS reads/writes per Phase = 45
- Sum of RAM/AIPS reads/writes per second = 45reads * 8phases * 128Hz = 46,08 KHz

RAM port bandwidth calculations

- Crossbar Frequency = 64MHz
- Driver RAM bandwidth [%] = 46,08KHz / 64000KHz = 0.72 %
- Driver AIPS bandwidth (1 wait state) [%] = (46,08KHz *2) / 64000KHz = 1.44 %

Above calculations show that the percentage of LCD related transactions on the crossbar are in a very low range. For real world application, performance decrease can be considered as negligible as for code executed out of Flash, the required bandwidth for RAM and AIPS ports will typically be significantly lower than 50%.

3.5 Driver Resources

Table below shows the driver resources for both solutions:

Unit	Method 1 (external R)	Method 2 (PWM+external RC)
Flash	9.1K	8.7K
RAM	1.4K	1.5K
DMA ch	3	3
PIT ch	1	1
GPIO	depending on #BP and #FP	depending on #BP and #FP

Table 3. Driver resources

4 References

1. “Twisted nematic liquid crystal display”, Stephen Palmer, LC-TEC Displays AB, 2005
2. “100 Years of Commercial Liquid-Crystal Materials”, Werner Becker and Hans-Juergen Lemp, Merck & Co Inc., 2004

How to Reach Us:

Home Page:
www.freescale.com

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

AN4702
Rev 0
04/2013