Section 43. Graphics Controller Module (GFX)

HIGHLIGHTS

This section of the manual contains the following topics:

43.1 INTRODUCTION

The Graphics Controller (GFX) module is specifically designed to directly interface with display glasses, with built-in analog drive, to individually control pixels in the screen. The module also provides accelerated rendering of vertical and horizontal lines, rectangles, copying of a rectangular area between different locations on the screen, drawing texts and decompressing compressed data. The use of the accelerated rendering is performed using command FIFO. Once initiated, the hardware will perform the rendering freeing up the CPU for other tasks. Software can either poll the status or use interrupts to continue rendering succeeding shapes.

43.1.1 GFX Module Features

The GFX module supports three categories of display glasses:

- Monochrome STN
- Color STN
- Color TFT

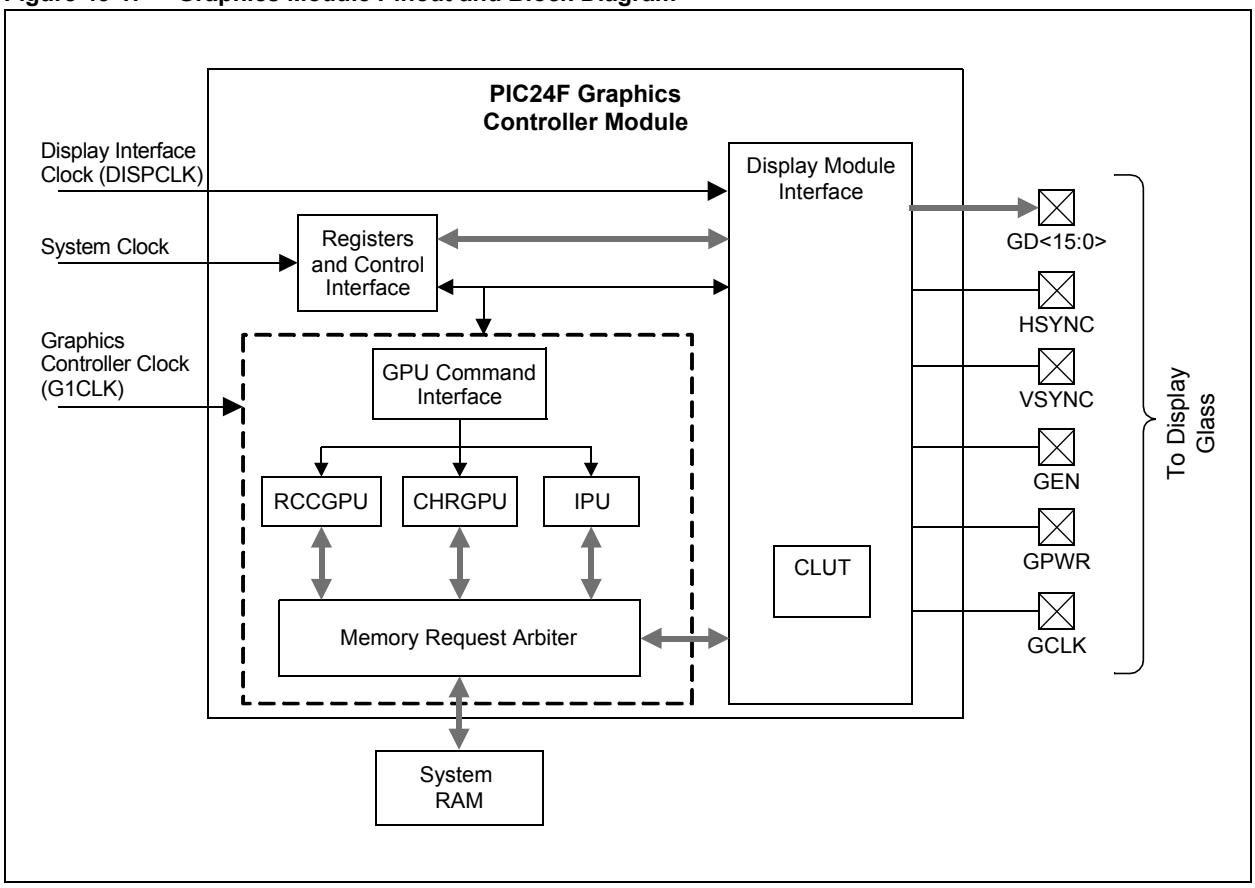
Programmable vertical and horizontal synchronization signals' timing is provided to meet the display's timing requirements.

The GFX module includes the following features:

- Graphics Hardware Accelerators:
  - Character Graphics Processing Unit (CHRGPU)
  - Rectangle Copy Graphics Processing Unit (RCCGPU)
  - Inflate Processing Unit (IPU)
- 256 Color Look-up Table (CLUT) Entries
- Supports 1/2/4/8/16 bits-per-pixel (bpp) Color Depth
- Programmable Display Resolution
- Supports the following Display Interfaces:
  - 4/8/16-Bit Monochrome STN
  - 4/8/16-Bit Color STN
  - 9/12/18/24-Bit Color TFT (18 and 24-bit displays are connected as 16-bit 5-6-5 RGB color format)

Figure 43-1 illustrates the GFX module pinout and image buffer sources. In some devices, system RAM can also reside externally to the device. Refer to the applicable device data sheet for details.

Figure 43-1: Graphics Module Pinout and Block Diagram



43.2 MODULE REGISTERS

The GFX module includes the following control and status registers:

- G1CMDL and G1CMDH – GPU Command Low/High Register (Register 43-1 and Register 43-2)

  These registers form the 32-bit command field. Commands written to these registers are loaded to the command FIFO.
- G1CON1 – Display Control Register 1 (Register 43-3)

  This register controls the enabling of the GFX module, sets the watermark level that triggers the interrupt for a number of queued commands in the command FIFO and sets up the color depth of the GPU operations in bits-per-pixel (bpp). This register also holds the status of the current queued commands in the 16-deep command FIFO.
- G1CON2 – Display Control Register 2 (Register 43-4)

  This register sets up the type of display glass hooked up to the module, the display color depth in bits-per-pixel, the display data stagger control to reduce simultaneous switching output noise and the type of test pattern that will be generated on the screen when in test mode.
- G1CON3 – Display Control Register 3 (Register 43-5)

  This register controls the display power signal, polarities and enabling of the display clock, vertical and horizontal synchronization, and data enable signals.
- G1STAT – Status Register (Register 43-6)

  This register contains the status of the GPUs, vertical/horizontal blanking, command watermark level and command FIFO.
- G1IE – Interrupt Enable Register (Register 43-7)

  This register contains the control bits to enable the different module's interrupt sources that go to the CPU.
- G1IR – Interrupt Status Register (Register 43-8)

  This register contains the current status of the module's interrupt sources.
- G1W1ADRL and G1W1ADRH – GPU Work Area 1 Start Address Register Low/High (Register 43-9 and Register 43-10)

  These registers define Work Area 1; they specifically define the start of the memory address on which the GPU will operate. All GPUs use these two registers to define their work areas.
- G1W2ADRL and G1W2ADRH – GPU Work Area 2 Start Address Register Low/High (Register 43-11 and Register 43-12)

  These registers define Work Area 2; they specifically define the start of the memory address on which the GPU will operate. IPU and RCCGPU use these two registers to define its 2nd work area.
- G1PUW – GPU Work Area Width Register (Register 43-13)

  This register defines the GPU Work Areas 1 and 2 width in pixels.
- G1PUH – GPU Work Area Height Register (Register 43-14)

  This register defines the GPU Work Areas 1 and 2 height in pixels.
- G1DPADRL and G1DPADRH – Display Buffer Start Address Register Low/High (Register 43-15 and Register 43-16)

  These registers define the start address of the display buffer.
- G1DPW – Display Buffer Width Register (Register 43-17)

  This register defines the display buffer width in pixels.
- G1DPH – Display Buffer Height Register (Register 43-18)

  This register defines the display buffer height in pixels.
- G1DPWT – Display Total Width Register (Register 43-19)

  This register defines the total width of the display in pixels. This parameter may be greater than the actual pixels displayed because of the vertical, non-display blanking requirement in TFT displays.

- G1DPHT – Display Total Height Register (Register 43-20)

  This register defines the total height of the display in pixels. This parameter may be greater than the actual pixels displayed because of the horizontal, non-display blanking requirement in TFT displays.
- G1ACTDA – Active Display Area Register (Register 43-21)

  This register controls the number of lines before the first visible line is drawn and the number of display clocks before the first visible pixel is drawn.
- G1HSYNC – Horizontal Synchronization Control Register (Register 43-22)

  This register controls the horizontal synchronization pulse width and the start of the synchronization signal.
- G1VSYNC – Vertical Synchronization Control Register (Register 43-23)

  This register controls the vertical synchronization pulse width and the start of the synchronization signal.
- G1DBLCON – Display Blanking Control Register (Register 43-24)

  This register controls the number of lines between the start of the vertical blanking and the first displayed line, and the number of display clock cycles before data enable that indicates the first valid data for each line.
- G1CLUT – Color Look-up Table Control Register (Register 43-25)

  This register enables the CLUT usage, CLUT read/write enable bit, CLUT read trigger bit and the address of the CLUT entry that will be read or modified. This register also contains the CLUT read/write busy bit.
- G1CLUTWR – CLUT Memory Write Data Register (Register 43-26)

  This register specifies the data written to the CLUT memory with the address specified in the G1CLUT register.
- G1CLUTRD – CLUT Memory Read Data Register (Register 43-27)

  This register specifies the data read from the CLUT memory with the address specified in the G1CLUT register.
- G1MRGN – Interrupt Advance Register (Register 43-28)

  This register specifies the number of vertical and horizontal blanking bits that will trigger in advance the interrupts for the vertical and horizontal blanking signals.
- G1CHRX and G1CHRY – Character X and Y Coordinate Print Position Register (Register 43-29 and Register 43-30)

  These registers specify the X and Y coordinate position of the character to be rendered in the image buffer by the CHRGPU.
- G1IPU – Inflate Processor Status Register (Register 43-31)

  This register contains the status bits associated with the inflate processor.
- G1DBEN – Data I/O Pad Enable Register (Register 43-32)

  This register controls the enabling of the I/O pads to switch to the display controller.

Register 43-1: G1CMDL: GPU Command Low Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCMD15	GCMD14	GCMD13	GCMD12	GCMD11	GCMD10	GCMD9	GCMD8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCMD7	GCMD6	GCMD5	GCMD4	GCMD3	GCMD2	GCMD1	GCMD0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **GCMD<15:0>:** Low GPU Command bits

The full 32-bit command is defined by G1CMDH and G1CMDL (GCMD<31:0>). Writes to this register will not trigger the loading of GCMD<31:0> to the command FIFO. For command FIFO loading, see the G1CMDH register description.

Register 43-2: G1CMDH: GPU Command High Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCMD31	GCMD30	GCMD29	GCMD28	GCMD27	GCMD26	GCMD25	GCMD24
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCMD23	GCMD22	GCMD21	GCMD20	GCMD19	GCMD18	GCMD17	GCMD16
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **GCMD<31:16>:** High GPU Command bits

The full 32-bit command is defined by G1CMDH and G1CMDL (GCMD<31:0>). A word write to the G1CMDH register triggers the loading of GCMD<31:0> to the command FIFO. Byte writes to G1CMDH are allowed, but only a high byte write will trigger the command loading to the FIFO. Low byte write to this register will only update the G1CMDH<7:0> bits.

Register 43-3: G1CON1: Display Control Register 1

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
G1EN	—	G1SIDL	GCMDWMK4	GCMDWMK3	GCMDWMK2	GCMDWMK1	GCMDWMK0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0
PUBPP2	PUBPP1	PUBPP0	GCMDCNT4	GCMDCNT3	GCMDCNT2	GCMDCNT1	GCMDCNT0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **G1EN:** Module Enable bit
1 = Display module enabled
0 = Display module disabled

bit 14 **Unimplemented:** Read as '0'

bit 13 **G1SIDL:** Stop in Idle bit
1 = Display module stops in Idle mode
0 = Display module does not stop in Idle mode

bit 12-8 **GCMDWMK<4:0>:** Command FIFO Watermark bits
Sets the command watermark level that triggers the CMDLVIF interrupt and sets the CMDLV flag.
GCMDWMK<4:0> > 10000 = Reserved:
10000 = If the number of commands present in the FIFO goes from 16 to 15 commands, the CMDLVIF interrupt will trigger and the CMDLV flag will be set
01111 = If the number of commands present in the FIFO goes from 15 to 14 commands, the CMDLVIF interrupt will trigger and the CMDLV flag will be set
•
•
•
00001 = If the number of commands present in the FIFO goes from 1 to 0 commands, the CMDLVIF interrupt will trigger and the CMDLV flag will be set
00000 = The CMDLVIF interrupt will not trigger and the CMDLV flag will not be set

bit 7-5 **PUBPP<2:0>:** GPU bits-per-pixel Setting bits
Other = Reserved
100 = 16 bits-per-pixel
011 = 8 bits-per-pixel
010 = 4 bits-per-pixel
001 = 2 bits-per -pixel
000 = 1 bit-per-pixel

bit 4-0 **GCMDCNT<4:0>:** Command FIFO Occupancy Status bits
When FIFO is full, any additional commands written to the FIFO are discarded.
10000 = 16 commands present in the FIFO
01111 = 15 commands present in the FIFO
•
•
•
00001 = 1 command present in the FIFO
00000 = 0 commands present in the FIFO

Register 43-4: G1CON2: Display Control Register 2

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
DPGWDTH1	DPGWDTH0	DPSTGER1	DPSTGER0	—	—	DPTEST1	DPTEST0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
DPBPP2	DPBPP1	DPBPP0	—	—	DPMODE2	DPMODE1	DPMODE0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-14 **DPGWDTH<1:0>:** STN Display Glass Data Width bits

11 = Reserved
10 = 16 bits wide
01 = 8 bits wide
00 = 4 bits wide

These bits have no effect on TFT mode. The TFT display glass data width is always assumed to be 16 bits wide.

bit 13-12 **DPSTGER<1:0>:** Display Data Timing Stagger bits

Refer to section **43.4.4 "Display Data Stagger"** for details.

11 = Delays of the display data are staggered in groups:
Bit Group 0: 0 4 8 12 – not delayed
Bit Group 1: 1 5 9 13 – delayed by ½ GPUCLK cycle
Bit Group 2: 2 6 10 14 – delayed by full GPUCLK cycle
Bit Group 3: 3 7 11 15 – delayed by 1½ GPUCLK cycle
10 = Even bits of the display data are delayed by 1 full GPUCLK cycle; odd bits are not delayed
01 = Odd bits of the display data are delayed by ½ GPUCLK cycle; even bits are not delayed
00 = Display data timing is all synchronized on one GPUCLK edge

bit 11-10 **Unimplemented:** Read as '0'

bit 9-8 **DPTEST<1:0>:** Display Test Pattern Generator bits

Test patterns can be used to test the interface to the display glass without the need to set up memory interface and display buffer.

11 = Borders
10 = Bars
01 = Black screen
00 = Normal Display mode, test patterns are off

bit 7-5 **DPBPP<2:0>:** Display bits-per-pixel Setting bits

This setting must match the GPU bits-per-pixel set in the PUBPP<2:0> (G1CON1<7:5>) bits.

100 = 16 bits-per-pixel
011 = 8 bits-per-pixel
010 = 4 bits-per-pixel
001 = 2 bits-per-pixel
000 = 1 bit-per-pixel
Other = Reserved

bit 4-3 **Unimplemented:** Read as '0'

bit 2-0 **DPMODE<2:0>:** Display Glass Type bits

011 = Color STN type
010 = Mono STN type
001 = TFT type
000 = Display off
Other = Reserved

Register 43-5: G1CON3: Display Control Register 3

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	DPPINOE	DPPOWER
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DPCLKPOL	DPENPOL	DPVSPOL	DPHSPOL	DPPWROE	DPENOE	DPVSOE	DPHSOE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-10 **Unimplemented:** Read as '0'

bit 9 **DPPINOE:** Display Pin Output Pad Enable bit
DPPINOE is the master output enable and must be set to allow GDBEN<15:0>, DPENOE, DPPWROE, DPVSOE and DPHSOE to enable the associated pads.
1 = Enable display output pads
0 = Disable display output pads
Pins used by the signals are assigned to the next enabled module that uses the same pins. For data signals, GDBEN<15:0> can be used to disable or enable specific data signals while DPPINOE is set.

bit 8 **DPPOWER:** Display Power Sequencer Control bit
(Refer to **Section 43.7 "Display Power Sequencing"** for details)
1 = Set display power sequencer control port (GPWR) to '1'
0 = Set display power sequencer control port (GPWR) to '0'

bit 7 **DPCLKPOL:** Display Glass Clock (GCLK) Polarity bit
1 = Display latches data on positive edge of GCLK
0 = Display latches data on negative edge of GCLK

bit 6 **DPENPOL:** Display Enable Signal (GEN) Polarity bit
For TFT mode (DPMODE (G1CON2<2:0>) = 001):
1 = Active-high (GEN)
0 = Active-low ($\overline{\text{GEN}}$)
For STN mode (DPMODE (G1CON2<2:0>) = 010 or 011):
1 = GEN connects to shift clock input of the display (Shift Clock mode)
0 = GEN connects to MOD input of the display (Line/Frame Toggle mode)

bit 5 **DPVSPOL:** Display Vertical Synchronization (VSYNC) Polarity bit
1 = Active-high (VSYNC)
0 = Active-low ($\overline{\text{VSYNC}}$)

bit 4 **DPHSPOL:** Display Horizontal Synchronization (HSYNC) Polarity bit
1 = Active-high (HSYNC)
0 = Active-low ($\overline{\text{HSYNC}}$)

bit 3 **DPPWROE:** Display Power Sequencer Control Port (GPWR) Enable bit
1 = GPWR port enabled (pin controlled by DPPOWER (G1CON3<8>) bit)
0 = GPWR port disabled (pin can be used as ordinary I/O)

bit 2 **DPENOE:** Display Enable Port Enable (GEN) bit
1 = GEN port enabled
0 = GEN port disabled

bit 1 **DPVSOE:** Display Vertical Synchronization Port Enable bit
1 = VSYNC port enabled
0 = VSYNC port disabled

bit 0 **DPHSOE:** Display Horizontal Synchronization Port Enable bit
1 = HSYNC port enabled
0 = HSYNC port disabled

Register 43-6: G1STAT: STATUS Register

R-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
PUBUSY	—	—	—	—	—	—	—
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
IPUBUSY	RCCBUSY	CHRBUSY	VMRGN	HMRGN	CMDLV	CMDFUL	CMDMPT
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **PUBUSY:** Processing Units are Busy Status bit
This bit is logically equivalent to the ORed combination of IPUBUSY, RCCBUSY and CHRBUSY.
1 = At least one processing unit is busy
0 = None of the processing units is busy

bit 14-8 **Unimplemented:** Read as '0'

bit 7 **IPUBUSY:** Inflate Processing Unit Busy Status bit
1 = IPU is busy
0 = IPU is not busy

bit 6 **RCCBUSY:** Rectangle Copy Graphics Processing Unit Busy Status bit
1 = RCCGPU is busy
0 = RCCGPU is not busy

bit 5 **CHRBUSY:** Character Graphics Processing Unit Busy Status bit
1 = CHRGPU is busy
0 = CHRGPU is not busy

bit 4 **VMRGN:** Vertical Blanking Status bit
1 = Display interface is in the vertical blanking period
0 = Display interface is not in the vertical blanking period

bit 3 **HMRGN:** Horizontal Blanking Status bit
1 = Display interface is in the horizontal blanking period
0 = Display interface is not in the horizontal blanking period

bit 2 **CMDLV:** Command Watermark Level Status bit
The number of commands in the command FIFO changed from equal (=) to the command watermark value, to less than (<) the command watermark value set in the GCMDWMK (G1CON1<12:8>) register bits.
1 = Command in FIFO is less than the set GCMDWMK value
0 = Command in FIFO is equal to, or greater than, the set GCMDWMK value

bit 1 **CMDFUL:** Command FIFO Full Status bit
1 = Command FIFO is full
0 = Command FIFO is not full

bit 0 **CMDMPT:** Command FIFO Empty Status bit
1 = Command FIFO is empty
0 = Command FIFO is not empty

Register 43-7: G1IE: Interrupt Enable Register

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
PUIE	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IPUIE	RCCIE	CHRIE	VMRGNIE	HMRGNIE	CMDLVIE	CMDFULIE	CMDMPTIE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **PUIE:** Processing Units Complete Interrupt Enable bit
1 = Enables the PU complete interrupt
0 = Disables the PU complete interrupt

bit 14-8 **Unimplemented:** Read as '0'

bit 7 **IPUIE:** Inflate Processing Unit Complete Interrupt Enable bit
1 = Enables the IPU complete interrupt
0 = Disables the IPU complete interrupt

bit 6 **RCCIE:** Rectangle Copy Graphics Processing Unit Complete Interrupt Enable bit
1 = Enables the RCCGPU complete interrupt
0 = Disables the RCCGPU complete interrupt

bit 5 **CHRIE:** Character Graphics Processing Unit Complete Interrupt Enable bit
1 = Enables the CHRGPU complete interrupt
0 = Disables the CHRGPU complete interrupt

bit 4 **VMRGNIE:** Vertical Blanking Interrupt Enable bit
1 = Enables the vertical blanking period interrupt
0 = Disables the vertical blanking period interrupt

bit 3 **HMRGNIE:** Horizontal Blanking Interrupt Enable bit
1 = Enables the horizontal blanking period interrupt
0 = Disables the horizontal blanking period interrupt

bit 2 **CMDLVIE:** Command Watermark Interrupt Enable bit
1 = Enables the command watermark interrupt bit
0 = Disables the command watermark interrupt bit

bit 1 **CMDFULIE:** Command FIFO Full Interrupt Enable bit
1 = Enables the command FIFO full interrupt
0 = Disables the command FIFO full interrupt

bit 0 **CMDMPTIE:** Command FIFO Empty Interrupt Enable bit
1 = Enables the command FIFO empty interrupt
0 = Disables the command FIFO empty interrupt

Register 43-8: G1IR: Interrupt Status Register

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
PUIF	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IPUIF	RCCIF	CHRIF	VMRGNIF	HMRGNIF	CMDLVIF	CMDFULIF	CMDMPTIF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **PUIF:** Processing Units Complete Interrupt Flag bit
PUIF is an ORed combination of IPUIF, RCCIF and CHRIF.
1 = One or more PUs completed command execution (must be cleared in software)
0 = All PUs are Idle or busy completing command execution

bit 14-8 **Unimplemented:** Read as '0'

bit 7 **IPUIF:** Inflate Processing Unit Complete Interrupt Flag bit
1 = IPU completed command execution (must be cleared in software)
0 = IPU is Idle or busy completing command execution

bit 6 **RCCIF:** Rectangle Copy Graphics Processing Unit Complete Interrupt Flag bit
1 = RCCGPU completed command execution (must be cleared in software)
0 = RCCGPU is Idle or busy completing command execution

bit 5 **CHRIF:** Character Graphics Processing Unit Complete Interrupt Flag bit
1 = CHRGPU completed command execution (must be cleared in software)
0 = CHRGPU is Idle or busy completing command execution

bit 4 **VMRGNIF:** Vertical Blanking Interrupt Flag bit
1 = Display interface is in the vertical blanking period (must be cleared in software)
0 = Display interface is not in the vertical blanking period

bit 3 **HMRGNIF:** Horizontal Blanking Interrupt Flag bit
1 = Display interface is in the horizontal blanking period (must be cleared in software)
0 = Display interface is not in the horizontal blanking period

bit 2 **CMDLVIF:** Command Watermark Interrupt Flag bit
1 = Command watermark level is reached (must be cleared in software)
0 = Command watermark level is not yet reached

bit 1 **CMDFULIF:** Command FIFO Full Interrupt Flag bit
1 = Command FIFO is full (must be cleared in software)
0 = Command FIFO is not full

bit 0 **CMDMPTIF:** Command FIFO Empty Interrupt Flag bit
1 = Command FIFO is empty (must be cleared in software)
0 = Command FIFO is not empty

Register 43-9: G1W1ADRL: GPU Work Area 1 Start Address Register Low

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
W1ADR15	W1ADR14	W1ADR13	W1ADR12	W1ADR11	W1ADR10	W1ADR9	W1ADR8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
W1ADR7	W1ADR6	W1ADR5	W1ADR4	W1ADR3	W1ADR2	W1ADR1	W1ADR0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **W1ADR<15:0>:** GPU Work Area 1 Start Address Low bits
Work area address must point to an even byte address in memory.

Register 43-10: G1W1ADRH: GPU Work Area 1 Start Address Register High

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
W1ADR23	W1ADR22	W1ADR21	W1ADR20	W1ADR19	W1ADR18	W1ADR17	W1ADR16
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8 **Unimplemented:** Read as '0'

bit 7-0 **W1ADR<23:16>:** GPU Work Area 1 Start Address High bits
Work area address must point to an even byte address in memory.

Register 43-11: G1W2ADRL: GPU Work Area 2 Start Address Register Low

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
W2ADR15	W2ADR14	W2ADR13	W2ADR12	W2ADR11	W2ADR10	W2ADR9	W2ADR8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
W2ADR7	W2ADR6	W2ADR5	W2ADR4	W2ADR3	W2ADR2	W2ADR1	W2ADR0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **W2ADR<15:0>:** GPU Work Area 2 Start Address Low bits
Work area address must point to an even byte address in memory.

Register 43-12: G1W2ADRH: GPU Work Area 2 Start Address Register High

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
W2ADR23	W2ADR22	W2ADR21	W2ADR20	W2ADR19	W2ADR18	W2ADR17	W2ADR16
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8 **Unimplemented:** Read as '0'

bit 7-0 **W2ADR<23:16>:** GPU Work Area 2 Start Address High bits
Work area address must point to an even byte address in memory.

Register 43-13: G1PUW: GPU Work Area Width Register

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	PUW10	PUW9	PUW8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PUW7	PUW6	PUW5	PUW4	PUW3	PUW2	PUW1	PUW0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-11 **Unimplemented:** Read as '0'

bit 10-0 **PUW<10:0>:** GPU Work Area Width bits (in pixels)

Register 43-14: G1PUH: GPU Work Area Height Register

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	PUH10	PUH9	PUH8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PUH7	PUH6	PUH5	PUH4	PUH3	PUH2	PUH1	PUH0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-11 **Unimplemented:** Read as '0'

bit 10-0 **PUH<10:0>:** GPU Work Area Height bits (in pixels)

Register 43-15: G1DPADRL: Display Buffer Start Address Register Low

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DPADR15	DPADR14	DPADR13	DPADR12	DPADR11	DPADR10	DPADR9	DPADR8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DPADR7	DPADR6	DPADR5	DPADR4	DPADR3	DPADR2	DPADR1	DPADR0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-0 **DPADR<15:0>:** Display Buffer Start Address Low bits
Display buffer start address must point to an even byte address in memory.

Register 43-16: G1DPADRH: Display Buffer Start Address Register High

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DPADR23	DPADR22	DPADR21	DPADR20	DPADR19	DPADR18	DPADR17	DPADR16
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-8 **Unimplemented:** Read as ‘0’

bit 7-0 **DPADR<23:16>:** Display Buffer Start Address High bits
Display buffer start address must point to an even byte address in memory.

Register 43-17: G1DPW: Display Buffer Width Register

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	DPW10	DPW9	DPW8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DPW7	DPW6	DPW5	DPW4	DPW3	DPW2	DPW1	DPW0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-11 **Unimplemented:** Read as '0'

bit 10-0 **DPW<10:0>:** Display Frame Width bits (in pixels)

Register 43-18: G1DPH: Display BUFFER Height Register

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	DPH10	DPH9	DPH8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DPH7	DPH6	DPH5	DPH4	DPH3	DPH2	DPH1	DPH0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-11 **Unimplemented:** Read as '0'

bit 10-0 **DPH<10:0>:** Display Frame Height bits (in pixels)

Register 43-19: G1DPWT: Display Total Width Register

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	DPWT10	DPWT9	DPWT8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DPWT7	DPWT6	DPWT5	DPWT4	DPWT3	DPWT2	DPWT1	DPWT0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-11 **Unimplemented:** Read as '0'

bit 10-0 **DPWT<10:0>:** Display Total Width bits (in pixels)

Register 43-20: G1DPHT: Display Total Height Register

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	DPHT10	DPHT9	DPHT8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DPHT7	DPHT6	DPHT5	DPHT4	DPHT3	DPHT2	DPHT1	DPHT0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-11 **Unimplemented:** Read as '0'

bit 10-0 **DPHT<10:0>:** Display Total Height bits (in pixels)

Register 43-21: G1ACTDA: Active Display Area Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ACTLINE7	ACTLINE6	ACTLINE5	ACTLINE4	ACTLINE3	ACTLINE2	ACTLINE1	ACTLINE0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ACTPIX7	ACTPIX6	ACTPIX5	ACTPIX4	ACTPIX3	ACTPIX2	ACTPIX1	ACTPIX0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8 **ACTLINE<7:0>:** Number of Lines Before the First Active (displayed) Line bits
Typically, ACTLINE = VENST (G1DBLCON<15:8>). This register is added for versatility in the timing of the active lines. For TFT mode, the DPMODE bits (G1CON2<2:0>) = 001 and the minimum value is 2. For STN mode, the DPMODE bits (G1CON2<2:0>) = 010, 011, 100 and the minimum value is '0'.

bit 7-0 **ACTPIX<7:0>:** Number of Pixels Before the First Active (displayed) Pixel bits (in DISPCLKs)
Typically, ACTPIX = HENST (G1DBLCON<7:0>). This register is added for versatility in the timing of the active pixels. Note that the programmed value is computed in DISPCLK cycles. This value is dependent on DPGWDTH (G1CON2<15:14>). Refer to Figure 43-12 for details.

Register 43-22: G1HSYNC: Horizontal Synchronization Control Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
HSLEN7	HSLEN6	HSLEN5	HSLEN4	HSLEN3	HSLEN2	HSLEN1	HSLEN0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
HSST7	HSST6	HSST5	HSST4	HSST3	HSST2	HSST1	HSST0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8 **HSLEN<7:0>:** HSYNC Pulse-Width Configuration bits (in DISPCLKs)
DPHSOE (G1CON3<0>) must be set for the HSYNC signal to toggle; minimum value is '1'.

bit 7-0 **HSST<7:0>:** HSYNC Start Delay Configuration bits (in DISPCLKs)
This is the number of DISPCLK cycles from the start of horizontal blanking to the start of HSYNC active.

Register 43-23: G1VSYNC: Vertical Synchronization Control Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
VSLEN7	VSLEN6	VSLEN5	VSLEN4	VSLEN3	VSLEN2	VSLEN1	VSLEN0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
VSST7	VSST6	VSST5	VSST4	VSST3	VSST2	VSST1	VSST0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8 **VSLEN<7:0>:** VSYNC Pulse-Width Configuration bits (in lines)
DPVSOE (G1CON3<1>) must be set for VSYNC signal to toggle; minimum value is '1'.

bit 7-0 **VSST<7:0>:** VSYNC Start Delay Configuration bits (in lines)
This is the number of lines from the start of vertical blanking to the start of VSYNC active.

Register 43-24: G1DBLCON: Display Blanking Control Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
VENST7	VENST6	VENST5	VENST4	VENST3	VENST2	VENST1	VENST0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
HENST7	HENST6	HENST5	HENST4	HENST3	HENST2	HENST1	HENST0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8 **VENST<7:0>:** Vertical Blanking Start to First Displayed Line Configuration bits (in lines)
This is the number of lines from the start of vertical blanking to the first displayed line of a frame.

bit 7-0 **HENST<7:0>:** Horizontal Blanking Start to First Displayed Pixel Configuration bits (in DISPCLKs)
This is the number of DISPCLK cycles from the start of horizontal blanking to the first displayed pixel of each displayed line.

Register 43-25: G1CLUT: Color Look-up Table Control Register

R/W-0	R-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
CLUTEN	CLUTBUSY	—	—	—	—	CLUTTRD	CLUTRWEN
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CLUTADR7	CLUTADR6	CLUTADR5	CLUTADR4	CLUTADR3	CLUTADR2	CLUTADR1	CLUTADR0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **CLUTEN:** Color Look-up Table Enable Control bit
1 = Color look-up table enabled
0 = Color look-up table disabled

bit 14 **CLUTBUSY:** Color Look-up Table Busy Status bit
1 = A CLUT entry read/write access is being executed
0 = No CLUT entry read/write access is being executed

bit 13-10 **Unimplemented:** Read as '0'

bit 9 **CLUTTRD:** Color Look-up Table Read Trigger bit
Enabling this bit will trigger a read to the CLUT location determined by the CLUTADR bits (G1CLUT<7:0>) with CLUTRWEN enabled.
1 = CLUT read trigger enabled (must be cleared in software after reading data in G1CLUTRD register)
0 = CLUT read trigger disabled

bit 8 **CLUTRWEN:** Color Look-up Table Read/Write Enable Control bit
This bit must be set when reading or modifying entries on the CLUT and it must also be cleared when CLUT is used by the display controller. Refer to **Section 43.3.4.1 "Reading and Writing CLUT Entries"** for details.
1 = Color look-up table read/write enabled; display controller cannot access the CLUT
0 = Color look-up table read/write disabled; display controller can access the CLUT

bit 7-0 **CLUTADR<7:0>:** Color Look-up Table Memory Address bits

Register 43-26: G1CLUTWR: CLUT Memory Write Data Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CLUTWR15	CLUTWR14	CLUTWR13	CLUTWR12	CLUTWR11	CLUTWR10	CLUTWR9	CLUTWR8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CLUTWR7	CLUTWR6	CLUTWR5	CLUTWR4	CLUTWR3	CLUTWR2	CLUTWR1	CLUTWR0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **CLUTWR<15:0>:** Color Look-up Table Memory Write Data bits

A word write or a high byte write to this register triggers a write to the CLUT memory at the address pointed to by the CLUTADR bits. Low byte write to this register will only update the G1CLUTWR<7:0> bits and no write to CLUT memory will be triggered. During power-up and power-down of the display, the most recent data written to this register will be used to control the timing of the GPWR signal. Refer to **Section 43.7 "Display Power Sequencing"** for details.

Register 43-27: G1CLUTRD: CLUT Memory Read Data Register

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CLUTRD15	CLUTRD14	CLUTRD13	CLUTRD12	CLUTRD11	CLUTRD10	CLUTRD9	CLUTRD8
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CLUTRD7	CLUTRD6	CLUTRD5	CLUTRD4	CLUTRD3	CLUTRD2	CLUTRD1	CLUTRD0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **CLUTRD<15:0>:** Color Look-up Table Memory Read Data bits

This register contains the most recent read from the CLUT memory pointed to by the CLUTADR (G1CLUT<7:0>) bits. Reading of the CLUT memory is triggered when the CLUTTRD (G1CLUT<9>) bit goes from '0' to '1'.

Register 43-28: G1MRGN: Interrupt Advance Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
VBAMGN7	VBAMGN6	VBAMGN5	VBAMGN4	VBAMGN3	VBAMGN2	VBAMGN1	VBAMGN0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
HBAMGN7	HBAMGN6	HBAMGN5	HBAMGN4	HBAMGN3	HBAMGN2	HBAMGN1	HBAMGN0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8 **VBAMGN<7:0>:** Vertical Blanking Advance bits

The number of DISPCLK cycles, in advance, the vertical blanking interrupt will assert ahead of the actual start of the vertical blanking.

bit 7-0 **HBAMGN<7:0>:** Horizontal Blanking Advance bits

The number of DISPCLK cycles, in advance, the horizontal blanking interrupt will assert ahead of the actual start of the horizontal blanking.

Register 43-29: G1CHRX: Character X Coordinate Print Position Register

U-0	U-0	U-0	U-0	U-0	R-0	R-0	R-0
—	—	—	—	—	CURPOSX10	CURPOSX9	CURPOSX8
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CURPOSX7	CURPOSX6	CURPOSX5	CURPOSX4	CURPOSX3	CURPOSX2	CURPOSX1	CURPOSX0
bit 7							bit 0

Legend:			
R = Readable bit	U = Unimplemented bit, read as '0'		
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-11 **Unimplemented:** Read as '0'

bit 10-0 **CURPOSX<10:0>:** Current Character Position in the X Coordinate bits

Register 43-30: G1CHRY: Character Y Coordinate Print Position Register

U-0	U-0	U-0	U-0	U-0	R-0	R-0	R-0
—	—	—	—	—	CURPOSY10	CURPOSY9	CURPOSY8
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CURPOSY7	CURPOSY6	CURPOSY5	CURPOSY4	CURPOSY3	CURPOSY2	CURPOSY1	CURPOSY0
bit 7							bit 0

Legend:			
R = Readable bit	U = Unimplemented bit, read as '0'		
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-11 **Unimplemented:** Read as '0'

bit 10-0 **CURPOSY<10:0>:** Current Character Position in the Y Coordinate bits

Register 43-31: G1IPU: Inflate Processor Status Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	R-0	R-0	R-0	R-0	R-0	R-0
—	—	HUFFERR(2)	BLCKERR(2)	LENERR(2)	WRAPERR(2)	IPUDONE(1,2)	BFINAL(1,2)
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-6 **Unimplemented:** Read as '0'

bit 5 **HUFFERR:** Undefined Huffmann Code Encountered Status bit(2)
- 1 = Undefined code encountered
- 0 = No undefined code encountered

bit 4 **BLCKERR:** Undefined Block Code Encountered Status bit(2)
- 1 = Undefined block encountered
- 0 = No undefined block encountered

bit 3 **LENERR:** Mismatch in Expected Block Length Status bit(2)
- 1 = Mismatch in block length detected
- 0 = No mismatch in block length detected

bit 2 **WRAPERR:** Wrap-Around Error Status bit(2)
- 1 = Wrap-around error encountered
- 0 = No wrap-around error encountered

bit 1 **IPUDONE:** IPU Decompression Status bit(1,2)
- 1 = Decompression done
- 0 = Decompression not yet done

bit 0 **BFINAL:** Final Block Encountered Status bi(1,2)
- 1 = Final block encountered
- 0 = Final block not encountered

Note 1: The IPUDONE and BFINAL status bits are set after successful decompression.

2: All IPU status bits are available after each decompression. All status bits are automatically cleared every time a decompression command (IPU_DECOMPRESS) is issued.

Register 43-32: G1DBEN: Data I/O Pad Enable Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GDBEN15	GDBEN14	GDBEN13	GDBEN12	GDBEN11	GDBEN10	GDBEN9	GDBEN8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GDBEN7	GDBEN6	GDBEN5	GDBEN4	GDBEN3	GDBEN2	GDBEN1	GDBEN0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **GDBEN<15:0>:** Display Data Pads Output Enable bits

1 = Corresponding Display Data (GD<x>) pin is enabled

0 = Corresponding Display Data (GD<x>) pin is disabled

GDBEN<15:0> can be used to disable or enable specific data signals while DPPINOE (G1CON3<9>) is set.

DPPINOE	GDBENx (where x = 0 to 15)	
1	1	Display data signal (GD) associated with GDBENx is enabled.
1	0	Display data signal (GD) associated with GDBENx is disabled.
0	Don't care	Display data signal (GD) associated with GDBENx is enabled.

43.3 OPERATION

This section gives a brief overview of the operation of the Graphics Controller module followed by three implementation examples for the three supported display modes.

43.3.1 Functional Overview

The module is composed of the command interface that receives commands from the processor through the module registers, three Graphical Processing Units (GPUs), the display controller and the memory arbiter.

The display controller will continuously refresh the display unit from a defined display buffer while the GPUs access the memory. The display buffer and GPU's work areas can be configured to point to the same memory address. Only one GPU can be active at any time. There is no need for special software routines to schedule the processing units, since they all share a single, 16-deep command FIFO.

The refresh rate, resolution and color depth of the chosen display are used to determine the parameters of the display controller.

The three GPUs are the following:

- Rectangle Copy Graphics Processing Unit (RCCGPU) – This processing unit draws filled rectangles with specified colors. A subset of a rectangle will be a point, vertical lines and horizontal lines. Rectangle copies or modifications can also be performed with raster operations and operating with transparent colors.
- Character Graphics Processing Unit (CHRGPU) – This processing unit can render strings, with selectable background and foreground colors, with area masking enabled or disabled. Rendering can also be performed with a transparent background.
- Inflate Processing Unit (IPU) – This processing unit decompresses the compressed data (compressed using the loss-less DEFLATE algorithm – RFC 1951 with fixed Huffman codes) stored in the memory.

Note:	DEFLATE – "Compressed Data Format Specification" version 1.3; May 1998, RFC 1951, IETF.

The display controller supports interfaces for TFT displays with a parallel RGB bus, Monochrome STN displays and color STN displays. For STN displays, a 16-level dithering unit is used. CLUT is available as an option for both TFT and STN type displays.

43.3.2 Pixel Addressing

Depending on the color depth setting, each pixel in the display buffer is addressed differently. Figure 43-2 and Figure 43-3 illustrate how each pixel can be addressed in the different color depth settings.

Figure 43-2: Pixel Addressing for 1 bpp and 2 bpp

Color Depth = 1 bpp, Each Memory Word Address is Accessing 16 Pixels

Memory Word Address	MSb															LSb
DPADR + 0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPADR + 1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DPADR + 2	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
DPADR + 3	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
DPADR + n	m + 15	m + 14	m + 13	m + 12	m + 11	m + 10	m + 9	m + 8	m + 7	m + 6	m + 5	m + 4	m + 3	m + 2	m + 1	m + 0

Where m = n * 16

Color Depth = 2 bpp, Each Memory Word Address is Accessing 8 Pixels

Memory Word Address	MSb							LSb
DPADR + 0	7	6	5	4	3	2	1	0
DPADR + 1	15	14	13	12	11	10	9	8
DPADR + 2	23	22	21	20	19	18	17	16
DPADR + 3	31	30	29	28	27	26	25	24
...	...	...	...	...	...	...	...	...
DPADR + n	m + 7	m + 6	m + 5	m + 4	m + 3	m + 2	m + 1	m + 0

Where m = n * 8

Figure 43-3: Pixel Addressing for 4 bpp, 8 bpp and 16 bpp

Color Depth = 4 bpp, Each Memory Word Address is Accessing 4 Pixels

Memory Word Address	MSb			LSb
DPADR + 0	3	2	1	0
DPADR + 1	7	6	5	4
...	...	...	...	...
DPADR + n	m + 3	m + 2	m + 1	m

Where m = n * 4

Color Depth = 8 bpp, Each Memory Word Address is Accessing 2 Pixels

Memory Word Address	MSb	LSb
DPADR + 0	1	0
DPADR + 1	3	2
...	...	...
DPADR + n	m + 2	m

Where m = n * 2

Color Depth = 16 bpp, Each Memory Word Address is Accessing 1 Pixel

Memory Word Address	MSb LSb
DPADR + 0	0
DPADR + 1	1
...	...
DPADR + n	n

The GPUs will process commands and write to the display buffer using the color depth setting in the PUBPP<2:0> (G1CON1<7:5>). The display controller will read the display buffer using the color depth setting in the DPBPP<2:0> (G1CON2<7:5>). It is important that the color depth used by the GPU (PUBPP) must be equal to the color depth used in displaying the data (DPBPP). In some applications, software rendered images can be processed in a separate memory with different color depth setting. Displaying the software processed images will need programming DPBPP to match the color depth used and switching the display buffer location, DPADR, to the location of the processed images, or copying the processed images to the current DPADR location.

43.3.3 Image Coordinates to Pixel Addressing

There are two ways to copy an image to the display buffer: using the GPUs or manually copying it by software. In either case, the translation of the image's pixel coordinates to the pixel address in the display buffer is calculated based on the color depth setting and the "Memory Pitch" of the display buffer. Memory pitch is defined as the number of byte addresses (offset) added to the address of a given pixel to reach the next pixel of the next line belonging to the same column on the display buffer. Equation 43-1 shows how to calculate for the memory pitch of a given display buffer.

Equation 43-1:

$$\text{Memory Pitch} = (\text{Display Width} \times \text{color depth})/8$$

Note: The Graphics Controller module supports display widths that are a multiple of 16 pixels. The minimum display width supported is 32 pixels. For MSTN displays with a 16-bit interface, the minimum supported width is 48 pixels.

Figure 43-4: Image Pixel Addressing Cases

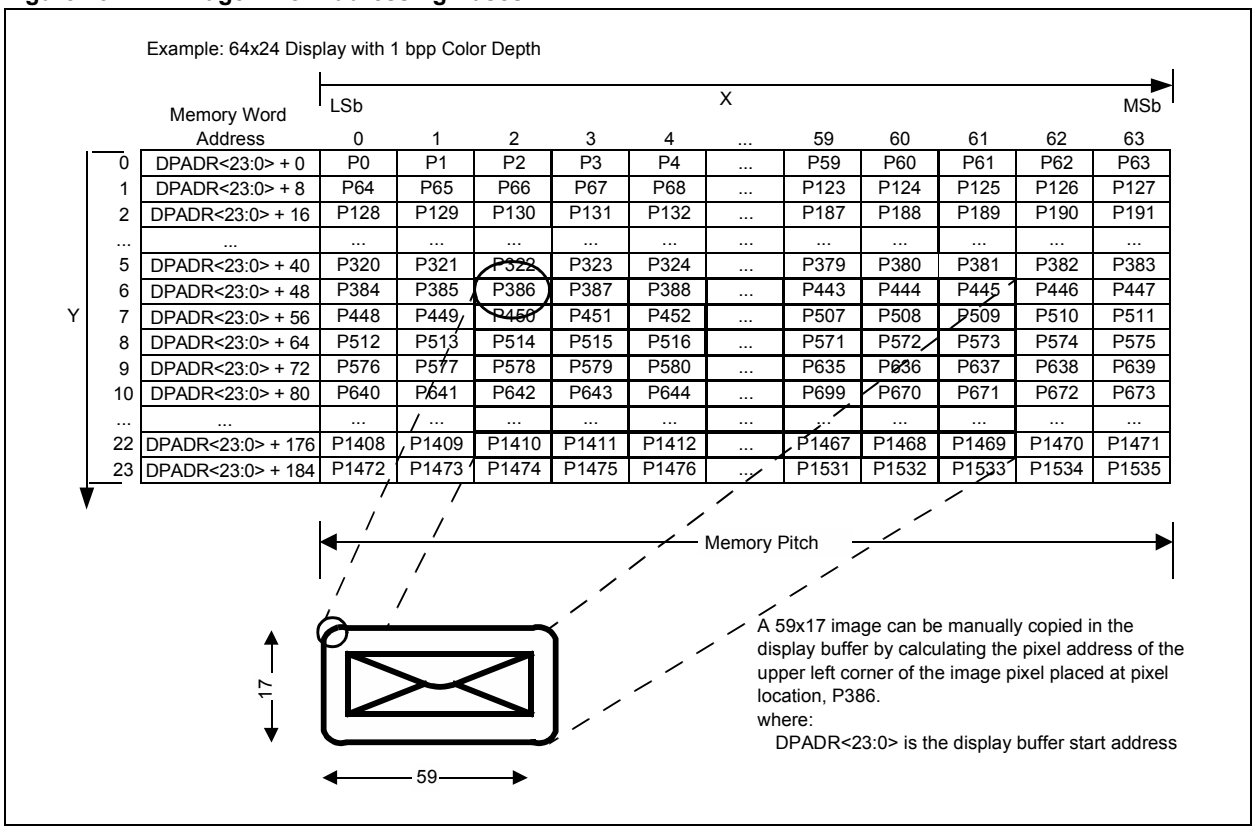


Figure 43-4 shows an example where a 64x24 display, with the display buffer starting at DPADR<23:0>, is initialized in memory. Memory pitch is calculated as: (64*1)/8 = 8. If a 59x17 image is to be copied into the display buffer at location (2,6) or P386, the pixel address offset of P386 from the display buffer start address is calculated using Equation 43-2.

Equation 43-2:

Pixel Address Offset = (Memory Pitch x Y position) + (X position x color depth/8)

Substituting the values, (8*6) + (2/8), yields an offset of 48, ignoring the fraction component of the result. The fraction component determines the bit position of the pixel at location (2,6) at 1 bpp. Therefore, the address of the left top pixel of the copied image in the display buffer is DPADR<23:0> + 48.

A general way of calculating the pixel address for an m x n display, with m divisible by 16, is shown in Figure 43-5. For example, a display with m = 320 and n = 240 is used. To locate the relative pixel address of the upper left corner pixel of an image placed at X = 3 and Y = 3, simply use the equation provided in the intersection of X = 3 and Y = 3. Calculating with m = 320, we get (320 * 3) + 3 = 963. The pixel address offset is then calculated from the result shown in Equation 43-3.

Equation 43-3:

Pixel Address Offset = Relative Pixel Address x color depth/8

For a 4 bpp setting, the pixel is located in address: DPADR + (963*4/8) or DPADR + 481. Again, the fractional component that determines the pixel location in the addressed byte is ignored.

Figure 43-5: X, Y Coordinates to Pixel Address

Image Coordinates Relative to Pixel Addresses

X

Y	0	1	2	3	...	m – 3	m – 2	m – 1
0	0	1	2	3	...	m – 3	m – 2	m – 1
1	m	m + 1	m + 2	m + 3	...	m + (m – 3)	m + (m – 2)	m + (m – 1)
2	m*2	(m * 2) + 1	(m * 2) + 2	(m * 2) + 3	...	(m * 2) + (m – 3)	(m * 2) + (m – 2)	(m * 2) + (m – 1)
3	m*3	(m * 3) + 1	(m * 3) + 2	(m * 3) + 3	...	(m * 3) + (m – 3)	(m * 3) + (m – 2)	(m * 3) + (m – 1)
...	...	...	...	...	...	...	...	...
n – 3	m*(n – 3)	(m * (n – 3)) + 1	(m * (n – 3)) + 2	(m * (n – 3)) + 3	...	(m * (n – 3)) + (m – 3)	(m * (n – 3)) + (m – 2)	(m * (n – 3)) + (m – 1)
n – 2	m*(n – 2)	(m * (n – 2)) + 1	(m * (n – 2)) + 2	(m * (n – 2)) + 3	...	(m * (n – 2)) + (m – 3)	(m * (n – 2)) + (m – 2)	(m * (n – 2)) + (m – 1)
n – 1	m*(n – 1)	(m * (n – 1)) + 1	(m * (n – 1)) + 2	(m * (n – 1)) + 3	...	(m * (n – 1)) + (m – 3)	(m * (n – 1)) + (m – 2)	(m * (n – 1)) + (m – 1)

Where:
m = Display Width in Pixels
n = Display Height in Pixels

43.3.4 Color Look-up Table (CLUT)

The display controller has the option to use the built-in 256 entry Color Look-up Table (CLUT) to represent pixels from the display buffer in memory. If the CLUT is enabled, each pixel in the display buffer is assumed to contain the color index. This color index is used as the address of the CLUT entry that contains the color value that will be used for the given pixel. The number of valid entries in the CLUT is dependent on the Display bits-per-pixel (DPBPP) setting. Table 43-1 provides the CLUT valid entries with respect to the DPBPP setting.

Table 43-1: Display bpp Settings vs. CLUT Entries

DPBPP	Valid CLUT Entries (n)
1	2
2	4
4	16
8	256
16	256

All valid entries always start at Index 0 and the last entry at n – 1. Therefore, if the DPBPP setting is set to 2 bpp, there will be 4 valid entries in the CLUT. These four entries are placed in indices, 0-3.

Note: Using a CLUT with 16 bpp pixel data in memory will result in the 8 Least Significant bits (LSbs) of the pixel data as the index of the color from the CLUT. The upper bits of the pixel data are ignored. This will result in a larger memory used for the display buffer, but with no improvement in the image quality. Therefore, it is recommended to disable the CLUT when the display buffer is set to contain 16 bpp data. CLUT uses a 256 entry built-in table. This table is not mapped on the system memory; access is only through the SFR registers controlling the CLUT.

For the TFT and CSTN type of displays, each entry of the CLUT is always 16-bit colors, regardless of the DPBPP setting, or the number of bits in the display interface. For the MSTN type of display, each entry is always 4 bits wide or 16 color shades.

Table 43-2: RGB and Luminance Output from CLUT for Color and Monochrome Displays

Output Type	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGB	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
Luma	Unused												M3	M2	M1	M0

The CLUT output is then mapped to the display pins depending on the display type set in the DPMODE (G1CON2<2:0>) bits and the display data bus width set in the DPGWDTH (G1CON2<15:14>). Mapping of the color data is shown in **Section 43.3.6 “RGB Mapping”**.

The CLUT entries are not initialized upon a module Reset and are not ensured to retain previous programmed values after a module Reset. Application must program the contents of the CLUT prior to first use.

43.3.4.1 READING AND WRITING CLUT ENTRIES

Reading and writing entries to the CLUT by software requires that the CLUT must be configured for CPU access. The steps prepare the CLUT to shift from display controller access to CPU access. When CLUT is set for CPU access, the display controller cannot access the CLUT entries, and for this reason, CLUT manipulation should be done in synchronization periods to avoid artifacts on the display. For TFT mode, CLUT manipulation is best performed on vertical and horizontal blanking periods. During these periods, the screen is not being refreshed. However, for STN mode, the screen is refreshed constantly. To avoid artifacts, the display screen can be turned off when manipulating the contents of the CLUT. The following steps must be observed when reading or writing CLUT entries:

Writing entries to the CLUT:

1. Set the CLUTADR (G1CLUT<7:0>) bits.
2. Set the CLUTRWEN (G1CLUT<8>) bits.
3. Write data to the G1CLUTWR register.
4. Read the CLUTBUSY (G1CLUT<14>) status bit and check. If it is '0', data has been written; if it is '1', keep reading the status until it is cleared.
5. To write the next data, go back to Step 1.
6. After writing all the data, clear CLUTRWEN.

Reading entries from the CLUT:

1. Set the CLUTADR bits.
2. Set the CLUTRWEN bit.
3. Set the CLUTTRD bit to trigger the data fetch from CLUT.
4. Clear the CLUTTRD bit.
5. Read the CLUTBUSY status bit and check. If it is '0', data is ready, if it is '1', keep reading the status until it is cleared.
6. Read the data from the G1CLUTRD register.
7. Go back to Step 1 if more data is to be read from CLUT.
8. After reading all the data, clear CLUTRWEN.

Clearing CLUTRWEN after reading or writing entries to the CLUT is important for the display controller to be able to access the CLUT entries.

43.3.5 Dithering

Dithering is always enabled in MSTN and CSTN modes; it is not used in TFT mode.

43.3.6 RGB Mapping

For the TFT mode with CLUT bypassed, the RGB mapping of data in memory to the 16-bit RGB data bus interface is provided in Table 43-3.

Table 43-3: TFT Mode Memory Pixel Value Mapping to RGB Data Bus

Color	Memory Value	Red<4:0> GD<15:11>	Green<5:0> GD<10:5>	Blue<4:0> GD<4:0>
1 bpp				
Black	0	00000	000000	00000
White	1	11111	111111	11111
2 bpp				
Black	00	00000	000000	00000
Dark Gray	01	01000	010000	01000
Light Gray	10	11000	110000	11000
White	11	11111	111111	11111
4 bpp				
Black	0000	00000	000000	00000
Dark Blue	0001	00000	000000	10000
Dark Red	0010	10000	000000	00000
Dark Magenta	0011	10000	000000	10000
Dark Green	0100	00000	100000	00000
Dark Cyan	0101	00000	100000	10000
Dark Yellow	0110	10000	100000	00000
Gray	0111	10000	100000	10000
Black	1000	00000	000000	00000
Blue	1001	00000	000000	11111
Red	1010	11111	000000	00000
Magenta	1011	11111	000000	11111
Green	1100	00000	111111	00000
Cyan	1101	00000	111111	11111
Yellow	1110	11111	111111	00000
White	1111	11111	111111	11111
8 bpp				
	D<7:0>	D<7,6,5,5,5>	D<4,3,2,2,2,2>	D<1,0,0,0,0>
16 bpp				
	D<15:0>	D<15:11>	D<10:5>	D<4:0>

If the CLUT is enabled, the mapping will still be the same except that the color generated will now be dependent on the color data assigned to the particular CLUT index. For example, in 1 bpp mode, Index 0 is programmed to contain the value for white and Index 1 is programmed to contain the value for black. The image displayed on the screen will swap its colors; black will turn to white and vice versa.

For CSTN and MSTN modes, the mapping is dependent on the STN display's data bus width and the timing sequence of the output data. Refer to **Section 43.4.2 "STN Display Interface"** for details.

43.4 DISPLAY CONTROLLER

The display controller continuously reads data from the display buffer and outputs it to the display with the display clock, vertical and horizontal synchronization signals, and enable signal configured to the specifications of the display. Timing of the synchronization signals, polarity of the signals, and required frame rate of the display are determined from the display specifications and translated to values to be programmed into the registers of the display controller. There are differences in the way the signals are generated for the TFT type and STN type displays (DPMODE settings). The following sections describe these differences and give examples on how the registers are configured for each type of display.

43.4.1 TFT Display Interface

For a TFT display, Figure 43-6 illustrates typical timing parameters to generate an active frame.

Figure 43-6: TFT Display Active Frame Timing

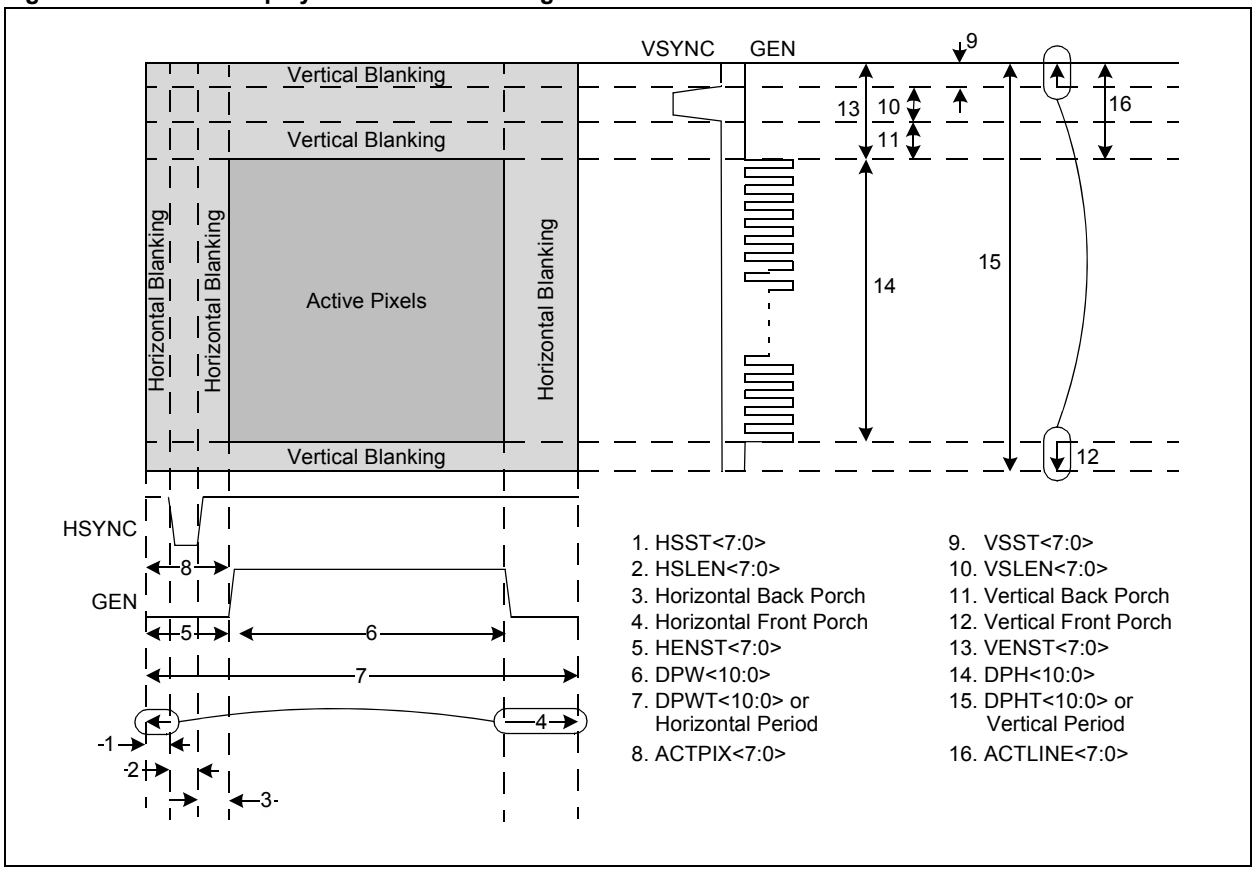
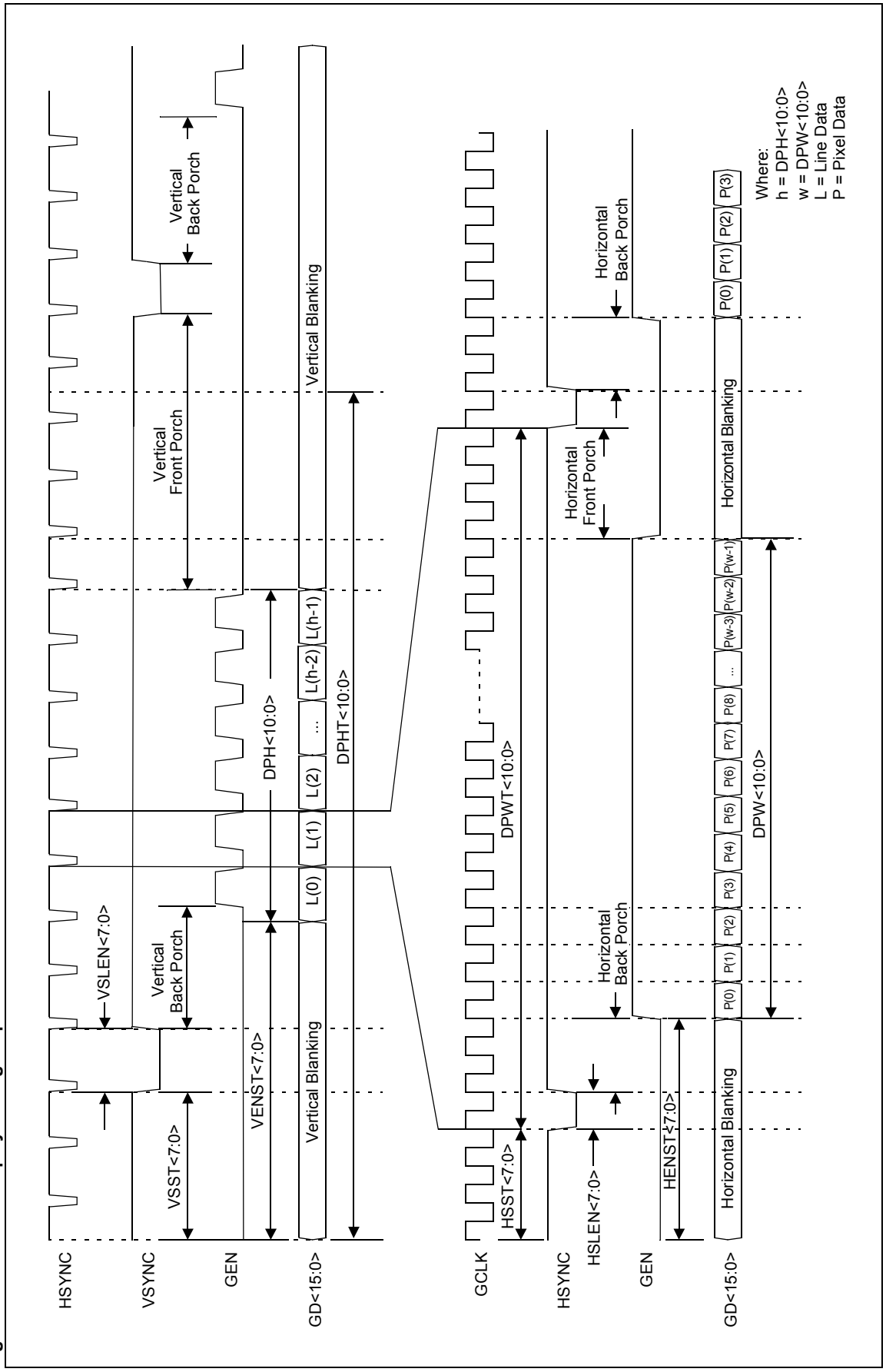


Figure 43-7 illustrates the detailed timing of the signals.

Figure 43-7: TFT Display Timing Sequence



The frame rate can be derived from the total width and height of the display. The total width and height are also known as horizontal and vertical periods.

Equation 43-4:

Frame Rate = GCLK Frequency/(DPWT x DPHT)

The horizontal and vertical periods are computed based on the specifications of the display.

Equation 43-5:

DPWT = DPW + HSLEN + Horizontal Back Porch + Horizontal Front Porch

Equation 43-6:

DPHT = DPH + VSLEN + Vertical Back Porch + Vertical Front Porch

Table 43-4 shows the relationship of the display signals to the different parameters of the display controller.

Table 43-4: Display Signal Timing Control Summary

Display Signals	Timing Controlled by Parameters
HSYNC	HSST, HSLEN
VSYNC	VSST, VSLEN
GEN	HENST, VENST, DPW
GD	ACTPIX, ACTLINE

Both the vertical and horizontal programmable parameters are based on vertical and horizontal counters that loop from 0 to DPHT – 1 and 0 to DPWT – 1, respectively. Therefore, if the first line is the first active line and the first pixel is the first active pixel, the ACTPIX and ACTLINE parameters should both be set to '0'.

For TFT displays, ACTPIX and ACTLINE values depend on the display's specifications on synchronized pulse widths, front and back porches. Generally, the values for HENST and VENST are the same for ACTPIX and ACTLINE. The two sets are provided for flexibility.

In cases, where the vertical and the horizontal front and back porches are given in the specifications, the timing parameters for the HSYNC and VSYNC signals are determined by calculation.

Equation 43-7:

Horizontal Blanking = Horizontal Front Porch + HSLEN + Horizontal Back Porch

Equation 43-8:

Vertical Blanking = Vertical Front Porch + VSLEN + Vertical Back Porch

Since HSYNC and VSYNC are signals that the display depends on to time sampling of valid data, the overall timing of HSYNC and VSYNC to GEN, and valid data must meet the requirement of the display specifications.

For the HSYNC signal, the HSLEN and Horizontal Back Porch are given. Therefore, HSST should not exceed the Horizontal Front Porch.

Equation 43-9:

HSST ≤ Horizontal Front Porch

Equation 43-10:

HENST = HSST + HSLEN + Horizontal Back Porch

Analogously, for the VSYNC signal, the VSLEN and Vertical Back Porch are given. Therefore, VSST should not exceed the Vertical Front Porch.

Equation 43-11:

VSST ≤ Vertical Front Porch

Equation 43-12:

VENST = VSST + VSLEN + Vertical Back Porch

ACTPIX and ACTLINE can be set equal to HENST and VENST, respectively. In cases where the Horizontal and Vertical Blanking periods are given, the equations are rearranged to derive the timing required for the parameters. In cases where the active pixel and line timing need to be adjusted, modify the programmed values to ACTPIX and ACTLINE.

Table 43-5 shows a sample of the configuration of a QVGA TFT display. This particular display has the following typical parameters taken from its specifications document:

- Display Clock Period – 183 ns
- Horizontal Period – 280 Clocks
- Horizontal Front Porch – 10 Clocks
- Horizontal Back Porch – 20 Clocks
- Horizontal Synchronization Pulse – 10 Clocks
- Vertical Period – 326 Lines
- Vertical Front Porch – 2 Lines
- Vertical Back Porch – 1 Line
- Vertical Synchronization Pulse – 3 Lines

Table 43-5: QVGA TFT Display Sample Configuration

Parameter	Register	Register Bit(s)	Value	Description
Display Type	G1CON2	DPMODE	001	Use TFT type display.
Display Data Bus Enable	G1DBEN	GDBEN	0xFFFF	Display uses all 16-bit data lines so all data bus pins are enabled.
Display Data Bus Width	G1CON2	DPGWDTH	x	Ignored for TFT mode. Display data bus width in this mode is always assumed to be 16 bits wide.
Display Width	G1DPW	DPW	240	Active frame width.
Display Height	G1DPH	DPH	320	Active frame height.
Display Width Total	G1DPWT	DPWT	280	Based on the display's specifications.
Display Height Total	G1DPHT	DPHT	326	Based on the display's specifications.
Display Clock Sampling Edge	G1CON3	DPCLKPOL	0	Display samples data on the falling edge.
Use Data Enable Signal	G1CON3	DPENOE	1	Display uses a data enable signal (GEN). If the display does not use an enable signal, set DPENOE to '0' to free up the pin for other functions.
Data Enable Signal Active Level	G1CON3	DPENPOL	1	Signal is active-high.
Use VSYNC Signal	G1CON3	DPVSOE	1	Enable the use of the VSYNC signal.
Use HSYNC Signal	G1CON3	DPHSOE	1	Enable the use of the HSYNC signal.
VSYNC Signal Active Level	G1CON3	DPVSPOL	0	Signal is active-low.
HSYNC Signal Active Level	G1CON3	DPHSPOL	0	Signal is active-low.
Active Line	G1ACTDA	ACTLINE	6	Derived by the equation: Vertical Front Porch + Vertical Sync Pulse Width (VSLEN) + Vertical Back Porch (2 + 3 + 1).
Active Pixel	G1ACTDA	ACTPIX	40	Derived by the equation: Horizontal Front Porch + Horizontal Sync Pulse Width (VSLEN) + Horizontal Back Porch (10 + 10 + 20).
VSYNC Start	G1VSYNC	VSST	2	Taken from Vertical Front Porch value.
HSYNC Start	G1HSYNC	HSST	10	Taken from Horizontal Front Porch value.
VSYNC Length	G1VSYNC	VSLEN	3	Based on the display's specifications.

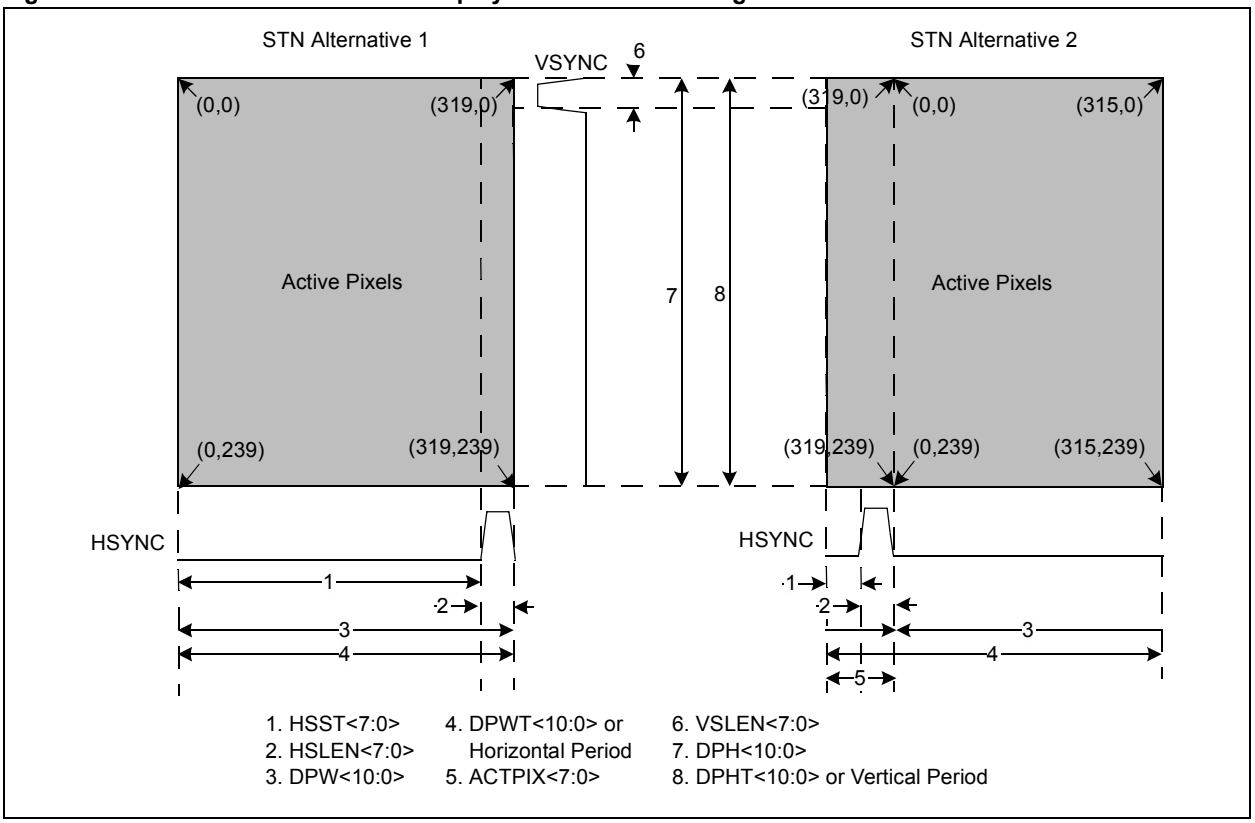
Table 43-5: QVGA TFT Display Sample Configuration (Continued)

Parameter	Register	Register Bit(s)	Value	Description
HSYNC Length	G1HSYNC	HSLEN	10	Based on the display's specifications.
Vertical Enable Start	G1DBLCON	VENST	6	Typically, this value is the same as ACTLINE.
Horizontal Enable Start	G1DBLCON	HENST	40	Typically, this value is the same as ACTPIX.
Enable Display Controller Pins	G1CON3	DPPINOE	1	Enable pin output pads.
Enable Display Power Signal	G1CON3	DPPOWER	1	Set display power signal to '1'.
Use Display Power Signal Pin	G1CON3	DPPWROE	1	Use the GPWR pin as the display power signal. Use of this pin will depend on the display's power requirements.
Enable Display Controller Module	G1CON1	G1EN	1	Turn on the display controller module.

43.4.2 STN Display Interface

Figure 43-8 illustrates the typical timing parameters to generate one active frame on a 320x240 screen for color and mono STN displays.

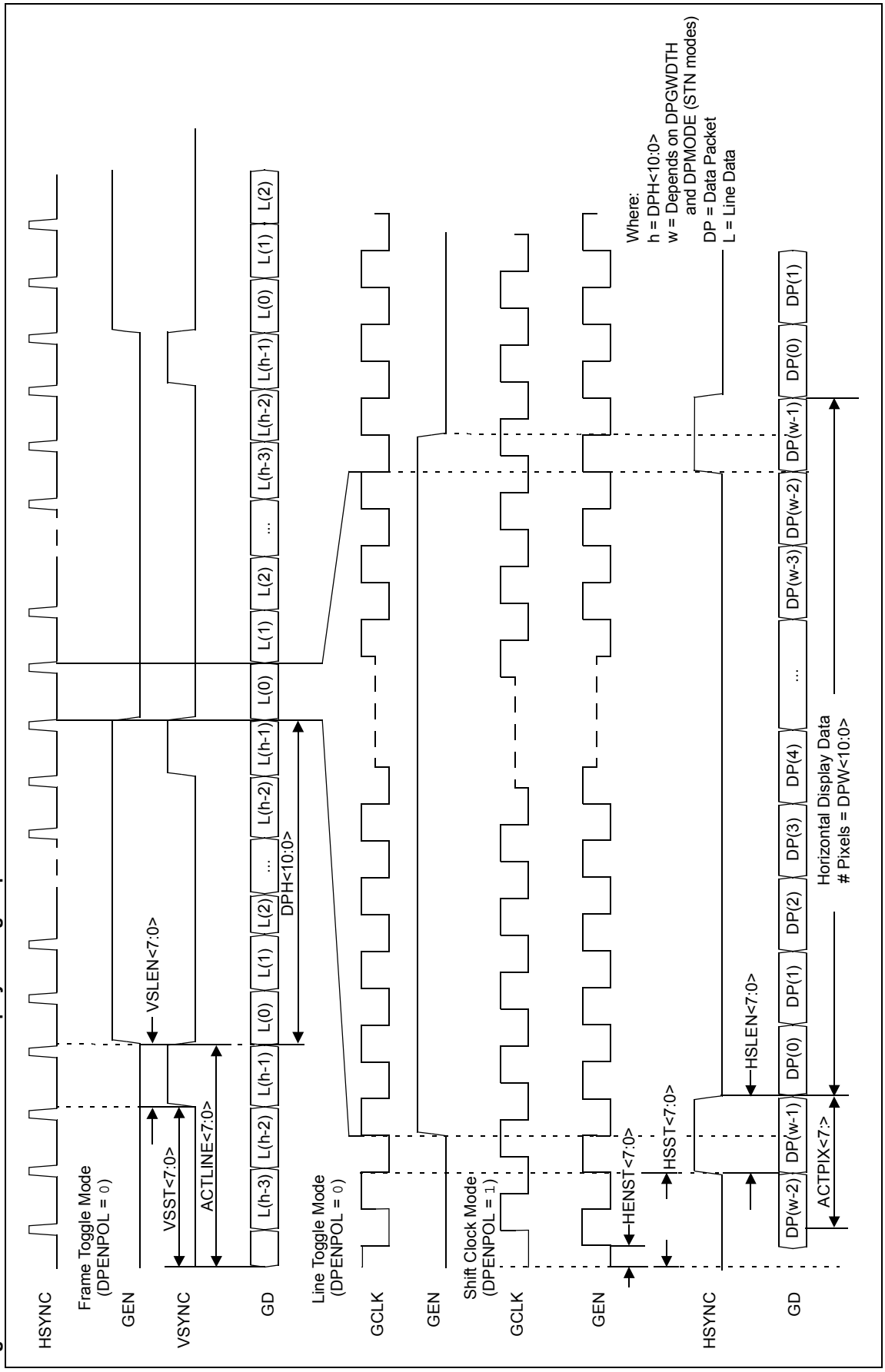
Figure 43-8: Color and Mono STN Display Active Frame Timing



In Figure 43-8, the frame on the left shows a case when the asserted high HSYNC pulse is required at the last pixel of each line. Since the HSST parameter has only 8 bits, this will be a problem for larger displays with a horizontal pixel count greater than 256. An alternative is to generate the HSYNC signal from a low value (refer to ACTPIX timing of STN Alternate 2 in Figure 43-8). HSST will trigger the HSYNC pulse with ACTPIX set to start outputting valid pixel data right after the HSYNC pulse.

Figure 43-9 displays the detailed timing of the signals.

Figure 43-9: Color and Mono STN Display Timing Sequence



43.4.2.1 STN INTERFACE MODES

Interfacing to an STN device will use one of the three modes.

43.4.2.1.1 Shift Clock Mode

In Shift Clock mode, the displays will require a latch clock input and a shift clock input. The shift clock input will be running at the same frequency as the latch clock input, but shifted 180° out of phase. Additionally, the shift clock can be delayed by a number of DISPCLK cycles programmed in the HENST register bits.

To enable this mode, program the following register settings:

- DPENPOL = 1
- VENST = Don't care
- HENST = # of DISPCLK to delay the shift clock

Signals to the display will be connected to the following outputs:

- GEN = Shift clock input
- GCLK = Latch clock input

43.4.2.1.2 Frame Toggle Mode

In Frame Toggle mode, the displays will require a MOD input that toggles at frame rate.

To enable this mode, program the following register settings:

- DPENPOL = 0
- VENST = 0
- HENST = Don't care

Signals to the display will be connected to the following outputs:

- GEN = MOD
- GCLK = Shift clock

43.4.2.1.3 Line Toggle Mode

In Line Toggle mode, the displays will require a MOD input that toggles at line rate. The toggle rate is equal to the value programmed in the VENST parameter. If VENST is set to three, the GEN signal will toggle every 3 lines. Additionally, the MOD signal can be delayed by a number of DISPCLK cycles programmed in the HENST register bits.

To enable this mode, program the following register settings:

- DPENPOL = 0
- VENST = Toggle rate
- HENST = # of DISPCLK to delay MOD signal

Signals to the display will be connected to the following outputs:

- GEN = MOD
- GCLK = Shift clock

43.4.2.2 STN DATA MAPPING

Figure 43-10 and Figure 43-11 illustrate the mapping of the color and monochrome pixel data to a CSTN and MSTN display, respectively.

Figure 43-10: MSTN Data Mapping

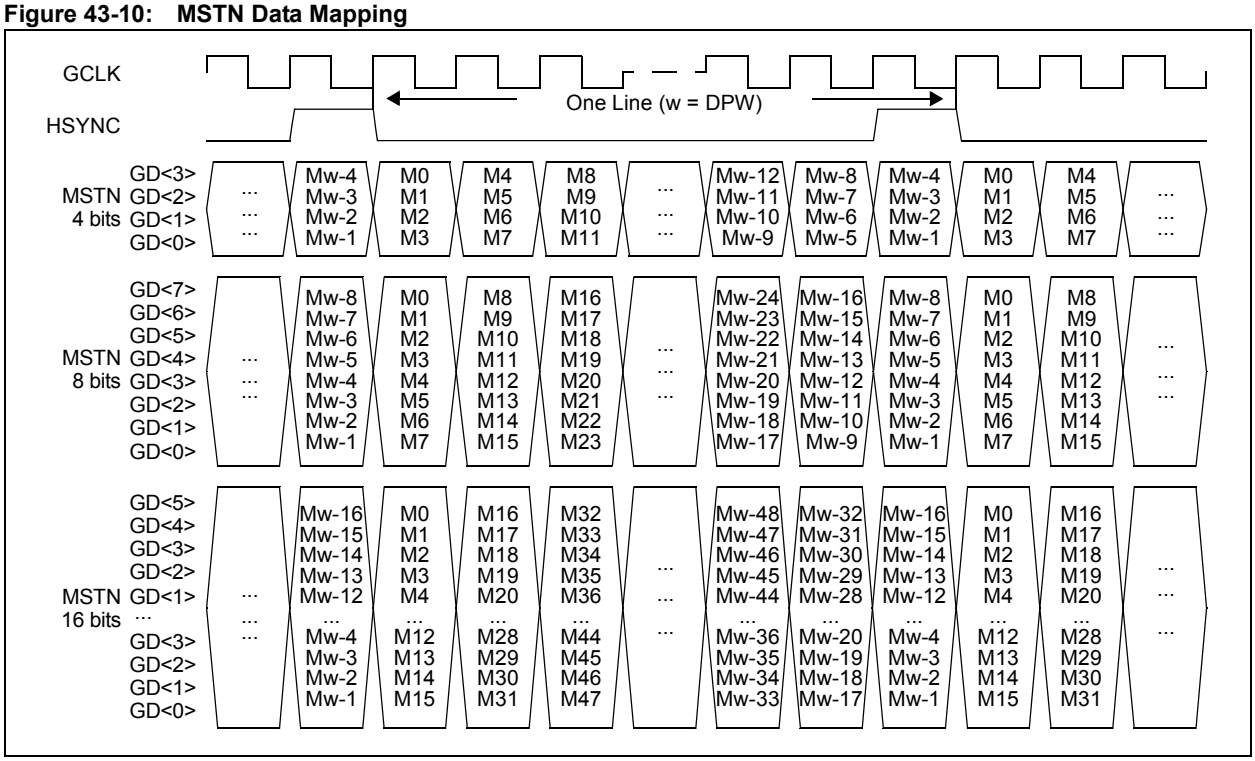
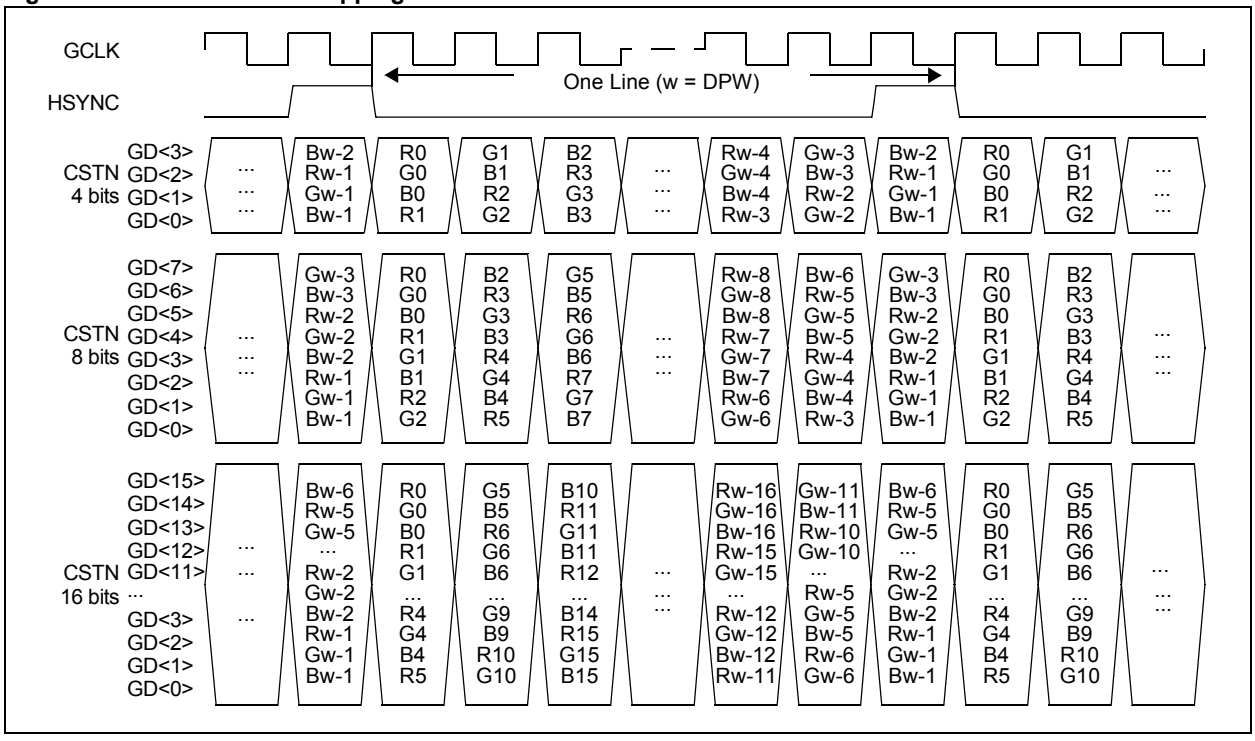


Figure 43-11: CSTN Data Mapping



Based on Equation 43-4, the frame rate can be derived from the total width and height of the display. However, for MSTN and CSTN displays, the calculation of the frame rate is affected by the display interface width (DPGWDTH). For MSTN, the frame rate is calculated by:

Equation 43-13:

$$\text{Frame Rate} = (\text{GCLK Frequency} \times \text{DPGWDTH})/(\text{DPWR} \times \text{DPHT})$$

For CSTN, the frame rate is calculated by:

Equation 43-14:

$$\text{Frame Rate} = (\text{GCLK Frequency} \times \text{DPGWDTH})/(3 \times \text{DPW} \times \text{DPH})$$

In most cases, the GCLK frequency of STN displays is taken from the display specifications along with the display interface width, active pixel width and height, and the total width and height. To conform to the display's frame rate, the frequency of the display clock input (DISPCLK) to the display controller needs to be determined.

Equation 43-15:

$$\text{DISPCLK Frequency} = (\text{GCLK Frequency} \times \text{DPGWDTH})/2$$

The GCLK frequency is automatically determined by the display controller based on the display interface width and the DISPCLK frequency. DISPCLK is an input to this module and its frequency is set externally. Refer to the device family data sheet for more information on this input.

Table 43-6 provides an example of an MSTN display with the following typical parameters taken from its specifications document:

- Display GCLK Period – 71 ns (minimum)
- Horizontal Period – 80 GCLK Cycles
- Horizontal Synchronization Pulse – 1 GCLK cycle
- Vertical Period – 240 Lines
- Vertical Synchronization Pulse – 1 Line

Table 43-6: QVGA MSTN Display Sample Configuration

Parameter	Register	Register Bit(s)	Value	Description
Display Type	G1CON2	DPMODE	010	Use MSTN type of display.
Display Data Bus Enable	G1DBEN	GDBEN	0x000F	Display uses 4 data lines. Enable data bus pins, GD<3:0>.
Display Data Bus Width	G1CON2	DPGWDTH	01	4 bits wide.
Display Width	G1DPW	DPW	320	Active frame width.
Display Height	G1DPH	DPH	240	Active frame height.
Display Width Total	G1DPWT	DPWT	320	Typically, MSTN displays have no horizontal blanking.
Display Height Total	G1DPHT	DPHT	240	Typically, MSTN displays have no vertical blanking.
Display Clock Sampling Edge	G1CON3	DPCLKPOL	0	Display samples data on the falling edge.
Use Data Enable Signal	G1CON3	DPENOE	0	Display will use GEN as the MOD or shift clock signal.
Data Enable Signal Mode	G1CON3	DPENPOL	1	GEN signal is used as the shift clock signal.
Use VSYNC Signal	G1CON3	DPVSOE	1	Enable the use of the VSYNC signal.
Use HSYNC Signal	G1CON3	DPHSOE	1	Enable the use of the HSYNC signal.
VSYNC Signal Active Level	G1CON3	DPVSPOL	1	Signal is active-high.

Table 43-6: QVGA MSTN Display Sample Configuration (Continued)

Parameter	Register	Register Bit(s)	Value	Description
HSYNC Signal Active Level	G1CON3	DPHSPOL	1	Signal is active-high.
Active Line	G1ACTDA	ACTLINE	0	—
Active Pixel	G1ACTDA	ACTPIX	4	Number of DISPCLK cycles in a GCLK cycle. This is equal to the value (either 4, 8 or 16) selected by DPGWDTH.
VSYNC Start	G1VSYNC	VSST	0	—
HSYNC Start	G1HSYNC	HSST	1	HSYNC to GCLK setup time.
VSYNC Length	G1VSYNC	VSLEN	1	Based on the display's specifications.
HSYNC Length	G1HSYNC	HSLEN	4	Typically this is equal to ACTPIX. See ACTPIX description for details.
MOD Signal Toggle Rate Type	G1DBLCON	VENST<7:0>	x	Don't care since display is using GEN as shift clock signal.
Shift Clock or MOD Signal Delay	G1DBLCON	HENST<7:0>	0	Shift clock is not delayed.
Enable Display Controller Pins	G1CON3	DPPINOE	1	Enable pin output pads.
Use Display Power Signal Pin	G1CON3	DPPWROE	1	Use the GPWR pin as the display power signal. Use of this pin will depend on the display's power requirements.
Enable Display Power Signal	G1CON3	DPPOWER	1	Display power signal is active-high from the device specifications. Set DPPOWER to '1' to set the GPWR pin to high when turning on the display.
Enable Display Controller Module	G1CON1	G1EN	1	Turn on the display controller module.

Table 43-7 provides an example of a CSTN display with the following typical parameters taken from its specifications document.

- Display GCLK Period – 25 ns (minimum)
- Horizontal Period – 120 GCLK Cycles
- Horizontal Synchronization Pulse – 25 ns (minimum)
- Vertical Period – 240 Lines
- Vertical Synchronization Pulse – 1 Line

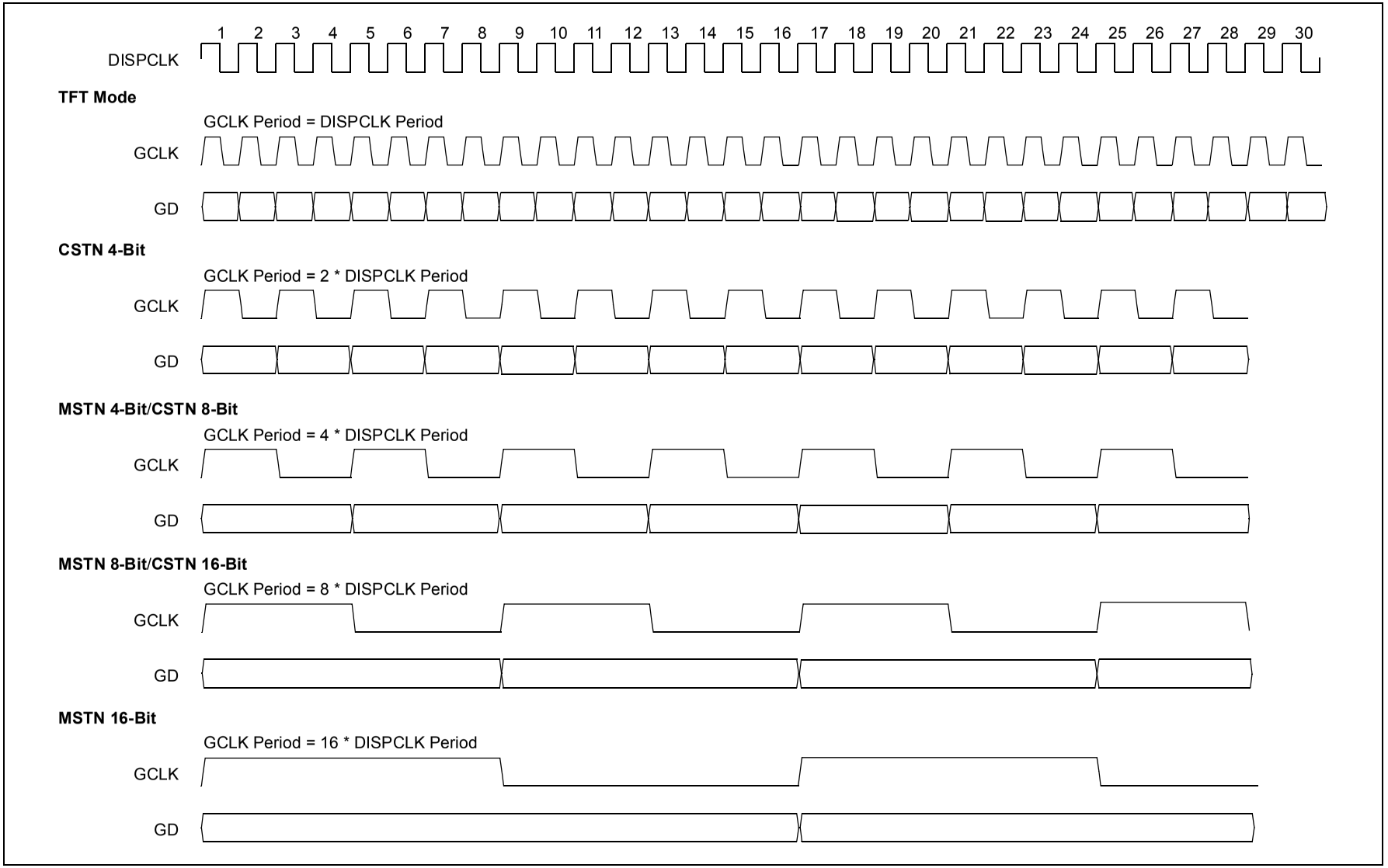
Table 43-7: QVGA CSTN Display Sample Configuration

Parameter	Register	Register Bit(s)	Value	Description
Display Type	G1CON2	DPMODE	011	Use CSTN type of display.
Display Data Bus Enable	G1DBEN	GDBEN	0x00FF	Display uses 8 data lines. Enable data bus pins, GD<7:0>.
Display Data Bus Width	G1CON2	DPGWDTH	10	8 bits wide.
Display Width	G1DPW	DPW	320	Active frame width.
Display Height	G1DPH	DPH	240	Active frame height.
Display Width Total	G1DPWT	DPWT	480	Value is computed by display width: (DPW) * (3/2). In this case, 320 * 3/2 = 480.
Display Height Total	G1DPHT	DPHT	240	Typically, CSTN display has no vertical blanking.
Display Clock Sampling Edge	G1CON3	DPCLKPOL	0	Display samples data on the falling edge.
Use Data Enable Signal	G1CON3	DPENOE	0	Display will use GEN as the MOD or shift clock signal.
Data Enable Signal Mode	G1CON3	DPENPOL	1	GEN signal is used as the shift clock signal.
Use VSYNC Signal	G1CON3	DPVSOE	1	Enable the use of the VSYNC signal.
Use HSYNC Signal	G1CON3	DPHSOE	1	Enable the use of the HSYNC signal.
VSYNC Signal Active Level	G1CON3	DPVSPOL	1	Signal is active-high.
HSYNC Signal Active Level	G1CON3	DPHSPOL	1	Signal is active-high.
Active Line	G1ACTDA	ACTLINE	0	—
Active Pixel	G1ACTDA	ACTPIX	4	For CSTN displays, value is computed by: DPGWDTH (4, 8 or 16)/2. Therefore, legal values are 2, 4 and 8. In this case, DPGWDTH = 8; thus, the programmed value is 4.
VSYNC Start	G1VSYNC	VSST	0	—
HSYNC Start	G1HSYNC	HSST	1	HSYNC to GCLK setup time.
VSYNC Length	G1VSYNC	VSLEN	1	Based on the display's specifications.
HSYNC Length	G1HSYNC	HSLEN	4	Typically, this is equal to ACTPIX. See the ACTPIX description for details.
MOD Signal Toggle Rate Type	G1DBLCON	VENST	x	Don't care, since the display is using GEN as the shift clock signal.
Shift Clock or MOD Signal Delay	G1DBLCON	HENST	0	Shift clock is not delayed.
Enable Display Controller Pins	G1CON3	DPPINOE	1	Enable pin output pads.
Use Display Power Signal Pin	G1CON3	DPPWROE	1	Use the GPWR pin as the display power signal. Use of this pin will depend on the display's power requirements.
Enable Display Power Signal	G1CON3	DPPOWER	1	Display power signal is active-high from the device specifications. Set DPPOWER to '1' to set the GPWR pin to high when turning on the display.
Enable Display Controller Module	G1CON1	G1EN	1	Turn on the display controller module.

43.4.3 Pixel Clock Rate

Figure 43-12 displays the pixel clock, GCLK, generated from the input display clock, which is different for each display mode.

Figure 43-12: Pixel Clock Generation



43.4.4 Display Data Stagger

To reduce the occurrence of ground noise caused by simultaneous switching of data signals, data signal staggering can be utilized. Different schemes can be enabled using the DPSTGER bits (G1CON2<13:12>). Figure 43-13 illustrates how the data signals going to the display are delayed using the GPUCLK. Since the GCLK runs at a lower frequency, delaying the data signals will not violate timing requirements for the display data signals.

When the DPSTGER control bits are at the default value, '00', the data signals are all synchronized at one clock edge. When DPSTGER = 01, only the odd numbered data bits are delayed by ½ GPUCLK cycle. When DPSTGER = 10, only the even numbered data bits are delayed by one GPUCLK cycle. For DPSTGER = 11, the data signals are organized into four groups:

- GD<0>, GD<4>, GD<8>, GD<12> – are set as Group 0 and will not be delayed
- GD<1>, GD<5>, GD<9>, GD<13> – are Group 1 and will be delayed by ½ GPUCLK cycle
- GD<2>, GD<6>, GD<10>, GD<14> – are Group 2 and will be delayed by 1 GPUCLK cycle
- GD<3>, GD<7>, GD<11>, GD<15> – are Group 3 and will be delayed by 1½ GPUCLK cycles

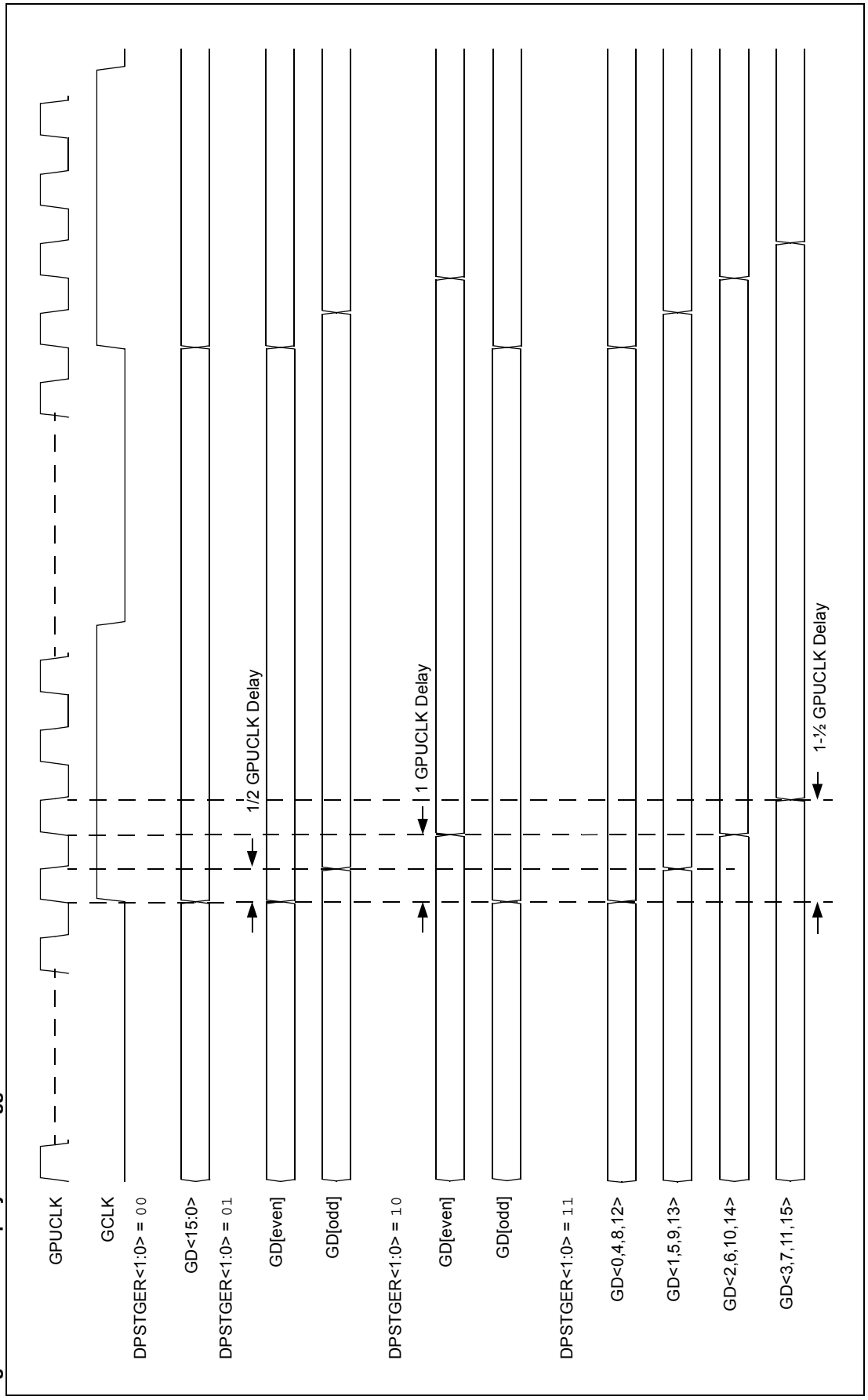


Figure 43-13: Display Data Stagger

Table 43-8 summarizes the frame rate calculation for different display types.

Table 43-8: Display Frame Rate Calculation

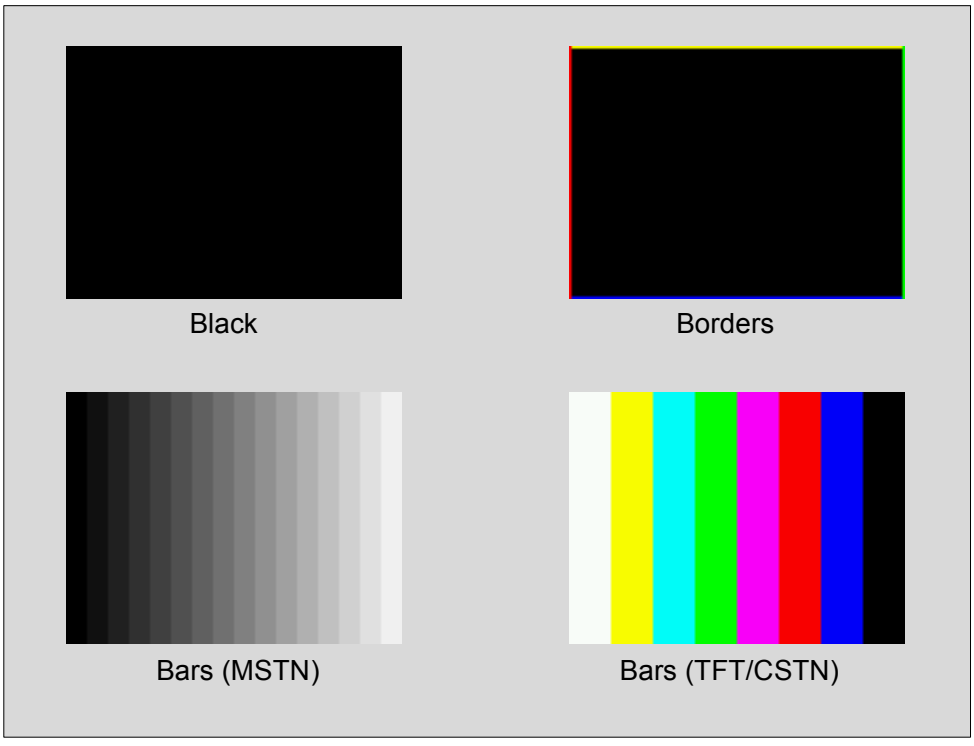
Display Type	DPGWDTH (G1CON2<15:14>)	Display Frame Rate
TFT	10 (16 bits wide)	GCLK Frequency/(DPWT * DPHT)
Mono STN	00 (4 bits wide)	(GCLK Frequency * 4)/(DPWT * DPHT)
	01 (8 bits wide)	(GCLK Frequency * 8)/(DPWT * DPHT)
	10 (16 bits wide)	(GCLK Frequency * 16)/(DPWT * DPHT)
Color STN	00 (4 bits wide)	(GCLK Frequency * 4)/(3 * DPW * DPHT)
	01 (8 bits wide)	(GCLK Frequency * 8)/(3 * DPW * DPHT)
	10 (16 bits wide)	(GCLK Frequency * 16)/(3 * DPW * DPHT)

Note: For CSTN, DPWT = (3/2) * DPW
GCLK Frequency = DISPCLK Frequency/DPGWDTH

43.4.5 Display Test Pattern Generator

A test pattern generator circuit is included in the module to test the display interface. The generator can be used to show a test pattern, using DPTEST<1:0> (G1CON2<9:8>). without the need for a working memory interface. After programming the registers appropriate to the chosen display, the selected test pattern will be displayed on the screen after the display controller is enabled. The available test patterns are Black, Bars and Borders. Figure 43-14 shows examples of the resulting test patterns.

Figure 43-14: Test Patterns



43.5 GRAPHICAL PROCESSING UNITS (GPUs)

43.5.1 Command Interface

Using the Graphical Processing Units (GPUs) requires programming of GCMD through G1CMDL<15:0> and the G1CMDH<15:0> registers. The programmed commands are sent to a 16-deep, 32-bit word FIFO. Commands are checked and routed to the specific GPU for execution. Only one GPU will be executing at any time. All queued commands will be waiting in the FIFO until a GPU is available to process it.

43.5.1.1 COMMAND FIFO STATUS

The command FIFO status can be monitored using three Status bits of the G1STAT register. The Command FIFO Full bit, CMDFUL (G1STAT<1>), indicates that the FIFO is full. The next command issued to a full FIFO is ignored and thrown away. The Command FIFO Empty bit, CMDMPT (G1STAT<0>), indicates that the FIFO is empty. This does not necessarily mean that all GPUs are Idle. It is possible that the last command in the FIFO is still being processed by one of the GPUs. This can be checked by the status bits of each GPU. The Command Watermark Level Status bit, CMDLV (G1STAT<2>), indicates that the number of commands in the FIFO is less than the programmed value of the Command Watermark Level bits, GCMDWMK<4:0> (G1CON1<12:8>). The Command Level Interrupt Flag bit, CMDLVIF (G1IR<2>), on the other hand, triggers when the number of commands present in the command FIFO decrements from the set command watermark level to a value below the set level. This is the only condition where the enabled CMDLVIF interrupt, using CMDLVIE (G1IE<2>), will trigger.

Figure 43-15 displays an example of how the command watermark level is used. In this example, GCMDWMK<4:0> = 4. While commands are sent to the FIFO, the different status flags of the command FIFO are set or reset depending on the number of commands present in the FIFO.

1. The FIFO starts as empty. Interrupts, CMDLVIF, CMDFULIF and CMDMPTIF, are disabled.
2. The number of commands (GCMDCNT<4:0>) sent to the FIFO grows to a total of 3. At this point, all interrupts are already enabled. Since the GDMDWMK bits are set to 4, the CMDLV status will remain at '1'. The CMDLVIF interrupt will not trigger since the command count is increasing.
3. If an additional command was sent, the CMDLV status will reset to '0' because it has reached the watermark level of 4. When the command count equals 16, the CMDFULIF interrupt will trigger.
4. GPUs can start consuming commands, even with one command present in the buffer. At this point, no more commands are sent to the FIFO. The command count starts to go down while the GPUs execute the commands.
5. The command count reaches 3. The CMDLVIF interrupt is triggered since the condition of the command count decrementing below the watermark level is met.
6. GPUs continue to consume commands and the GCMDCNT bits reach '0'; the CMDMPTIF is triggered and the CMDLV will remain set. Clearing CMDLVIF will not trigger the interrupt since the command count stayed equal, or less than, the watermark level of 3.
7. Commands are again sent to the FIFO, raising the count to 4. The CMDLV flag is reset since the watermark level is equal to the current command count.
8. A GPU consumes a command and the command count goes back to 3. The CMDLVIF interrupt is triggered since the condition of the command count decrementing below the watermark level is met; the CMDLV flag is also set.

Figure 43-15: Command Watermark Example

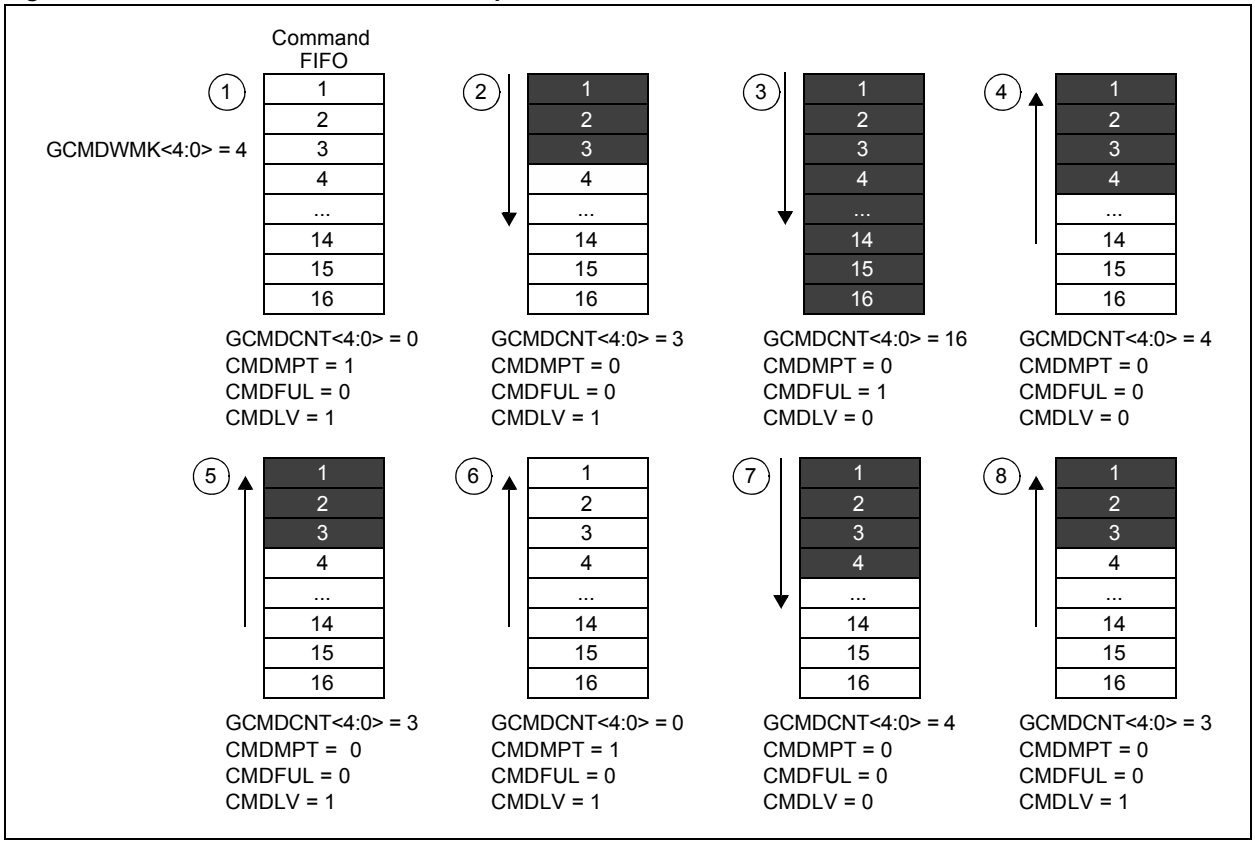
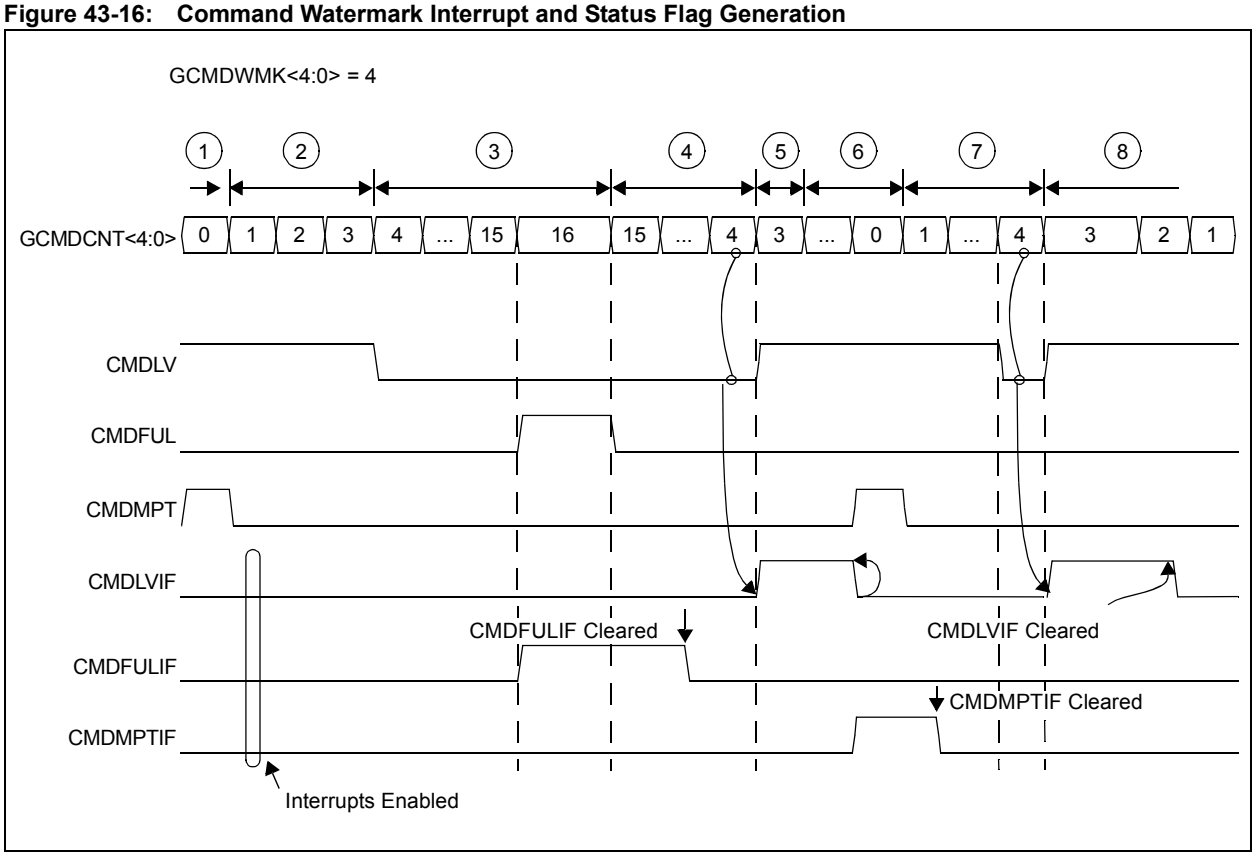


Figure 43-16: Command Watermark Interrupt and Status Flag Generation



43.5.1.2 COMMAND FORMAT

Table 43-9 provides the command format.

Table 43-9: GPU Command Format

GCMD<31:28>	GCMD<27:24>	GCMD<23:0>
GPU Address	Command Code	Command Attribute(s)
Specifies the GPU module that will process the commands.	Specifies the command that will be executed specific to the GPU module.	Specifies additional parameters for the given command.
GPU	**Address**	
CHRGPU	5	
RCCGPU	6	
IPU	7	

The 4 Most Significant bits (GCMD<31:28>) define the address of the GPU. Each command is identified by its command code (GCMD<27:24>) and the command attributes (GCMD<23:0>) are interpreted based on the given command code. Refer to **Section 43.5.2.2 "CHRGPU Commands"**, **Section 43.5.3.1 "RCCGPU Commands"** and **Section 43.5.4.1 "IPU Commands"** for the specific commands.

43.5.2 Character Graphics Processing Unit (CHRGPU)

The CHRGPU accelerates rendering of text by utilizing a font table, initialized in memory, and character codes sent through the GPU commands. The CHRGPU uses Work Area 1 with the starting address specified by W1ADR. This starting address is defined in the G1W1ADRL and G1W1ADRH registers. The width and height of Work Area 1 is specified using the PUW (G1PUW<10:0>) and PUH (G1PUH<10:0>) bits, respectively.

The initialized font table in memory contains the bit maps of the characters defined in the font. Multiple font tables can be initialized in memory. Selection of the font table used is defined by the command, `CHR_FONTBASE`. Each font table can have different heights, but all characters in one font table must have the same height though each character can have different widths. The characters are rendered into Work Area 1, using the set foreground color and background color. An option to render with transparency is also available. Within Work Area 1, a text area is specified by the command codes, `CHR_TEXTAREASTART` and `CHR_TEXTAREAEND`, where characters are allowed to be rendered. Within this text area, the upper left corner of the rendered character bit map is specified by the `CHR_SETPRINTPOS` command code. Any portion of the character bit map that falls outside of the text area will be clipped.

43.5.2.1 FONT TABLE FORMAT

Table 43-10 and Table 43-11 define the font table format and font table fields.

Table 43-10: Font Table Format

Byte Offset	Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Font Header																	
0x00		Reserved		Orient		Reserved				Font ID							
0x02		First Character ID															
0x04		Last Character ID															
0x06		Reserved								Height							
Character Table																	
0x08	1st Character	Offset <7:0>								Width							
0x0A		Offset <23:16>								Offset <15:8>							
0x0C	2nd Character	Offset <7:0>								Width							
0x0E		Offset <23:16>								Offset <15:8>							
…	…	…								…							
m-4	Last Character	Offset <7:0>								Width							
m		Offset <23:8>								Offset <15:8>							
Font Bit Maps																	
n	1st Character	1st Character Bit Map Data															
…																	
n	2nd Character	2nd Character Bit Map Data															
…																	
…	…	…															
n	Last Character	Last Character Bit Map Data															
…																	

Legend: m = (Last Character ID – First Character ID) + 1; n = Byte offset from font header start to the beginning of the character's bit map data defined by the character's Offset in the Character Table.

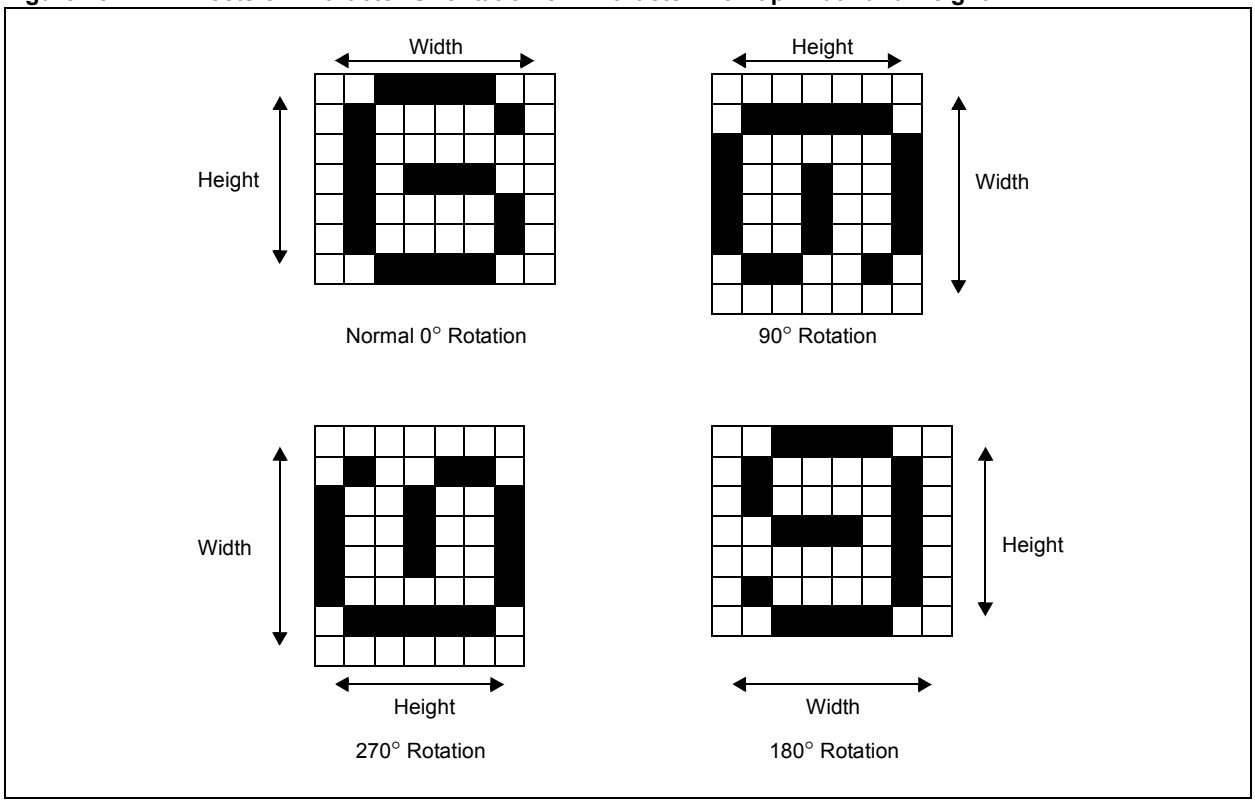
Table 43-11: Font Table Fields

Field	Bits	Description
Orient	2	Font Orientation: 00 = Normal 01 = Characters rotated 270 degrees clockwise 10 = Characters rotated 180 degrees 11 = Characters rotated 90 degrees clockwise
Font ID	8	User-defined value; CHRGPU ignores these bits.
First Character ID	16	Character ID for the first character in the font table.
Last Character ID	16	Character ID for the last character in the font table.
Height	8	Height of all the characters in the font table in pixels.
Width	8	Width of a character in pixels excluding the padding bits.(1)
Offset	24	Byte offset from the start of the font header to the beginning of the character's bit map.

Note 1: In cases where the character width is not equal to a byte boundary, padding bits are required. These padding bits are ignored by the CHRGPU when characters are rendered into the work area.

The font orientation field of the table is used for compensating the font characters if the display used is mounted in a rotated manner or if rotated text is required. Figure 43-17 shows how the font orientation rotates the characters. The width and height of the characters, when viewed on a 0° rotation is always the same. These are the same the same width and height values in the font table width and height fields, regardless of orientation.

Figure 43-17: Effects of Character Orientation on Character Bit Map Width and Height



Each character bit map represents one pixel. This means that the foreground color will be used for bits with the value of '1' and the background color will be used for those with the value of '0'. If the transparency bit is set in the CHR_PRINTCHAR command (see **Section 43.5.2.2 "CHRGPU Commands"**), the background color will not be rendered into the memory or work area.

The character table defines the width of each character and the character bit map location. This location is given by the byte offset from the start of the font header. The character table must contain entries for all the character codes from the first character ID to the last character ID. In some cases, not all characters in the character ID range are used. To save memory, these unused character IDs may not have a bit map defined. But since it is vital that all entries in the character table should be valid, characters that have no bit maps can be implemented using one of the following techniques: set the width of the character to 0 (characters will not be rendered), point to a specific error bit map (typically a rectangle) or point to the closest valid previous character bit map

Character bit maps can start on any byte address and each bit map line must start on a byte address. Depending on the font orientation used, the character bit map will be stored differently. As mentioned earlier, rotated displays will also need rotated fonts since each character rendered to the screen is always written to the display buffer (memory) in a normal orientation of the display. Due to this, each bit map row size will have two values: one for a 0° and 180° rotation, and one for 90° and 270° rotation. The row size is used to calculate how the bit maps are stored in memory. Since each line of a character bit map must start on a byte boundary, it is necessary to pad bit maps when the row size is not a multiple of 8. Note that for 90° and 270° rotation, the character widths are not equal to the row size.

Figure 43-18 shows an example of bit maps with 0° orientation; the height is 7 pixels. The three characters have widths of 11, 7 and 14 pixels. The first character is padded with 5 bits for each line. Figure 43-19 shows an example of a 90° orientation. In this example, the same characters are rotated 90°. The row size of all three characters is equal. Only 1-bit padding is needed to conform to the requirement and each of the bit map lines must start on a byte boundary.

Figure 43-18: Bit Maps with 0° Orientation

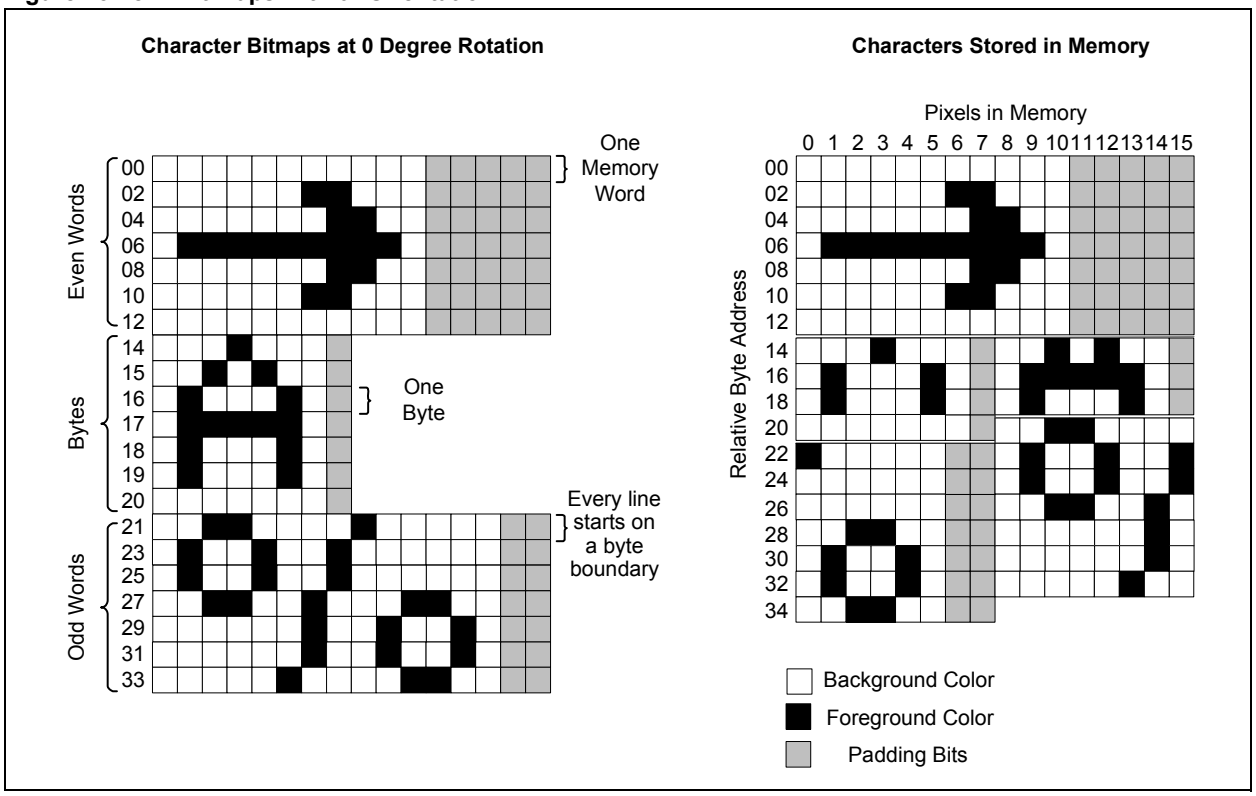
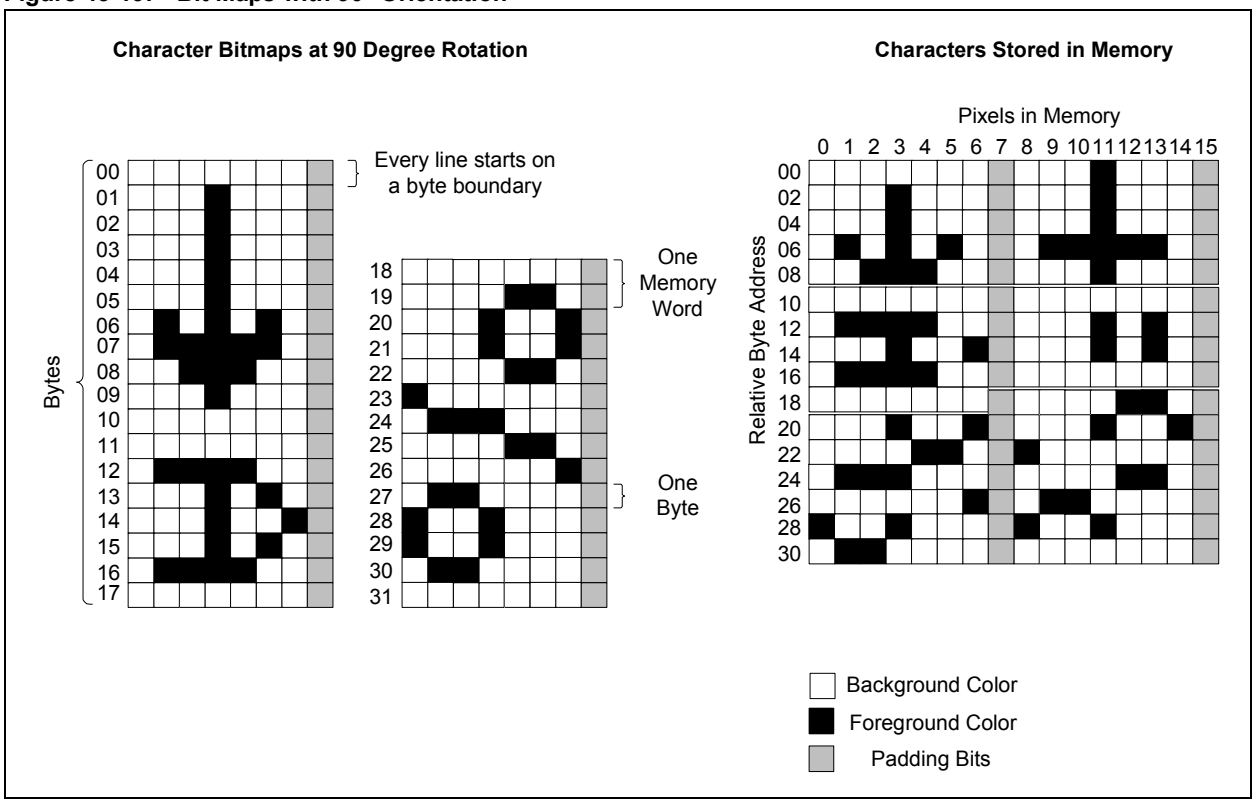


Figure 43-19: Bit Maps with 90° Orientation



43.5.2.2 CHRGPU COMMANDS

Table 43-12 through Table 43-18 list the commands available for CHRGPU operations.

Table 43-12: CHR_FGCOLOR Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:16>	GCMD<15:0>
0x5	**0x0**	**Reserved**	**Color**
CHRGPU Address	Set Foreground Color Command	—	Foreground Color Value
Sets the foreground color of the characters rendered. Note that the color value can be a color index to the CLUT if CLUT is enabled.			

Table 43-13: CHR_BGCOLOR Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:16>	GCMD<15:0>
0x5	**0x1**	**Reserved**	**Color**
CHRGPU Address	Set Background Color Command	—	Background Color Value
Sets the background color of the characters rendered. Note that the color value can be a color index to the CLUT if CLUT is enabled.			

Table 43-14: CHR_FONTBASE Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:0>
0x5	**0x2**	**Byte Address**
CHRGPU Address	Set Font Table Base Address Command	Base address of the font table to be used.
Sets the base address of the font table to be used. The byte address must be aligned to a memory word boundary.		

Table 43-15: CHR_PRINTCHAR Command

GCMD<31:28>	GCMD<27:24>	GCMD<23>	GCMD<22:16>	GCMD<15:0>
0x5	**0x3**	**Tr**	**Reserved**	**Character Code**
CHRGPU Address	Print Character Command	Transparency Bit	—	This specifies the character to be rendered.
Renders a character using the bit map associated with the character code. A special case in the operation of CHRGPU is the new line character. When new line character code is detected, the Y position of the next rendered character is adjusted to the next line. The adjustment is equal to the character height set for the currently used font table. If the Tr bit is set, the background color set in CHR_BGCOLOR will not be drawn.				

Table 43-16: CHR_TXTAREASTART Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:12>	GCMD<11:0>
0x5	**0x8**	**X**	**Y**
CHRGPU Address	Set Text Area Start Command	X coordinate position of the upper left corner of the text area.	Y coordinate position of the upper left corner of the text area.
Sets the upper left corner of the rectangular text area where characters will be allowed to be rendered. If CHR_PRINTPOS is set outside the rectangular area, only portions of the characters that fall within the rectangular area (inclusive of the edges) specified by CHR_TXTAREASTART and CHR_TXTAREAEND will be allowed.			

Table 43-17: CHR_TXTAREAEND Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:12>	GCMD<11:0>
0x5	**0x9**	**X**	**Y**
CHRGPU Address	Set Text Area End Command	X coordinate position of the lower right corner of the text area.	Y coordinate position of the lower right corner of the text area.
Sets the lower right corner of the rectangular text area where characters will be allowed to be rendered.			

Table 43-18: CHR_PRINTPOS Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:12>	GCMD<11:0>
0x5	**0xA**	**X**	**Y**
CHRGPU Address	Set Print Position Command	X coordinate position of the next character to be rendered.	Y coordinate position of the next character to be rendered.
Sets the character print position. This is the X-Y position of the upper left corner of the next rendered character specified by the CHR_PRINTCHAR command.			

43.5.2.3 CHRGPU OPERATION

Text rendering can be performed using the following steps:

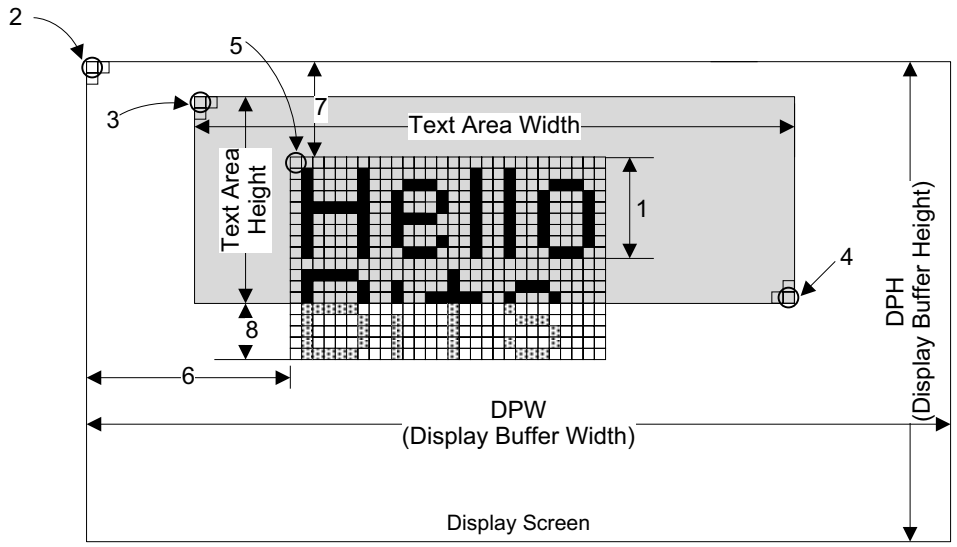
1. Store one or more font tables in memory.
2. Set the font base address using CHR_FONTBASE.
3. Set the text area, defined by CHR_TXTAREASTART and CHR_TXTAREAEND, where the characters will be rendered.
4. Set the foreground and background colors using CHR_FGCOLOR and CHR_BGCOLOR.
5. Set the print position using CHR_PRINTPOS.
6. Print characters using CHR_PRINTCHAR.

Steps 2-5 can be interchanged with the assumption that all font tables referred to by CHR_FONTBASE are already initialized in memory.

Characters are rendered into the work area by specifying the Character ID, and the ‘X’ and ‘Y’ position of the character on the screen. The ‘X’ and ‘Y’ values will determine the memory location of the upper left pixel of the rectangular area affected by the character rendering. The dimension of this rectangular area is determined by the character width and height. Only pixels of this rectangular area that fall within the text area (defined by CHR_TXTAREASTART and CHR_TEXTAREAEND) will be affected. Since the CHRGPU works only with Work Area 1, it is logical to make the width of Work Area 1 equal to that of the display buffer. By making them equal, each pixel referenced by the CHRGPU will always correspond to the same pixels in the display buffer.

Figure 43-20 provides an example of character rendering on the display buffer declared as Work Area 1. It is possible to declare a work area outside of the display buffer. When declaring Work Area 1 in a separate memory location, it is important that the widths for Work Area 1 and the display buffer are equal. This makes the intermediate step of copying the contents of Work Area 1 into the display buffer simple and straight forward. If the widths are not equal, application code must take into account the difference and make necessary adjustments in the data copying. If this is not performed, the image displayed on the screen will be broken.

Figure 43-20: CHRGPU Work Area Rendering



1. Font Height
2. W1ADR<23:0>
3. CHR_TXTAREASTART
4. CHR_TXTAREAEND
5. Upper Left Pixel Set by CHR_SETPRINTPOS
6. Upper Left Pixel X Position
7. Upper Left Pixel Y Position
8. Portion of Characters that Exceed the Text Area will Not be Drawn

Note: Work Area 1 width is equal to the display buffer width.

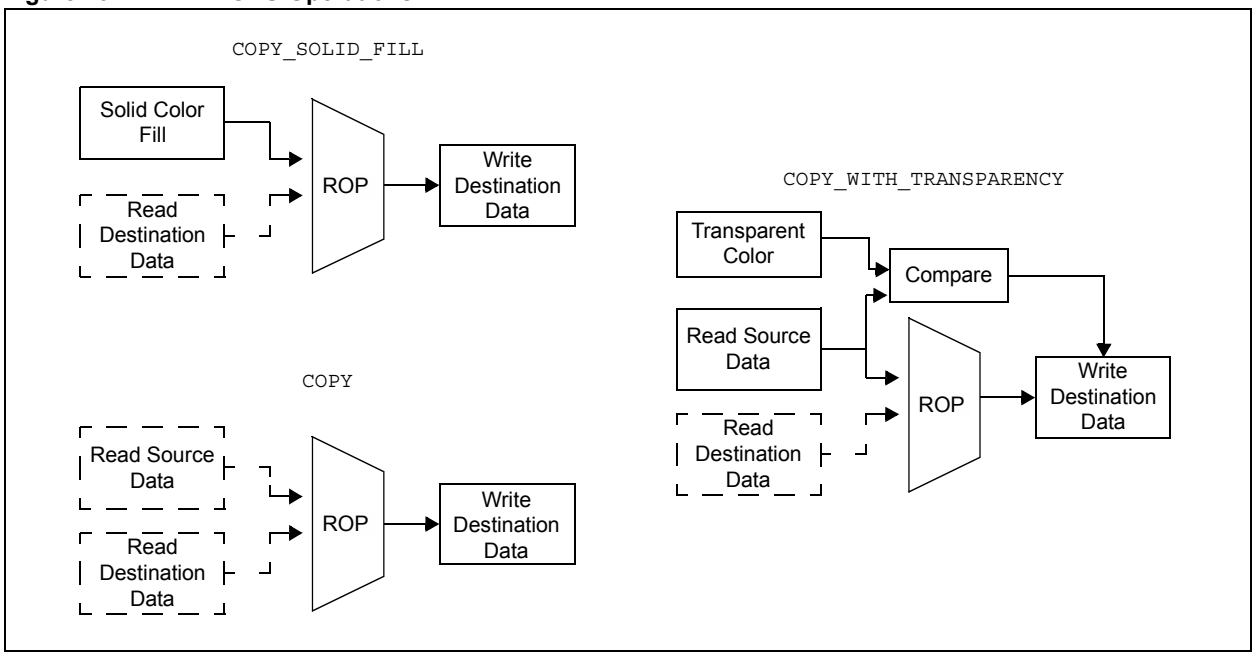
Note:	Rendering long strings in the 'X' or 'Y' direction that exceed the display screen dimension will still be executed. Once the internal counters that track the 'X' and 'Y' position of the characters reach the maximum, the counters will loop back to zero. Once this occurs, the next characters of the string will be rendered into the viewable area depending on the dimension set on the text area start and text area end commands. Software can check the current character print position through the G1CHRX and G1CHRY registers to detect if the character position is beyond the display screen.

43.5.3 Rectangle Copy Graphics Processing Unit (RCCGPU)

The Rectangle Copy Graphics Processing Unit (RCCGPU) is used to copy a rectangular area from one memory location to another. The source rectangular area can be specified as continuous data in memory or discontinuous data in memory. Similarly, the destination of the copied rectangular area can also be specified as continuous or discontinuous memory. This feature simplifies software routines when displaying images or texts into the screen.

There are three types of copy operation: COPY, COPY_SOLID_FILL and COPY_WITH_TRANSPARENCY. Figure 43-21 shows the three copy operations.

Figure 43-21: RCCGPU Operations



The source data is specified by Work Area 1 (W1ADR) and the destination by Work Area 2 (W2ADR). Depending on the type of copy operation, the source data and destination data are copied, modified or ignored.

Table 43-19: RCCGPU Operations

Name	Value	Description
COPY	001	Copy with ROP; performs a copy from source to destination rectangle with the selected ROP.
COPY_SOLID_FILL	000	Operation is the same as the COPY operation except that the source data is not used and the color value set with the RCC_COLOR command is utilized for all source pixels.
COPY_WITH_TRANSPARENCY	110	Operation is the same as the COPY operation except that the source data is compared against the color value set with the RCC_COLOR command. If the values match, the source data is not written to the destination. The source image is, therefore, transparent at such a location, allowing the pre-existing destination image to be preserved.

Each copy operation can be performed with a specified Raster Operation (ROP). Specific shapes such as rectangles, vertical and horizontal lines as well as individual pixel manipulation can be performed using different combinations of copy operation and ROP.

Table 43-20 enumerates the available ROP commands and their corresponding equations.

Table 43-20: RCCGPU Raster Operation Commands

Name	Value	Source (S) and Destination (D) Logical Operation
ROP_0	0000	0 (Black)
ROP_1	0001	not (S or D)
ROP_2	0010	(not S) and D
ROP_3	0011	not S
ROP_4	0100	S and (not D)
ROP_5	0101	not D
ROP_6	0110	S XOR D
ROP_7	0111	not (S and D)
ROP_8	1000	S and D
ROP_9	1001	not (S XOR D)
ROP_A	1010	D
ROP_B	1011	(not S) or D
ROP_C	1100	S
ROP_D	1101	S or (not D)
ROP_E	1110	S or D
ROP_F	1111	1 (White)

RCCGPU uses pixel word addresses when specifying the source and destination memories; however, RCCGPU always executes operations on individual pixels. COPY and COPY_WITH_TRANSPARENCY always enables the source data. If the ROP operation used is ROP_0 or ROP_F, the pixels of the source data with the same color as the transparency color will not be affected by the operation. This behavior can be utilized to clear portions of a work area, with black or white colors, with the area having the transparency color untouched.

Note: The ROPs always operate on the pixel data. The effect of the ROP is different depending upon whether or not the CLUT is being used. If CLUT is disabled, and for example, ROP_E is selected, an OR of a Red pixel and a Blue pixel will result in a Magenta colored pixel. If CLUT is enabled, and for example, ROP_E is selected, an OR of a pixel value of 4 and a pixel value of 2 will result in a pixel value of 6. The resulting color is determined by the color value written into the CLUT at address 6.

43.5.3.1 RCCGPU COMMANDS

Table 43-21 through Table 43-25 list the commands available for RCCGPU operations.

Table 43-21: RCC_SRCADDR Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:0>
0x6	**0x2**	**Pixel Word Address**
RCCGPU Address	Set Source Address Offset Command	Source Address Offset
Sets the source address offset of the source rectangle. The offset points to the pixel word address that contains the upper left corner of the rectangle. The effective source address is the sum of the Work Area 1 address plus the source address offset.		

Table 43-22: RCC_DESTADDR Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:0>
0x6	**0x3**	**Pixel Word Address**
RCCGPU Address	Set Destination Address Offset Command	Destination Address Offset
Sets the destination address offset of the destination rectangle. The offset points to the pixel word address that contains the upper left corner of the rectangle. The effective destination address is the sum of the Work Area 2 address plus the destination address offset.		

Table 43-23: RCC_RECTSIZE Command

GCMD<31:28>	GCMD<27:24>	GCMD<23>	GCMD<22:12>	GCMD<11>	GCMD<10:0>
0x6	**0x4**	**Reserved**	**Width**	**Reserved**	**Height**
RCCGPU Address	Set Rectangle Size Command	—	Rectangle Width	—	Rectangle Height
Sets the width and height in pixels of the rectangle to be copied.					

Table 43-24: RCC_COLOR Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:16>	GCMD<15:0>
0x6	**0x6**	**Reserved**	**Color**
RCCGPU Address	Set Color Command	—	Color value to be used.
Sets the color to be used in the selected ROP, such as solid fill and copy with transparency.			

Table 43-25: RCC_STARTCOPY Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:10>	GCMD<9:7>	GCMD<6:3>	GCMD<2>	GCMD<1>	GCMD<0>
0x6	**0x7**	**Reserved**	**OPER**	**ROP**	**DT**	**ST**	**Reserved**
RCCGPU Address	Set Copy Command	—	—	—	—	—	—

RCC_STARTCOPY command executes the copy with the selected raster operation.

- ST – Specifies the source address type:
  - 0 = Source address is discontinuous type. Use this setting when the rectangle is stored within an area in memory which can become the display buffer (i.e., will be shown in the display). In this case, there is a memory address discontinuity between the right most pixel of the rectangle line and the left most pixel of the next line.
  - 1 = Source address is continuous type. Use this setting when the rectangle is stored in an area in memory which will not be used as a display buffer. For example, a scratchpad in memory is needed to operate on some data before it is shown in the display. In this case, the rectangle is stored in a more compact way with no address discontinuity between the right most pixel of a rectangle line and the left most pixel of the next line.
- DT – Specifies the destination address type. This is analogous to the description of ST.
- ROP – Selects the raster operation to be performed. The operations are performed on each pixel of the source and destination rectangles with a bit-wise logical operation of the source and destination pixel color values. Refer to Table 43-20 for a list of possible raster operations.
- OPER – Selects the type of operation to be executed by RCCGPU (refer to Table 43-19 for details):
  - 000 = Solid fill with ROP
  - 001 = Copy with ROP
  - 110 = Copy with ROP and transparency

43.5.3.2 EXAMPLES

The following examples show how RCCGPU is used to generate simple objects. Work Area 1 (W1ADR<23:0>) is always set as the source and Work Area 2 (W2ADR<23:0>) as the destination.

43.5.3.2.1 One Pixel Point

- Set color value to the desired pixel color (RCC_COLOR)
- Set the pixel offset destination address (RCC_DESTADDR)
- Set width = height = 1 (RCCRECTSIZE)
- Start RCCGPU with OPER = 000, ROP = ROP_C and DT = address type of destination

43.5.3.2.2 Solid Colored Vertical/Horizontal Single Pixel Thick Line

- Set color value to the desired line color (RCC_COLOR). The color chosen will serve as source data and Work Area 1 is ignored.
- Set the pixel offset destination address; this will be the address of the upper most pixel for a vertical line or the address of the left most pixel for a horizontal line (RCC_DESTADDR)
- For Vertical Line: Set rectangle size width = 1 and height = line length for vertical line (RCCRECTSIZE)
- For Horizontal Line: Set width = line length and height = 1 for horizontal line (RCCRECTSIZE)
- Start RCCGPU with OPER = 000, ROP = ROP_C and DT = address type of destination; source will be ignored in this operation since it is not operated on by the selected ROP and OPER

43.5.3.2.3 Rectangle

When doing a rectangle with n-pixel width lines, perform the solid line operation above to create the four sides of the rectangle.

43.5.3.2.4 Solid Rectangles

- Set color value to the desired color (RCC_COLOR). The color chosen will serve as source data and Work Area 1 is ignored.
- Set the pixel offset destination address; this will be the address of the pixel that defines the upper left corner of the rectangular shape (RCC_DESTADDR)
- Set width = width of the rectangle and height = height of the rectangle (RCCRECTSIZE)
- Start RCCGPU with OPER = 000, ROP = ROP_C and DT = address type of destination

43.5.3.2.5 Copying Rectangular Area to a Different Location

- Set the pixel offset source address; this will be the address of the pixel that defines the upper left corner of the source (RCC_SRCADDR)
- Set the pixel offset destination address; this will be the address of the pixel that defines the upper left corner of the destination (RCC_DESTADDR)
- Start RCCGPU with OPER = 001, ROP = ROP_C, and ST and DT = address type of source and destination

43.5.3.2.6 Copying Generic Range of Continuous Data

- Set the offset source address of the first byte to be copied (RCC_SRCADDR)
- Set the offset destination address of the first copied byte (RCC_DESTADDR)
- Set the rectangle size height = 1 and width = # of bytes to copy * 8/bits-per-pixel (RCC_RECTSIZE), where the division result is rounded to the next upper address
- Start RCCGPU with OPER = 001 and ROP = ROP_C; ST and DT settings will not matter here since the operation will be performed with height = 1

43.5.4 Inflate Processing Unit (IPU)

The IPU is used to decompress data located in one area in memory and store the decompressed data to another area in memory. Compressed data is encoded using the DEFLATE algorithm with fixed Huffman codes; dynamic Huffman codes are not supported. This feature is especially useful in applications with limited memory resources.

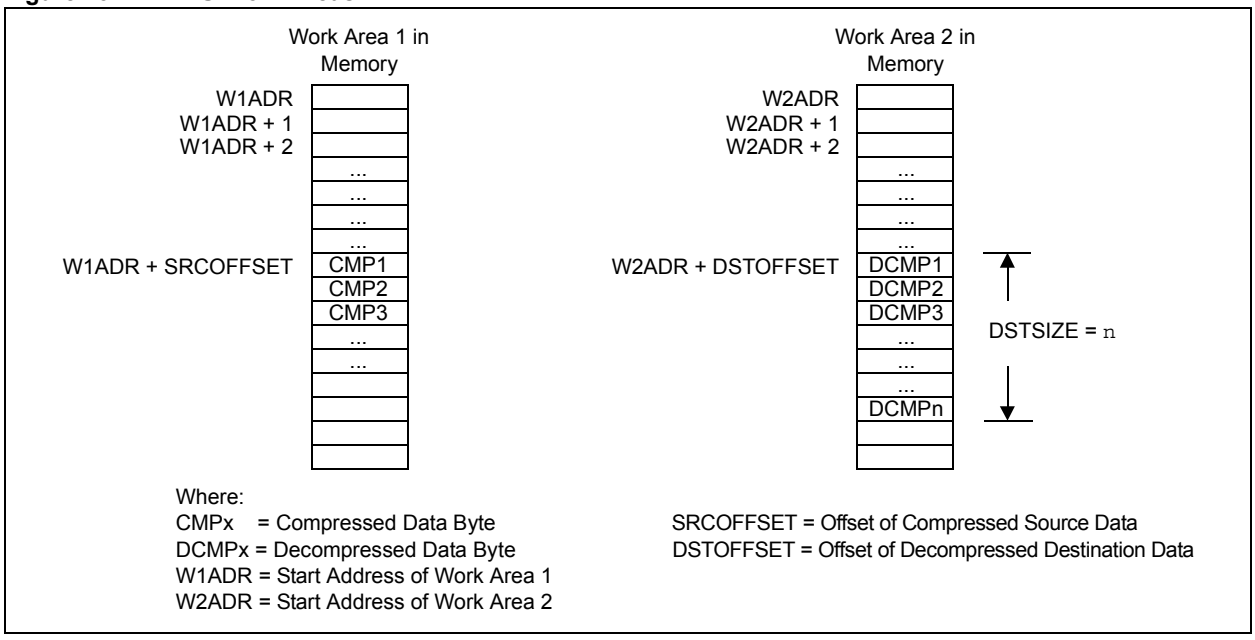
For example, an application that uses one font at a time, but will need to switch to different fonts at run time, can precompress all the fonts and use the IPU to decompress the selected font into one area in memory.

IPU also uses two work areas: Work Area 1 will contain the compressed data and Work Area 2 will hold the decompressed data. Similar to the RCCGPU, the two work areas are initialized using the W1ADR and W2ADR registers. The following steps are taken to decompress data using the IPU:

- Initialize addresses for Work Area 1 and Work Area 2 through the W1ADR and W2ADR bits.
- Set the effective location of the compressed data in Work Area 1 by setting the offset using the IPU_SRCADDR command. The effective source byte address must point to the start of the compressed block. If this is not met, decompression will fail. Effective Source Byte Address = Work Area 1 + SRCOFFSET (source offset).
- Set the effective location for the decompressed data in Work Area 2 by setting the offset, using the IPU_DESTADDR command. Effective Destination Byte Address = Work Area 2 + DSTOFFSET (Destination Offset).
- Start the decompression operation with the size of the uncompressed data (DSTSIZE), in bytes, using the IPU_DECOMPRESS command.
- Using the IPU status register, G1IPU, monitor the status of the decompression.

Figure 43-22 illustrates how the source data initialized in memory is accessed by IPU and the resulting decompressed data written to Work Area 2.

Figure 43-22: IPU Work Areas



43.5.4.1 IPU COMMANDS

Table 43-26 through Table 43-28 list the commands available for IPU operations.

Table 43-26: `IPU_SRCADDR` Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:0>
0x7	**0x1**	**SRCOFFSET**
IPU Address	Set Source Address Offset Command	Source Address Offset
Sets the source address offset of the first compressed byte from the start of Work Area 1.		

Table 43-27: `IPU_DESTADDR` Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:0>
0x7	**0x2**	**DSTOFFSET**
IPU Address	Set Destination Address Offset Command	Destination First Byte Address Offset
Sets the destination address offset of the first decompressed byte from the start of Work Area 2.		

Table 43-28: `IPU_DECOMPRESS` Command

GCMD<31:28>	GCMD<27:24>	GCMD<23:0>
0x7	**0x4**	**DSTSIZE**
IPU Address	Start Decompress Command	Decompressed Data Size
Starts the decompression of the compressed data with the specified amount of bytes.		

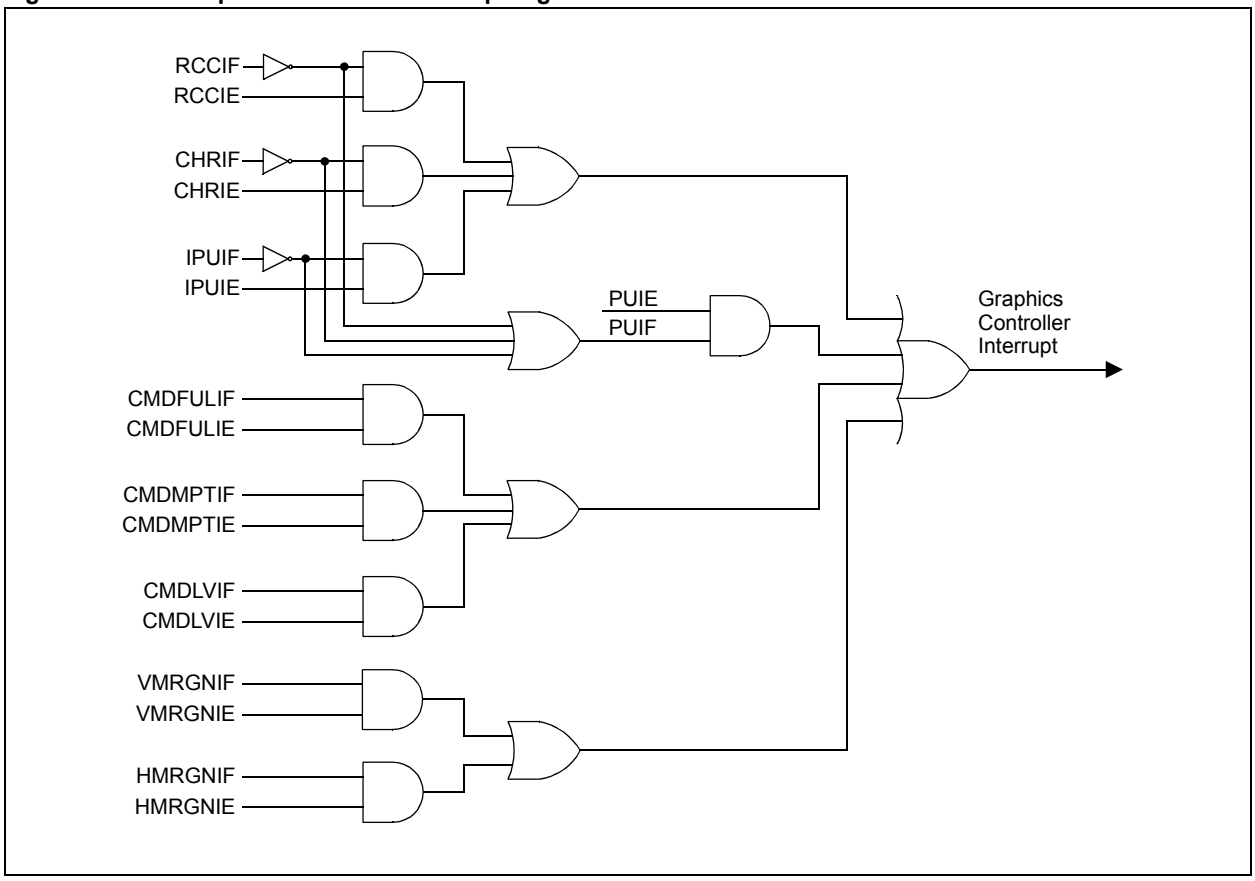
Although the IPU commands allow decompressed data to start on a byte location, it is possible that decompressed data must be placed on a memory word boundary. An example would be a font table, used by CHRGPU, or images directly decompressed to the display buffer.

43.6 INTERRUPTS

The Graphics Controller module provides nine interrupts; four are provided for GPU operation, two to indicate blanking periods and three for command FIFO operation.

Each interrupt source in the module has an interrupt status bit and a corresponding enable bit. All interrupts are then combined to form one Graphics Controller interrupt signal to the CPU (see Figure 43-23). The graphics Interrupt Service Routine (ISR) must then determine which graphics event(s) caused the CPU interrupt and services them appropriately. The Graphics Controller interrupt(s) are enabled through the G1IE register. The status of the interrupts is checked through the G1IR register. If no interrupt is enabled, status of the events can still be checked through the G1STAT register.

Figure 43-23: Graphics Controller Interrupt Signal



43.6.1 GPU Interrupts

Since only one GPU can be actively executing a command at any given time, each GPU is assigned an interrupt. Applications can use the IPUIF, RCCIF and CHRIF to detect each GPU entry to Idle and inform the application that the GPU is ready for new commands. Collective detection of GPU busy is also available through PUIF. This interrupt asserts when one of the GPUs executes a command. Notice the reversed assertion of the interrupt. For each dedicated GPU interrupt, interrupts assert when their corresponding GPU enters Idle mode. Collectively, the PUIF interrupt asserts when one GPU executes a command.

43.6.2 Command FIFO Interrupts

Refer to **Section 43.5.1 "Command Interface"** for details.

43.6.3 Blanking Interrupts

In TFT mode, blanking periods are a portion of the scanning process when refreshing the display screen. Vertical blanking period is composed of the vertical front and back porch, and the vertical synchronization pulse. Similarly, horizontal blanking is composed of the horizontal front and back porch, and the horizontal synchronization pulse. It is during these periods that the screen is not actively being refreshed or no pixel in the screen is being modified. Application may use these periods to perform operations that may affect the display refresh. An example of such operation is the CLUT modification.

The start of the vertical blanking interrupt (VMRGNIF) is enabled through the VMRGNIE (G1IE<4>) enable bit, while the start of horizontal blanking (HMRGNIF) is enabled through the HMRGNIE (G1IE<3>) enable bit.

43.7 DISPLAY POWER SEQUENCING

The display controller features display power sequencing. The generation of the display power signal, GPWR, and display enable signal, GEN, follows the display power states sequence displayed in Figure 43-25. For display units that require several voltages with sequence requirements, external circuitry must be implemented.

Figure 43-24: External Display Power Sequencing Circuitry

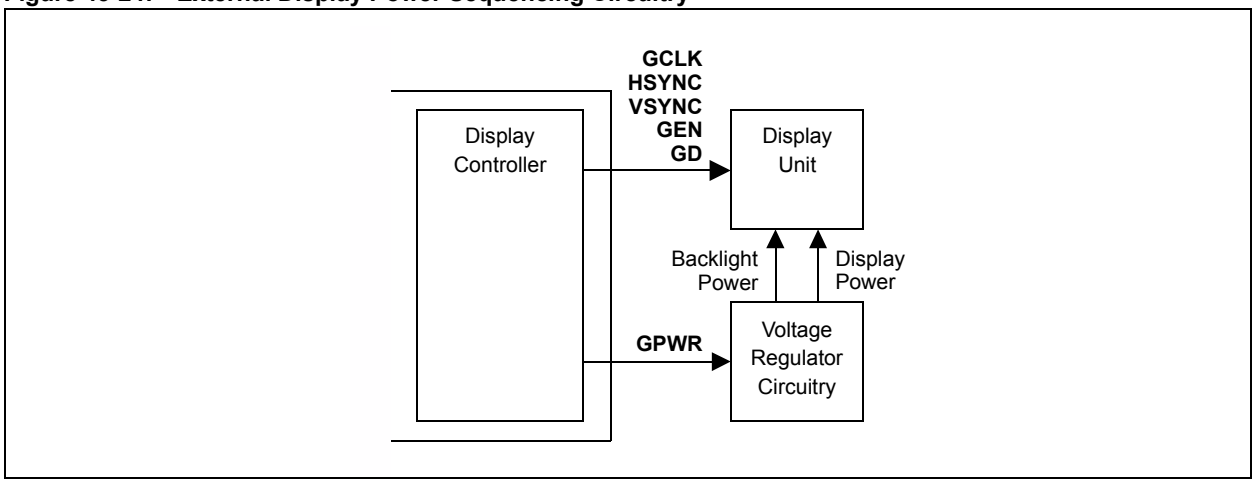
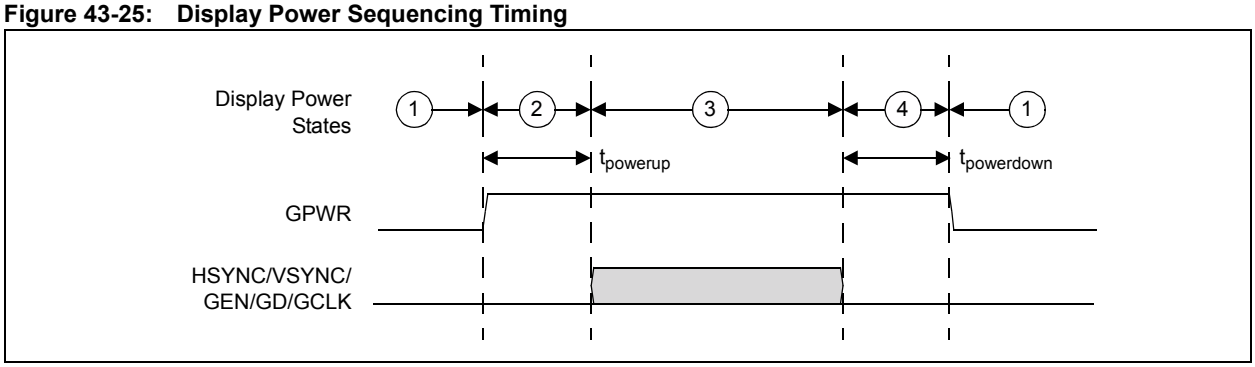


Figure 43-25: Display Power Sequencing Timing



The whole sequence is initiated by the DPPOWER bit (G1CON3<8>) set to enable. The GPWR signal to the display module is generated after some delays due to the synchronization of the SFR clock domain to the GLCK domain. If DPPOWER is disabled, the display power state is in the OFF state, indicated as 1 in Figure 43-25. When DPPOWER is set, the state is in the power-up state, as indicated by 2. Once the GLCK, HSYNC, VSYNC and GEN are active, the state is in the power-on state, as indicated by 3. DPPOWER can be reset in the register at any time. When this occurs, the display power state enters the power-down state. This is where the GEN, HSYNC, VSYNC and GLCK are set to their inactive levels, and eventually the GPWR signal is inactive, as indicated by 4. Once GPWR is reset to the inactive state, the state is back to the power-down state.

The GPWR assertion to HSYNC/VSYNC/GEN/GLCLK active ($t_{powerup}$) and HSYNC/VSYNC/GEN/GLCLK active to the GPWR deassertion ($t_{powerdown}$) is determined by Equation 43-16.

Equation 43-16:

$$t_{powerup} \text{ or } t_{powerdown} = \text{GICLUTWR<15:0> x 64 GCLK Cycles}$$

The G1CLUTWR register must be written with the correct value prior to enabling or disabling the DPPOWER register bit. The written value is dependent on the specifications of the display glass used.

It is not mandatory to use the GPWR and GEN signals from the display controller; externally generated signals are also possible.

43.8 OPERATION IN POWER-SAVING MODES

The PIC24F family of devices has three power modes: the Normal Operational (Full-Power) mode and the two power-saving modes, invoked by the PWRSAV instruction. Depending on the mode selected, entering a power-saving mode may also affect the operation of the module.

43.8.1 Sleep Mode

When the device enters Sleep mode, all clocks going into the Graphics module are disabled. This effectively stops the module operation. A display connected to the Graphics module port may suffer permanent damage if proper power-down sequencing is not performed before the device enters the Sleep mode. The application must make sure that proper power-up and power-down sequencing is performed when the device is entering and leaving Sleep mode.

43.8.2 Idle Mode

For the Graphics module, the G1SIDL bit (G1CON1 <13>) selects if the module will remain active or stop in Idle mode. If G1SIDL = 0, the module will continue normal operation when the device enters the Idle mode. All enabled interrupts in the module will trigger when the module meets the interrupt condition. If G1SIDL = 1, the module will behave similarly to Sleep mode.

43.9 EFFECTS OF A RESET

All forms of Reset force the Graphics module registers to the default state. Queued commands in the command FIFO are erased and any executing GPU will terminate and return to Idle state when re-enabled.

If an external power sequencer is connected to the display, the GPWR signal can be used to detect if the Graphics module is resetting or powering up. If the G1CLUTWR register is used to delay the GPWR signal generation to the display, the application should perform a proper power-down and power-up sequence on the display whenever an application controlled Reset of the device is performed (for example, a CPU Reset).

Note:	The Graphics Controller module can neither ensure the state of Work Areas 1 and 2, nor that of the display buffers contained in RAM following a Reset.

43.9.1 Device Reset ($\overline{\text{MCLR}}$)

A device Reset forces all Graphics Controller registers to their Reset state. This turns the module off.

43.9.2 Power-on Reset (POR)

A Power-on Reset forces all Graphics Controller registers to their Reset state. This turns the module off.

43.9.3 Watchdog Timer Reset (WDT)

A WDT Reset forces all Graphics Controller registers to their Reset state. This turns the module off.

43.10 DISPLAY INTERFACE TIMING

43.10.1 TFT Interface Timing

Figure 43-26: TFT Display Interface AC Timing Diagram

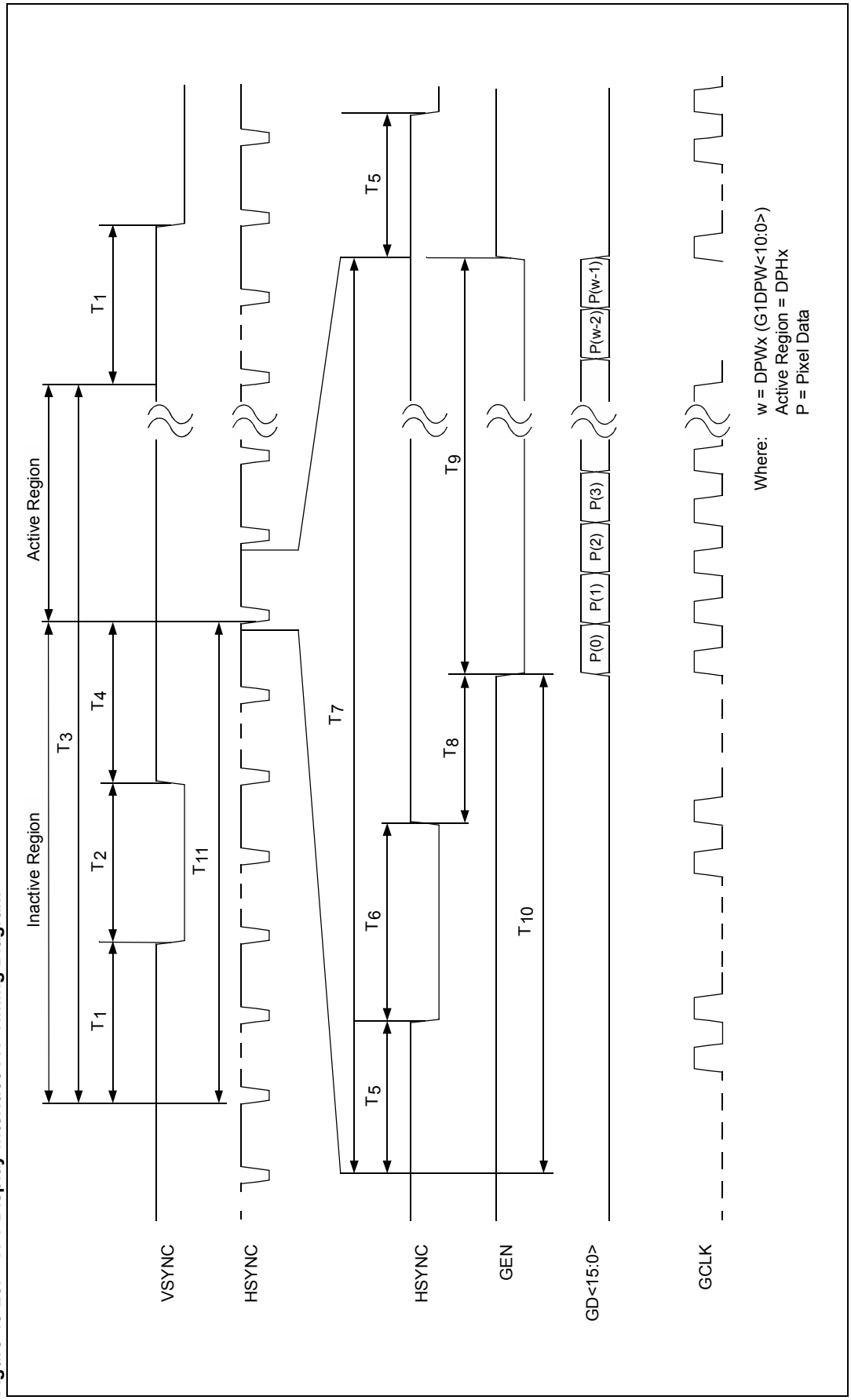


Table 43-29: TFT Display Interface AC Timing Requirements

Symbol	Parameter	Min	Typical	Max	Unit
T_1	VSYNC Start/Vertical Front Porch	0	VSST	—	Lines
T_2	VSYNC Width	1	VSLEN	—	Lines
T_3	VSYNC Period	—	DPHT	—	Lines
T_4	Vertical Back Porch	—	DPHT – (DPH + VSLEN + VSST)	—	Lines
T_5	HSYNC Start/Horizontal Front Porch	0	HSST	—	TGCLK
T_6	HSYNC Width	1	HSLEN	—	TGCLK
T_7	HSYNC Period	—	DPWT	—	TGCLK
T_8	Horizontal Back Porch	—	DPWT – (DPW + HSLEN + HSST)	—	TGCLK
T_9	Line Width	—	DPW	—	TGCLK
T_{10}	Active Pixel Start	1	HENST ($T_5 + T_6 + T_8$)	—	TGCLK
T_{11}	Active Line Start	2	VENST ($T_1 + T_2 + T_4$)	—	Lines

Note: TGCLK is the pixel clock (GCLK) period. All signals are assumed to be active-low.

43.10.2 STN Interface Timing

Figure 43-27: STN Display Interface AC Timing Diagram

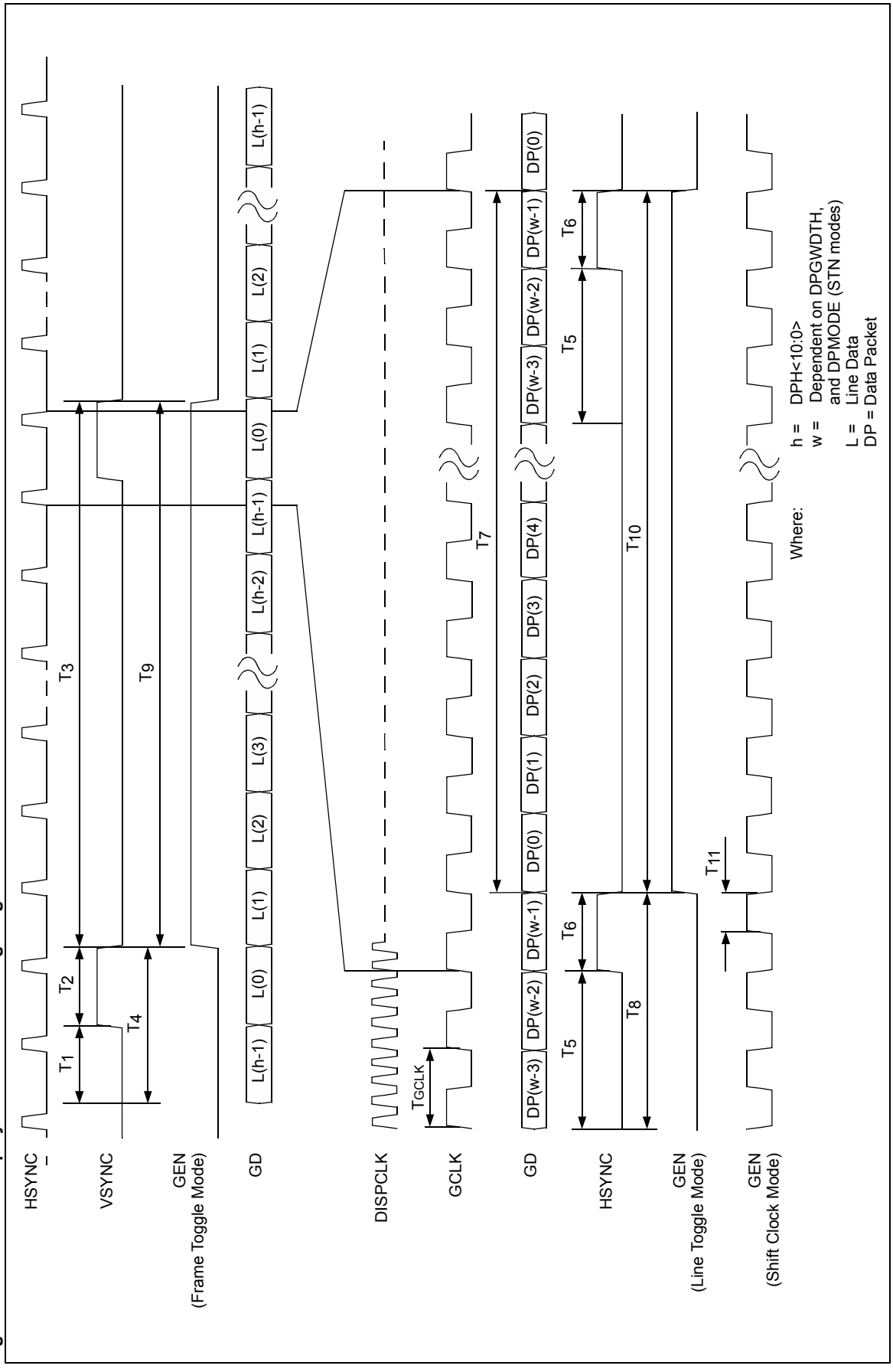


Table 43-30: STN Display Interface AC Timing Requirements

Symbol	Parameter	Min	Typical	Max	Unit
TGCLK	Pixel Clock for MSTN	—	TDISPCLK/DPGWDTH(1)	—	TDISPCLK
TGCLK	Pixel Clock for CSTN	—	TDISPCLK/(DPGWDTH * 2)(1)	—	TDISPCLK
T_1	VSYNC Start	0	VSST	—	Lines
T_2	VSYNC Width	0	VSLEN	—	Lines
T_3	VSYNC Period		DPHT	—	Lines
T_4	Active Line Start	0	VENST (T_1 + T_2)	—	Lines
T_5	HSYNC Start	0	HSST	—	TGCLK
T_6	HSYNC Width	1	HSLEN	—	TGCLK
T_7	HSYNC Period		DPWT	—	TGCLK
T_8	Active Pixel Start	0	HENST (T_5 + T_6)	—	TGCLK
T_9	GEN Toggle Rate for Frame Toggle Mode	—	T_3	—	TGCLK
T_{10}	GEN Toggle Rate for Line Toggle Mode	—	T_7	—	TGCLK
T_{11}	GEN Toggle Rate for Shift Clock Mode	—	TGCLK/2	—	TGCLK

Note 1: TDISPCLK is the Display Clock (DISPCLK) period.

43.11 REGISTER MAP

Table 43-31: Graphics Controller Register Map

File Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
G1CMDL	GCMD<15:0>																0000
G1CMDH	GCMD<31:16>																0000
G1CON1	G1EN	—	G1SIDL	GCMDWMK<4:0>					PUBPP<2:0>			GCMDCNT<4:0>					0000
G1STAT	PUBUSY	—	—	—	—	—	—	—	IPUBUSY	RCCBUSY	CHRBUSY	VMRGN	HMRGN	CMDLV	CMDFUL	CMDMPT	0000
G1IE	PUIE	—	—	—	—	—	—	—	IPUIE	RCCIE	CHRIE	VMRGNIE	HMRGNIE	CMDLVIE	CMDFULIE	CMDMPTIE	0000
G1IR	PUIF	—	—	—	—	—	—	—	IPUIF	RCCIF	CHRIF	VMRGNIF	HMRGNIF	CMDLVIF	CMDFULIF	CMDMPTIF	0000
G1W1ADRL	W1ADR<15:0>																0000
G1W1ADRH	—	—	—	—	—	—	—	—	W1ADR<23:16>								0000
G1W2ADRL	W2ADR<15:0>																0000
G1W2ADRH	—	—	—	—	—	—	—	—	W2ADR<23:16>								0000
G1PUW	—	—	—	—	—	PUW<10:0>											0000
G1PUH	—	—	—	—	—	PUH<10:0>											0000
G1DPADRL	DPADR<15:0>																0000
G1DPADRH	—	—	—	—	—	—	—	—	DPADR<23:16>								0000
G1DPW	—	—	—	—	—	DPW<10:0>											0000
G1DPH	—	—	—	—	—	DPH<10:0>											0000
G1DPWT	—	—	—	—	—	DPWT<10:0>											0000
G1DPHT	—	—	—	—	—	DPHT<10:0>											0000
G1CON2	DPGWDTH<1:0>		DPSTGER<1:0>		—	—	DPTEST<1:0>		DPBPP<2:0>			—	—	DPMODE<2:0>			0000
G1CON3	—	—	—	—	—	—	DPPINOE	DPPOWER	DPCLKPOL	DPENPOL	DPVSPOL	DPHSPOL	DPPWROE	DPENOE	DPVSOE	DPHSOE	0000
G1ACTDA	ACTLINE<7:0>								ACTPIX<7:0>								0000
G1HSYNC	HSLEN<7:0>								HSST<7:0>								0000
G1VSYNC	VSLEN<7:0>								VSST<7:0>								0000
G1DBLCON	VENST<7:0>								HENST<7:0>								0000
G1CLUT	CLUTEN	CLUTBUSY	—	—	—	—	CLUTTRD	CLUTRWEN	CLUTADR<7:0>								0000
G1CLUTWR	CLUTWR<15:0>																0000
G1CLUTRD	CLUTRD<15:0>																0000
G1MRGN	VBAMGN<7:0>								HBAMGN<7:0>								0000
G1CHRX	—	—	—	—	—	CURPOSX<10:0>											0000
G1CHRY	—	—	—	—	—	CURPOSY<10:0>											0000
G1IPU	—	—	—	—	—	—	—	—	—	—	HUFFERR	BLCKERR	LENERR	WRAPERR	IPUDONE	BFINAL	0000
G1DBEN	GDBEN<15:0>																0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

43.12 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the PIC24F device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Graphics Controller Module (GFX) of the PIC24F devices are:

Title	Application Note #
"How to Use Widgets in Microchip Graphics Library"	AN1136
"Fonts in the Microchip Graphics Library"	AN1182
"Using a Keyboard with the Microchip Graphics Library"	AN1227
"How to Create Widgets in Microchip Graphics Library"	AN1246

Note: Please visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the PIC24F family of devices.

43.13 REVISION HISTORY

Revision A (November 2009)

This is the initial released version of this document.