EPSON
EXCEED YOUR VISION

S1D13C00 Memory Display Controller

Hardware Functional Specification

Document Number: XB8A-A-001-01

Issue Date: 2019/10/16

SEIKO EPSON CORPORATION

Rev.1.2

NOTICE

Table of Contents

1. Introduction

1.1 Scope

This is the Hardware Functional Specification for the S1D13C00 Memory Display Controller (MDC). Included in this document are system connections, pin descriptions, timing diagrams, AC and DC characteristics, register descriptions, functional descriptions and power management descriptions. This document is intended for two audiences: Display Subsystem Designers and Software Developers.

1.2 Operational Overview

The S1D13C00 is a display controller for low-power memory displays used in wearable devices. It supports interfaces to 6-bit color memory-in-pixel (MIP) panels, 8-color or 1-bit black-and-white memory LCD panels (SPI interface), and 1/2/4/8 BPP grayscale LCD panels with 8-bit parallel or 3-/4-wire serial interface.

The S1D13C00 has 96Kbytes of RAM which can be used as a frame buffer and other functions. The display controller also includes a Drawing Engine and an Image/Bitmap Copying Engine with scaling and rotation or horizontal and vertical shear. A voltage booster circuit in the S1D13C00 provides two programmable supply voltages for powering the different panels supported. The S1D13C00 has built-in 32kHz and 20MHz internal oscillators which supply clocks needed for operations.

In addition to the display controller functions, the S1D13C00 also has other peripherals such as real-time clock/calendar (RTC), SPI, QSPI, I2C, DMA Controller, Sound Generator, and IR remote control transmitter.

The S1D13C00 has an Event Processor which has a simple instruction set for reading/writing memory and registers. For example, microcode can be written for the Event Processor to read the current time from the RTC and update the display with the current time value. The Host MCU does not need to wake up to perform this display update function.

The S1D13C00 is a flexible, low cost, low power multifunction solution that meets the display demands of wearable devices.

2. Features

2.1 Panels Supported

- 6-bit color MIP interface
- 3-bit (8-color) or 1-bit (black and white) Memory LCD with SPI interface
- 1/2/4/8BPP grayscale LCD panel with 8-bit parallel or 3-/4-wire serial interface
- 1/2/4/8BPP EPD (electrophoretic deposition) panel with 3-/4-wire serial interface
- 0, 90, 180, 270 degree rotation of frame buffer image to panel.
- Pixel formats: 6-bit color, 3-bit color, 1/2/4/8BPP grayscale with alpha channel option.

2.2 Memory

- 96Kbytes RAM

2.3 Voltage Booster/Regulator Supplies

- VMDL: 2.30V, 2.70V, 3.00V, 3.10V, 3.20V, 3.30V, 3.40V, or 3.60V selectable
- VMDH: 4.30V, 4.40V, 4.50V, 4.60V, 4.70V, 4.80V, 4.90V, or 5.00V selectable

2.4 Host Interfaces Supported

- SPI (Mode0 or Mode3)
- QSPI (Mode0 or Mode3)
- Indirect 8-bit

2.5 Peripherals

- Memory Display Controller (MDC) with drawing and image copying functions
- Event Processor with simple instruction set
- Real-time clock/calendar (RTC)
- SPI master/slave interface
- I2C master/slave interface
- QSPI master/slave interface with memory-mapped read access to external serial flash
- DMA controller
- 16-bit general-purpose timers
- General-purpose I/O pins

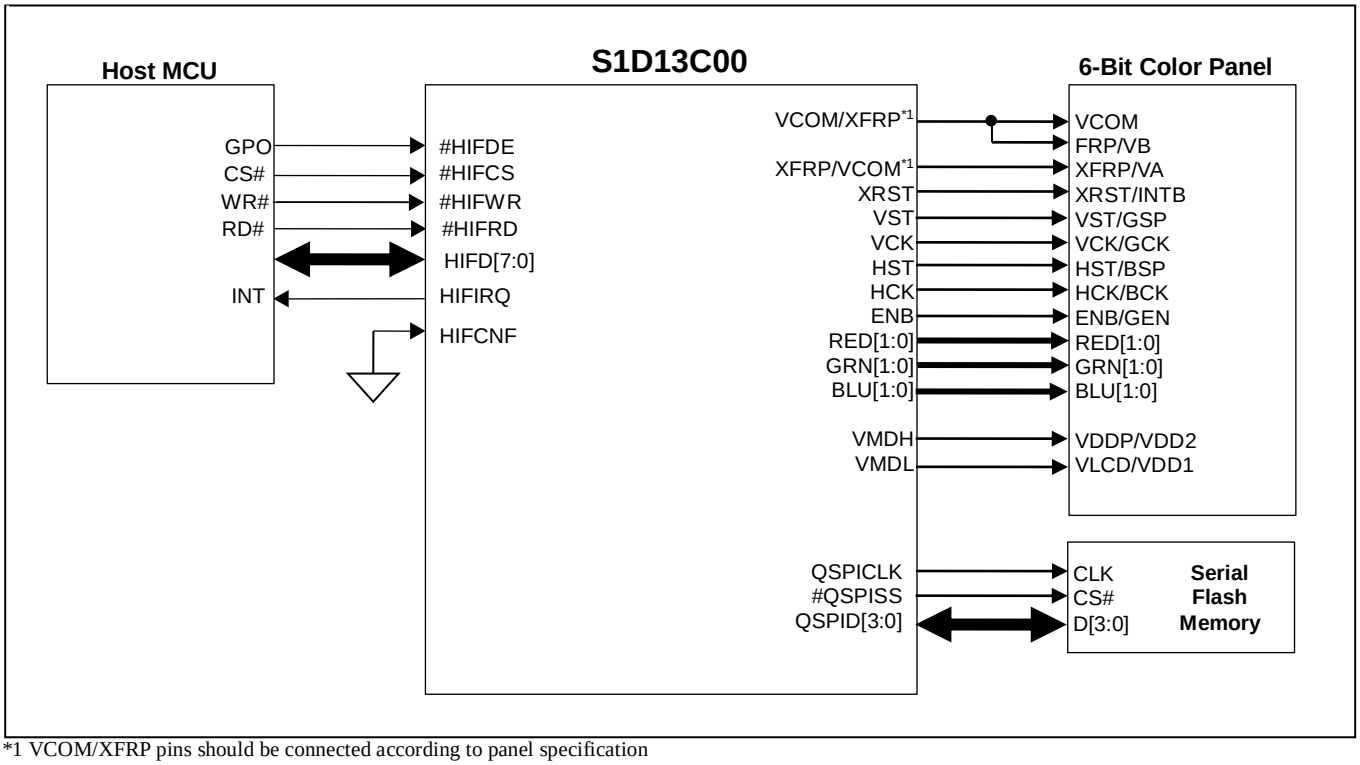
2.6 Miscellaneous

- 32kHz internal oscillator or 32.768kHz external crystal circuit
- 20MHz internal oscillator
- Input supply voltage: 1.8V to 5.5V (voltage booster off), 2.0V to 5.5V (voltage booster on)
- Operating Temperature: -40 to 85 ˚C
- Packages: TQFP13-64 (10x10mm, pin pitch: 0.5mm), WCSP-64 (3.45x3.45mm, ball pitch: 0.4mm)

3. System Overview

3.1 Typical System Connections

3.1.1 Indirect 8-bit Host Interface and 6-Bit Color Panel



*1 VCOM/XFRP pins should be connected according to panel specification

Figure 3.1 Indirect 8-bit Host Interface and 6-bit Color Panel

3.1.2 SPI Single-Data Host Interface and 1-Bit/3-Bit SPI Panel

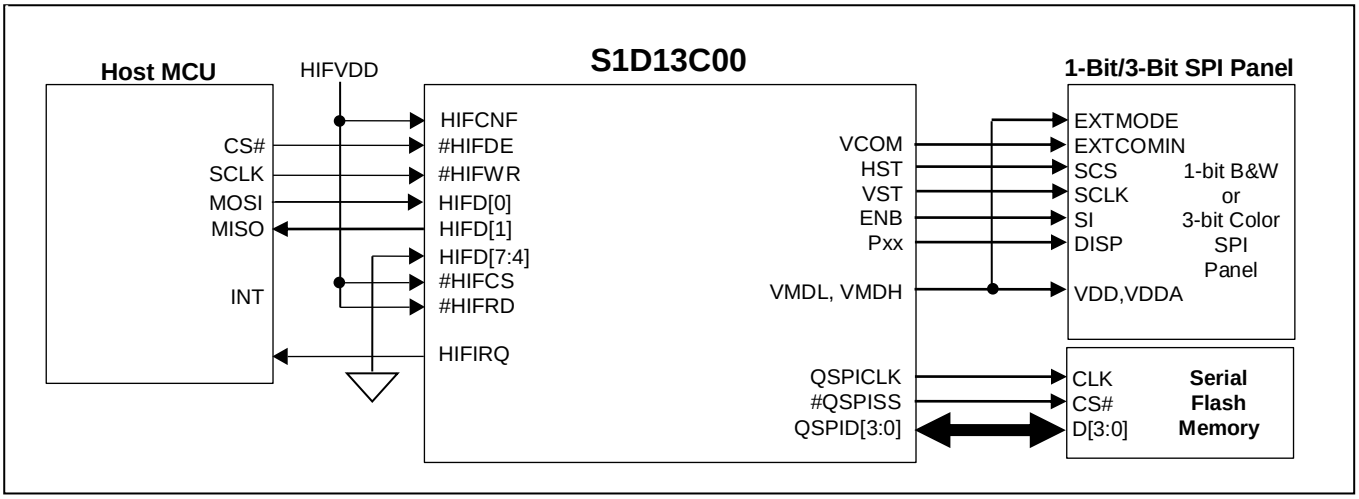


Figure 3.2 SPI Host Interface and 1-Bit/3-Bit Memory LCD Panel

4. Pins

4.1 Pinout Diagrams

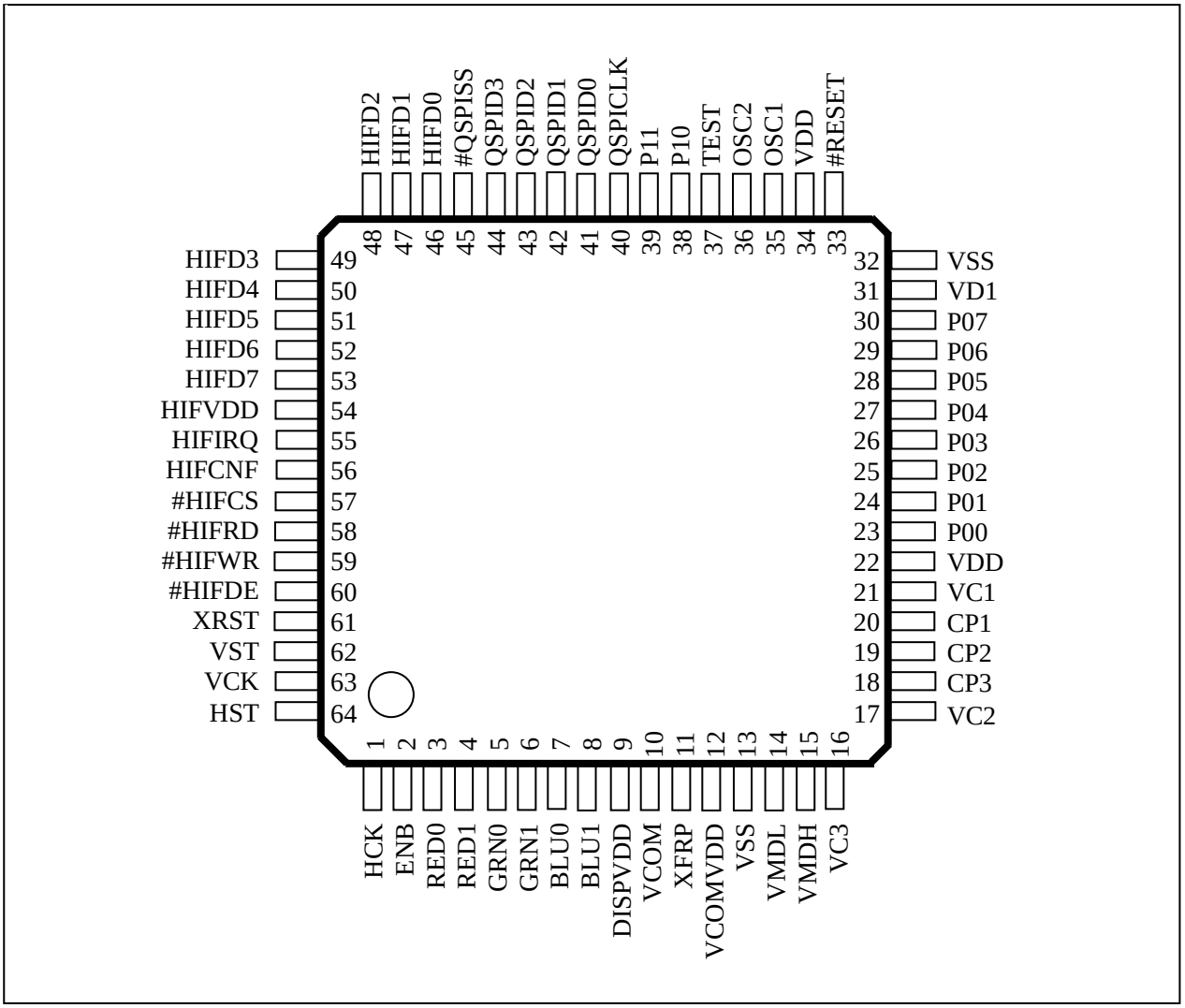
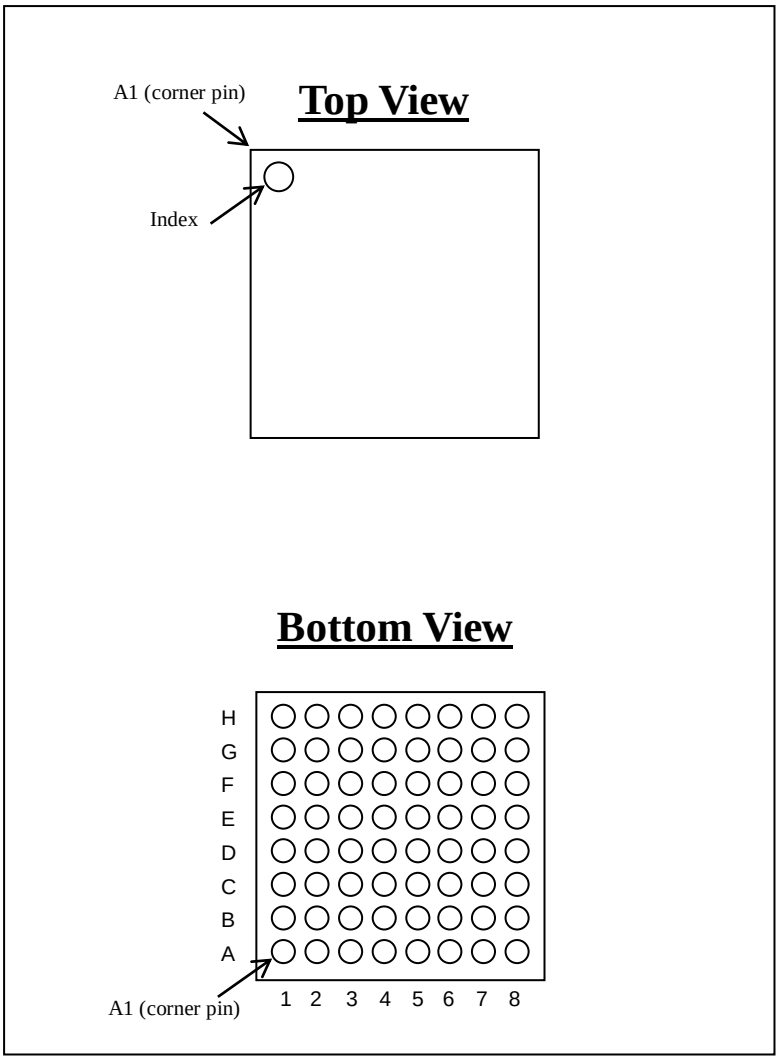


Figure 4.1 S1D13C00 TQFP13-64 Pinout Diagram - Top View



Pin	Name	Pin	Name
A1	HCK	E1	HIFIRQ
A2	RED0	E2	HIFD7
A3	GRN0	E3	HIFCNF
A4	BLU0	E4	#QSPISS
A5	DISPVDD	E5	TEST
A6	VCOMVDD	E6	P04
A7	VMDL	E7	P03
A8	VC3	E8	P02
B1	VCK	F1	HIFVDD
B2	HST	F2	HIFD6
B3	ENB	F3	HIFD4
B4	BLU1	F4	QSPID0
B5	VCOM	F5	QSPICLK
B6	VMDH	F6	P07
B7	VC2	F7	P06
B8	CP3	F8	P05
C1	#HIFDE	G1	HIFD5
C2	XRST	G2	HIFD2
C3	VST	G3	HIFD0
C4	GRN1	G4	QSPID2
C5	XFRP	G5	P10
C6	P00	G6	VDD
C7	CP2	G7	VSS
C8	CP1	G8	VD1
D1	#HIFRD	H1	HIFD3
D2	#HIFWR	H2	HIFD1
D3	#HIFCS	H3	QSPID3
D4	RED1	H4	QSPID1
D5	VSS	H5	P11
D6	P01	H6	OSC2
D7	VC1	H7	OSC1
D8	VDD	H8	#RESET

Figure 4.2 S1D13C00 WCSP-64 Pinout Diagram

4.2 Pin Descriptions

Key:

Pin Types

I	=	Input
O	=	Output
IO	=	Bi-Directional (Input/Output)
P	=	Power pin
AP	=	Analog Power pin

Initial State

I (Pull-up)	=	Input with pull-up
I (Pull-down)	=	Input with pull-down
O (H)	=	High level output
O (L)	=	Low level output
Hi-Z	=	High Impedance

4.2.1 Host Interface

Table 4.1 Host Interface Pin Descriptions

Pin Name	Type	TQFP13-64 Pin#	WCSP-64 Pin#	Power	Initial State	Description
HIFCNF	I	56	E3	HIFVDD	—	Host Interface Configuration: 0 = INDIRECT 8-BIT 1 = SPI/QSPI
#HIFCS	I	57	D3	HIFVDD	—	Active-low chip-select. For INDIRECT 8-BIT.
#HIFRD	I	58	D1	HIFVDD	—	Active-low read signal for INDIRECT 8-BIT.
#HIFWR	I	59	D2	HIFVDD	—	Active-low write signal for INDIRECT 8-BIT. HSPICLK clock for SPI/QSPI interface.
#HIFDE	I	60	C1	HIFVDD	—	Active-low #HIFDE signal for INDIRECT 8-BIT. #HSPISS for SPI/QSPI interface.
HIFD0	IO	46	G3	HIFVDD	—	Bidirectional D[0] for INDIRECT 8-BIT. HSPIDI input for SPI Single protocol. Bidirectional HSPID[0] for Extended Dual, Extended Quad, Dual, and Quad protocol.
HIFD1	IO	47	H2	HIFVDD	—	Bidirectional D[1] for INDIRECT 8-BIT. HSPIDO output for SPI Single protocol. Bidirectional HSPID[1] for Extended Dual, Extended Quad, Dual, and Quad protocol.
HIFD2	IO	48	G2	HIFVDD	—	Bidirectional D[2] for INDIRECT 8-BIT. Bidirectional HSPID[2] for Extended Quad and Quad protocol.
HIFD3	IO	49	H1	HIFVDD	—	Bidirectional D[3] for INDIRECT 8-BIT. Bidirectional HSPID[3] for Extended Quad and Quad protocol.
HIFD4	IO	50	F3	HIFVDD	—	Bidirectional D[4] for INDIRECT 8-BIT. HSPISEL0 configuration input for dual/quad. 0 = dual protocol 1 = quad protocol
HIFD5	IO	51	G1	HIFVDD	—	Bidirectional D[5] for INDIRECT 8-BIT. HSPISEL1 configuration input for SPI/QSPI. 0 = SPI 1 = QSPI (dual or quad protocol)
HIFD6	IO	52	F2	HIFVDD	—	Bidirectional D[6] for INDIRECT 8-BIT.
HIFD7	IO	53	E2	HIFVDD	—	Bidirectional D[7] for INDIRECT 8-BIT.
HIFIRQ	O	55	E1	HIFVDD	—	Interrupt output to Host MCU.
HIFVDD	P	54	F1	—	—	Supply input for Host Interface signals.

4.2.2 Panel Interface

Table 4.2 Panel Interface Pin Descriptions

Pin Name	Type	TQFP13-64 Pin#	WCSP-64 Pin#	Power	Initial State	Description
VCOM	O	10	B5	VCOMVDD	O (L)	FRP/VCOM or VA for 6-bit color panel. EXTCOMIN for 1-bit/3-bit SPI panel.
XFRP	O	11	C5	VCOMVDD	O (L)	XFRP or VB for 6-bit color panel.
XRST	O	61	C2	DISPVDD	O (L)	XRST or INTB for 6-bit color panel. A0 for grayscale panels.
VST	O	62	C3	DISPVDD	O (L)	VST or GSP for 6-bit color panel SCLK for 1-bit/3-bit SPI panel XRD for 8-bit Parallel grayscale panel SCL for 3-/4-Wire Serial grayscale panel SCL for 3-/4-Wire EPD serial interfaces
VCK	IO	63	B1	DISPVDD	O (L)	VCK or GCK output for 6-bit color panel DOUT[0] for 8-bit Parallel grayscale panel D/C for EPD 4-Wire serial interface
HST	O	64	B2	DISPVDD	O (L)	HST or BSP for 6-bit color panel SCS for 1-bit/3-bit SPI panel XCS for 8-bit Parallel or 3-/4-Wire Serial grayscale panel CSB for 3-/4-Wire EPD serial interfaces
HCK	O	1	A1	DISPVDD	O (L)	HCK or BCK for 6-bit color panel DOUT[1] for 8-bit Parallel grayscale panel
ENB	IO	2	B3	DISPVDD	O (L)	ENB or GEN output for 6-bit color panel SDI for 1-bit/3-bit SPI panel XWR for 8-bit Parallel grayscale panel SD for 3-/4-Wire Serial grayscale panel SDA for EPD serial interfaces
RED0	O	3	A2	DISPVDD	O (L)	RED0 for 6-bit color panel DOUT[2] for 8-bit Parallel grayscale panel
RED1	O	4	D4	DISPVDD	O (L)	RED1 for 6-bit color panel DOUT[3] for 8-bit Parallel grayscale panel
GRN0	O	5	A3	DISPVDD	O (L)	GRN0 for 6-bit color panel DOUT[4] for 8-bit Parallel grayscale panel
GRN1	O	6	C4	DISPVDD	O (L)	GRN1 for 6-bit color panel DOUT[5] for 8-bit Parallel grayscale panel
BLU0	O	7	A4	DISPVDD	O (L)	BLU0 for 6-bit color panel DOUT[6] for 8-bit Parallel grayscale panel
BLU1	O	8	B4	DISPVDD	O (L)	BLU1 for 6-bit color panel DOUT[7] for 8-bit Parallel grayscale panel
DISPVDD	P	9	A5	—	—	Supply input for panel interface signals.
VCOMVDD	P	12	A6	—	—	Supply input for VCOM and XFRP signals.

4.2.3 Voltage Booster/Regulator

Table 4.3 Voltage Booster Pin Descriptions

Pin Name	Type	TQFP13-64 Pin#	WCSP-64 Pin#	Power	Initial State	Description
CP1	P	20	C8	—	—	Voltage booster charge-pump capacitor connection.
CP2	P	19	C7	—	—	Voltage booster charge-pump capacitor connection.
CP3	P	18	B8	—	—	Voltage booster charge-pump capacitor connection.
VC1	P	21	D7	—	—	Voltage booster output capacitor connection.
VC2	P	17	B7	—	—	Voltage booster output capacitor connection.
VC3	P	16	A8	—	—	Voltage booster output capacitor connection.
VMDL	P	14	A7	—	—	Lower voltage booster output supply.
VMDH	P	15	B6	—	—	Higher voltage booster output supply.

4.2.4 QSPI Interface

Table 4.4 QSPI Interface Pin Descriptions

Pin Name	Type	TQFP13-64 Pin#	WCSP-64 Pin#	Power	Initial State	Description
QSPICLK	IO	40	F5	VDD	Hi-Z	QSPI clock.
#QSPISS	IO	45	E4	VDD	Hi-Z	QSPI active-low chip-select.
QSPID0	IO	41	F4	VDD	O(L)	QSPI D0 data.
QSPID1	IO	42	H4	VDD	Hi-Z	QSPI D1 data.
QSPID2	IO	43	G4	VDD	Hi-Z	QSPI D2 data.
QSPID3	IO	44	H3	VDD	Hi-Z	QSPI D3 data.

4.2.5 Clock Input

Table 4.5 Clock Input Pin Descriptions

Pin Name	Type	QFP13-64 Pin#	WCSP-64 Pin#	Power	Initial State	Description
OSC1	I	35	H7	VDD	—	32.768kHz crystal oscillator input.
OSC2	O	36	H6	VDD	—	32.768kHz crystal oscillator output.

4.2.6 Miscellaneous

Table 4.6 Miscellaneous Pin Descriptions

Pin Name	Type	TQFP13-64 Pin#	WCSP-64 Pin#	Power	Initial State	Description
TEST	I	37	E5	VDD	—	Test Enable input used for production test only. This pin must be connected directly to VSS for normal operation.
#RESET	I	33	H8	VDD	—	Active-low reset input.

4.2.7 General Purpose Input/Output (GPIO)

Table 4.7 GPIO Pin Descriptions

Pin Name	Type	TQFP13-64 Pin#	WCSP-64 Pin#	Power	Initial State	Description
P00	IO	23	C6	VDD	Hi-Z	General purpose input/output pin P00. #SPISS input for SPI.
P01	IO	24	D6	VDD	Hi-Z	General purpose input/output pin P01. SPICLK input/output clock for SPI.
P02	IO	25	E8	VDD	Hi-Z	General purpose input/output pin P02. SPIDI input for SPI.
P03	IO	26	E7	VDD	Hi-Z	General purpose input/output pin P03. SPIDO output for SPI.
P04	IO	27	E6	VDD	Hi-Z	General purpose input/output pin P04. I2CSCL input/output for I2C.
P05	IO	28	F8	VDD	Hi-Z	General purpose input/output pin P05. I2CSDA input/output for I2C.
P06	IO	29	F7	VDD	Hi-Z	General purpose input/output pin P06. REMO output for REMC3.
P07	IO	30	F6	VDD	Hi-Z	General purpose input/output pin P07. CLPLS output for REMC3.
P10	IO	38	G5	VDD	O (L)	General purpose input/output pin P10. BZOUT output for SNDA. RTC1S output for RTC.
P11	IO	39	H5	VDD	O (L)	General purpose input/output pin P11. #BZOUT output for SNDA. FOUT output for CLG.

4.2.8 Power and Ground

Table 4.8 Power and Ground Pin Descriptions

Pin Name	Type	TQFP13-64 Pin#	WCSP-64 Pin#	Power	Description
VDD	P	22, 34	D8, G6	VDD	Main input supply voltage
VD1	P	31	G8	—	VD1 regulator output.
VSS	P	13, 32	D5,G7	—	VSS ground.

4.3 Host Interface Configuration Pins and Connections

Table 4.9 Host Interface Configuration Pin Descriptions

Pin Name	HIFCNF = 1	HIFCNF = 0
	SPI/QSPI interface	INDIRECT 8-BIT interface
HIFD[5:4]	0xb = SPI Single/Extended protocol 10b = Dual protocol 11b = Quad protocol	-

Table 4.10 Host Interface Connections

Pin Name	Description	INDIRECT 8-Bit	Single or Extended	Dual	Quad
HIFCNF	Host Configuration	0	1		
#HIFDE	Device Enable/Serial Select	#HIFDE	#HSPISS		
#HIFCS	Chip-Select	#HIFCS	1		
#HIFWR	Write/SPI clock	#HIFWR	HSPICLK		
#HIFRD	Read	#HIFRD	1		
HIFIRQ	Interrupt output to Host	HIFIRQ			
HIFD0	HIFD[0], HSPIDI, HSPID[0]	HIFD[0]	HSPIDI	HSPID[0]	HSPID[0]
HIFD1	HIFD[1], HSPIDO, HSPID[1]	HIFD[1]	HSPIDO	HSPID[1]	HSPID[1]
HIFD2	HIFD[2], HSPID[2]	HIFD[2]	*A	0 or 1	HSPID[2]
HIFD3	HIFD[3], HSPID[3]	HIFD[3]	*A	0 or 1	HSPID[3]
HIFD4	HIFD[4], HSPISEL0	HIFD[4]	0 or 1	0	1
HIFD5	HIFD[5], HSPISEL1	HIFD[5]	0	1	1
HIFD6	HIFD[6]	HIFD[6]	0 or 1		
HIFD7	HIFD[7]	HIFD[7]	0 or 1		

*A For Single and Extended Dual protocol, HIFD[3:2] are not used and must be terminated to 0 or 1.
For Extended Quad protocol, HIFD[3:2] are used as bidirectional data. See Section 11.2 for details of the SPI protocols supported

4.4 GPIO Pins Mapping

Table 4.11 GPIO Pins Mapping

PxSEL[y] = 0	PxSEL[y] = 1			
	PxyMUX[1:0]			
GPIO (Pxy)	**0**	**1**	**2**	**3**
P00	#SPISS	—	—	—
P01	SPICLK	—	—	—
P02	SPIDI	—	—	—
P03	SPIDO	—	—	—
P04	I2CSCL	—	—	—
P05	I2CSDA	—	—	—
P06	REMO	—	—	—
P07	CLPLS	—	—	—
P10	BZOUT	RTC1S	—	—
P11	#BZOUT	FOUT	—	—

4.5 Basic External Connection Diagram

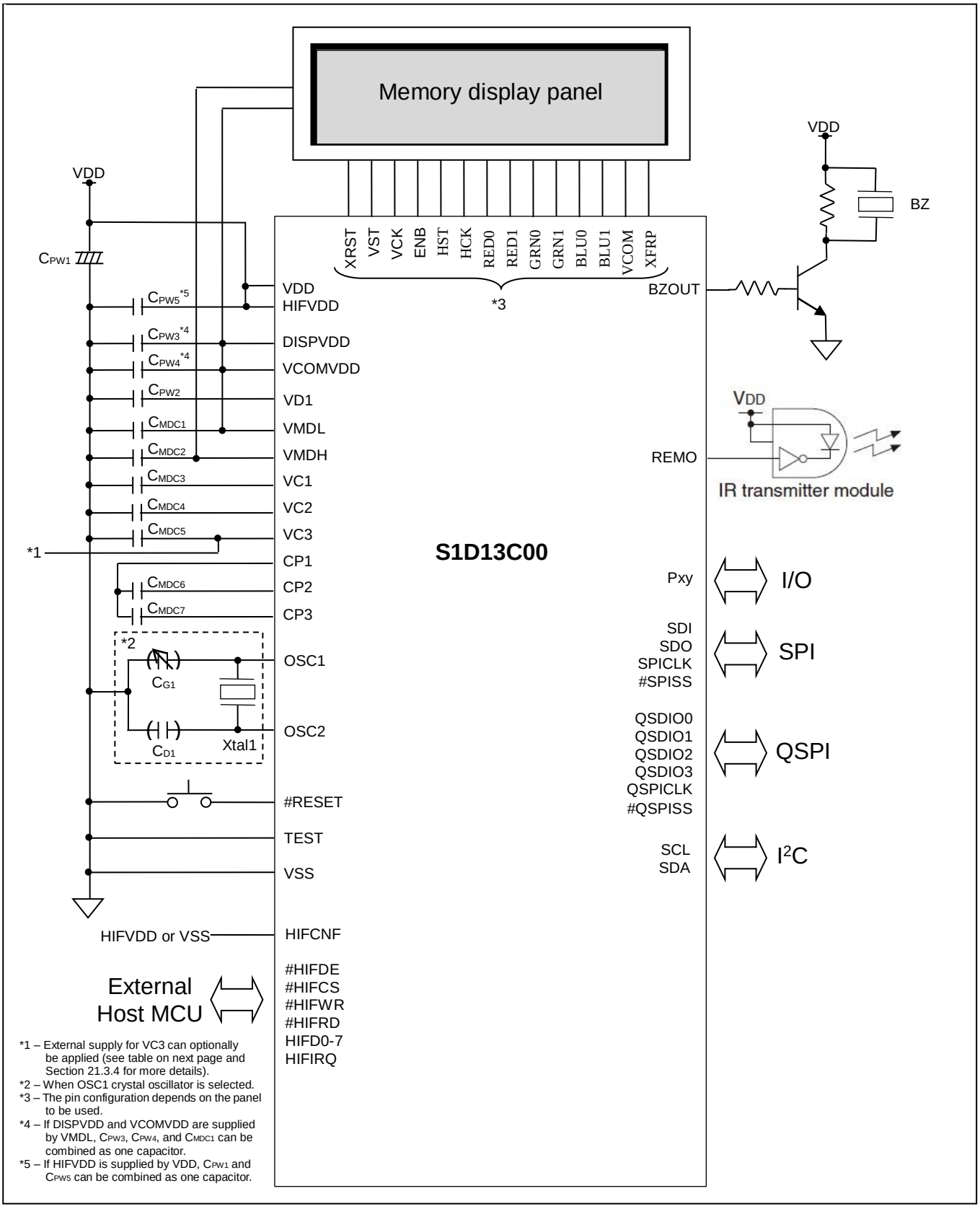


Figure 4.3 Basic External Connection Diagram

Table 4.12 Sample External Components

Symbol	Name	Recommend components
Xtal1	32.768kHz crystal	C-002RX (R_1=50kΩ (Max.), C_L = 7pF) manufactured by Seiko Epson Corp.
C_{G1}	OSC1 gate capacitor	Trimmer capacitor or ceramic capacitor
C_{D1}	OSC1 drain capacitor	Ceramic capacitor
C_{PW1}	Bypass capacitor between VSS and VDD	Ceramic capacitor or electrolytic capacitor
C_{PW2}	Capacitor between VSS and VD1	Ceramic capacitor
C_{PW3}	Capacitor between VSS and DISPVDD	Ceramic capacitor. Can be combined with C_{MDC1} if DISPVDD is supplied by VMDL.
C_{PW4}	Capacitor between VSS and VCOMVDD	Ceramic capacitor. Can be combined with C_{MDC1} if VCOMVDD is supplied by VMDL.
C_{PW5}	Capacitor between VSS and HIFVDD	Ceramic capacitor. Can be combined with C_{PW1} if HIFVDD is supplied by VDD.
C_{MDC1}	Capacitor between VSS and VMDL	Ceramic capacitor
C_{MDC2}	Capacitor between VSS and VMDH	Ceramic capacitor
C_{MDC3}	Capacitor between VSS and VC1	Ceramic capacitor. Not needed if external VC3 is applied.
C_{MDC4}	Capacitor between VSS and VC2	Ceramic capacitor. Not needed if external VC3 is applied.
C_{MDC5}	Capacitor between VSS and VC3	Ceramic capacitor.
C_{MDC6}	Capacitor between CP1 and CP2	Ceramic capacitor. Not needed if external VC3 is applied.
C_{MDC7}	Capacitor between CP1 and CP3	Ceramic capacitor. Not needed if external VC3 is applied.
BZ	Piezoelectric buzzer	PS1240P02 manufactured by TDK Corp.

5. Logic Diagram

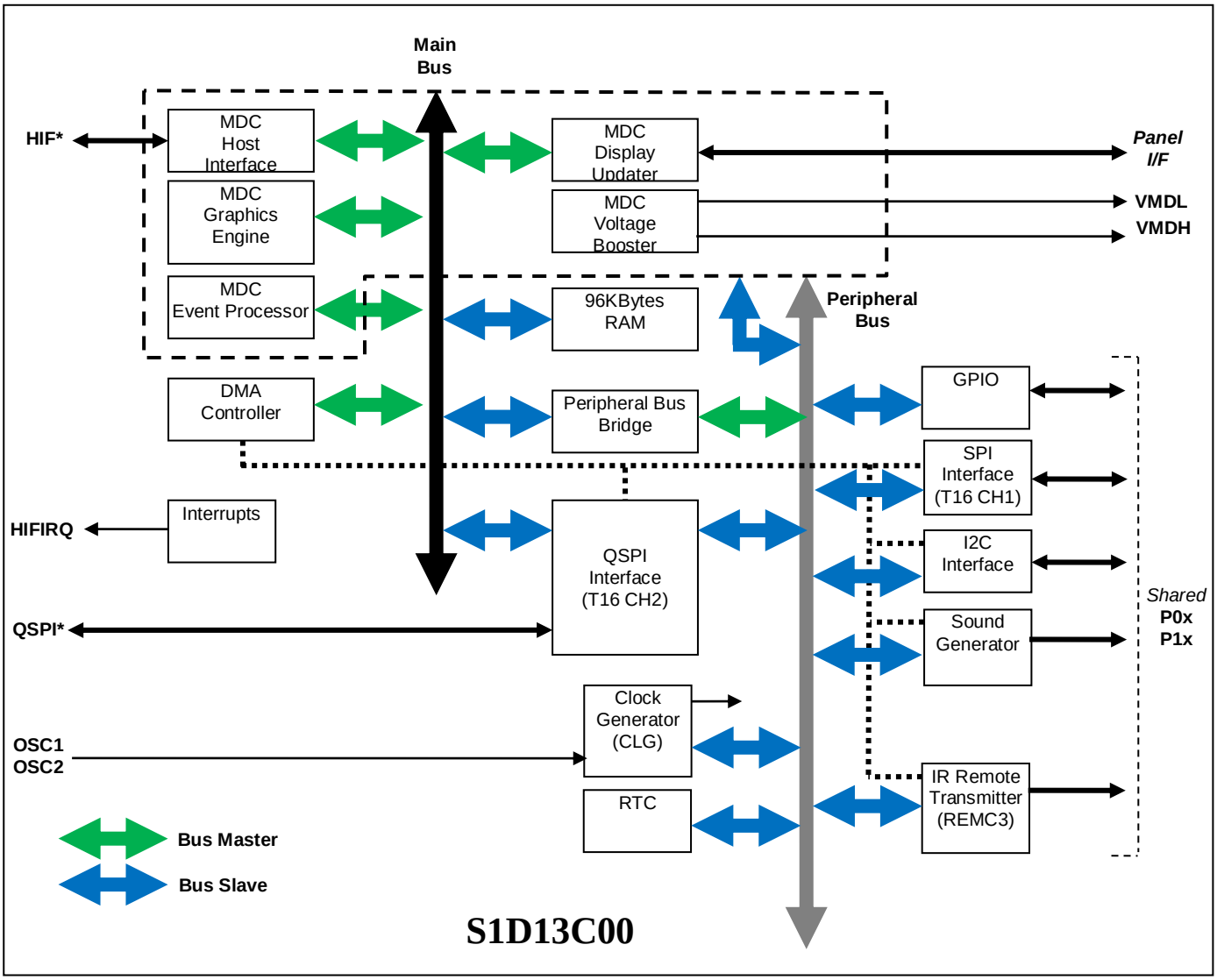


Figure 5.1 Internal Block Diagram

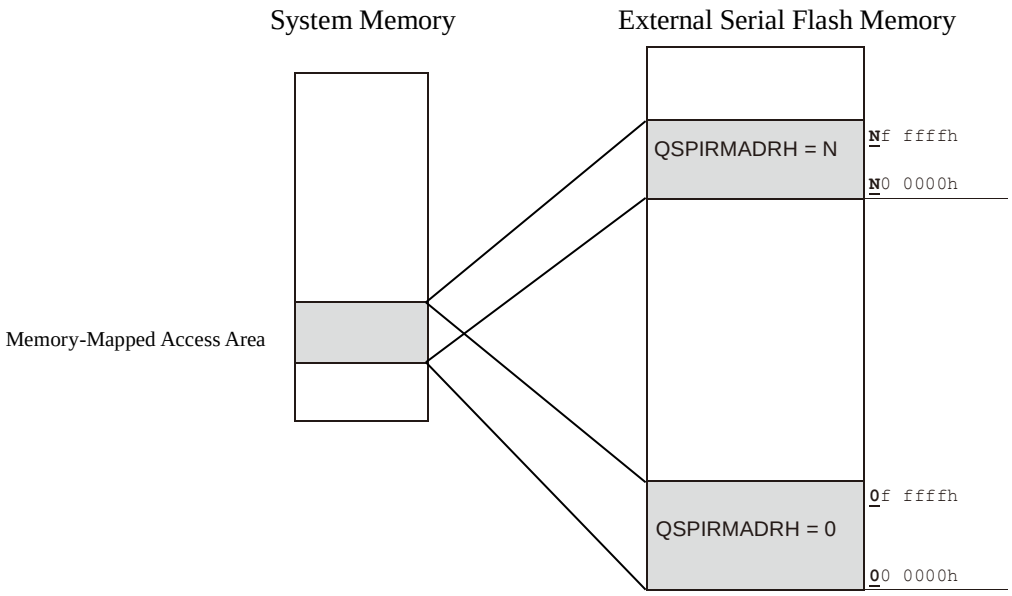
6. Memory Map

Table 6.1 Memory Map

FFFF_FFFFh 4000_4000h	Reserved
4000_3FFFh 4000_0000h	*__Registers__ area (16K bytes)
3FFF_FFFFh 2001_8000h	Reserved
2001_7FFFh 2000_0000h	**RAM** area (96K bytes)
1FFF_FFFFh 0014_0000h	Reserved
0013_FFFFh 0004_0000h	**__Memory-Mapped Access__ area for external serial Flash memory (1Mbyte page)
0003_FFFFh 0000_0000h	Reserved

*For details on the Registers area, see Section 10.

**The QSPI interface module can be programmed to provide a 1Mbyte page at 0004_0000h to 0013_FFFFh for the system to read the contents of a serial flash device connected to the QSPI interface. The QSPIRMADRH register specifies the upper 12 bits of the address for accessing a 1Mbyte (20-bit) page of the serial flash.



7. Clocks, Reset and Operating States

7.1 Clock Generator (CLG)

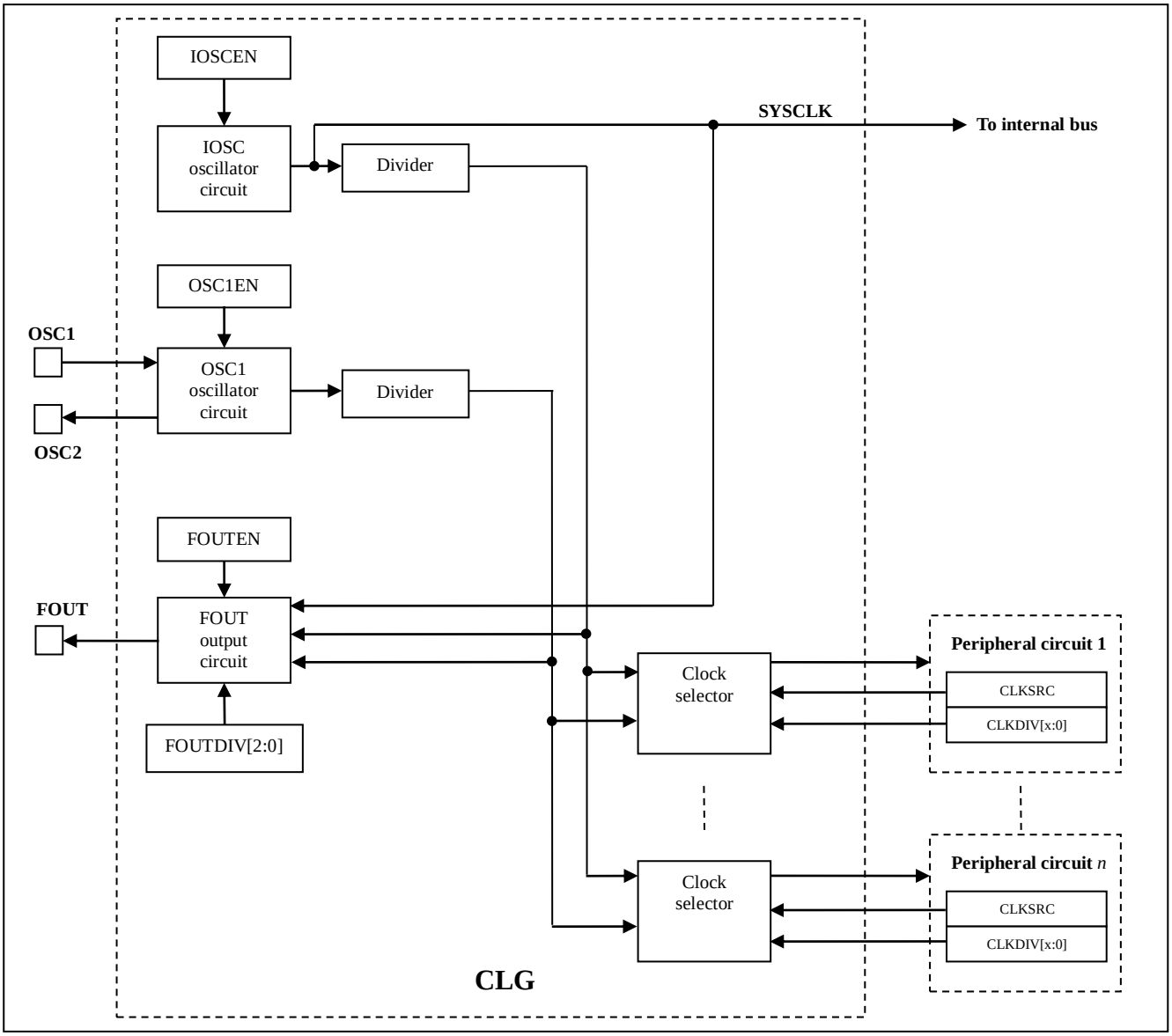


Figure 7.1 Clock Generator (CLG) Block Diagram

There are two clock sources for the system: IOSC and OSC1. The IOSC and OSC1 oscillator circuits are both off on reset. The IOSC oscillator frequency is selectable between 20MHz, 16MHz, 12MHz, and 8MHz. The OSC1 oscillator output is selectable between an internal 32kHz oscillator and an external 32.768kHz crystal circuit connected to the OSC1 and OSC2 pins.

The internal system bus and most of the registers for peripherals are clocked by SYSCLK. The clock source for SYSCLK is the IOSC oscillator. Before accessing synchronous registers and RAM, the Host MCU needs to turn on the IOSC oscillator which is controlled by asynchronous SYSCTRL register.

All the peripherals, except for the Real-Time Clock (RTC), have the option to select the clock source between IOSC and OSC1 (CLKSRC bit), and the option to select a divided-down version of the clock source (CLKDIV[x:0] bits). The RTC clock source is fixed to OSC1.

The FOUT pin is multiplexed with the P11 GPIO pin to provide a selectable clock output to monitor.

7.1.1 IOSC Oscillator

The IOSC oscillator circuit features a fast startup and no external parts are required for oscillating.
Figure 7.2 shows the IOSC oscillator circuit.

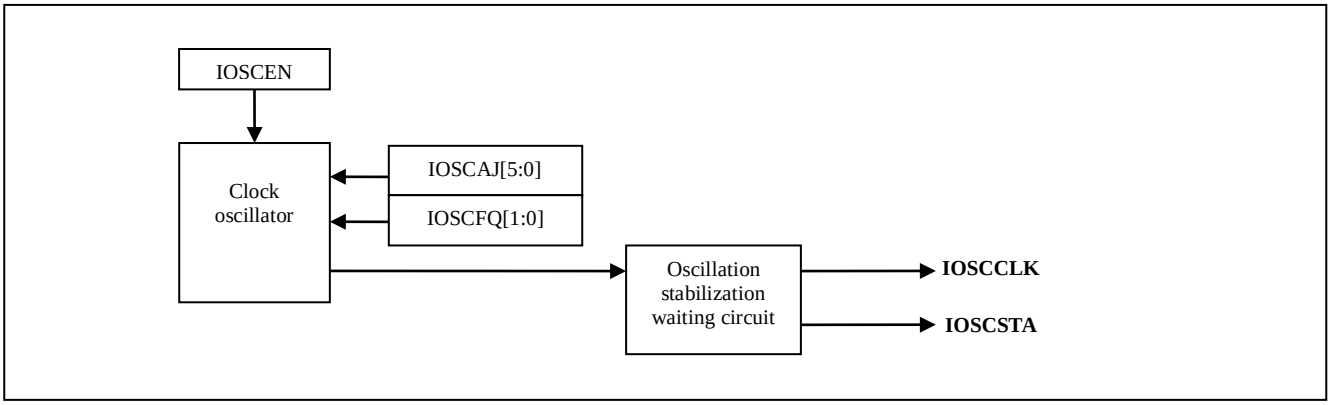


Figure 7.2 IOSC Oscillator Circuit

IOSCCLK is the IOSC clock output and IOSCSTA is a status bit (readable from the SYSCTL register) which indicates that the IOSC clock is stable. When the Host MCU turns on IOSC by setting IOSCEN bit to 1, it needs to wait for IOSCSTA to go high before accessing RAM and synchronous peripheral registers.

IOSCFQ[1:0] selects the frequency of IOSC (8/12/16/20MHz, default 8MHz) and IOSCAJ[5:0] (default value is 0) is used for manual trimming the IOSC output frequency if needed.

Follow the procedure shown below to start the IOSC clock:

1. Configure SYSCTL.IOSCFQ[1:0] to select the desired frequency.
2. Turn on IOSC by setting SYSCTL.IOSCEN to 1.
3. Wait for SYSCTL.IOSCSTA bit to go high.

When the Host MCU has finished reading or writing RAM and peripheral registers, it can turn off IOSC to reduce power consumption by setting SYSCTL.IOSCEN bit to 0.

7.1.2 OSC1 Oscillator

The OSC1 oscillator circuit is a low-power oscillator circuit that allows software to select the oscillator type from two different sources (internal and external) as shown in Figure 7.3.

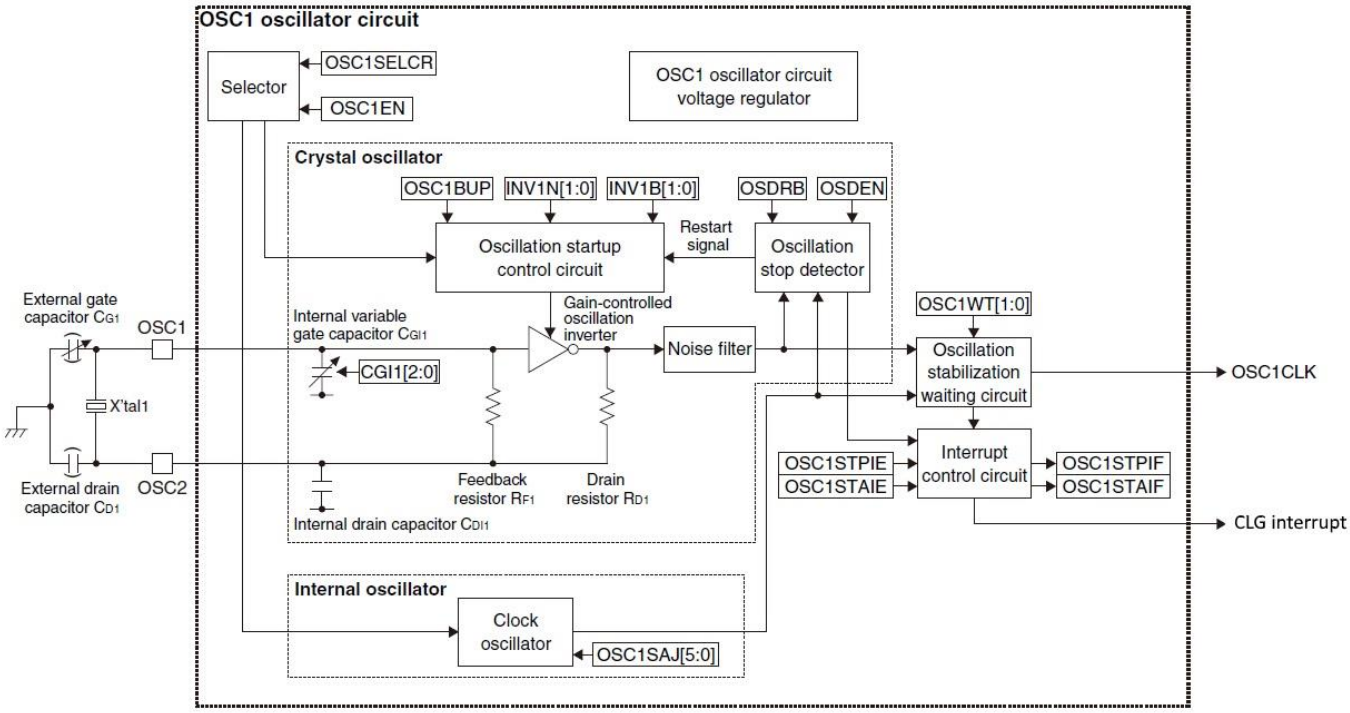


Figure 7.3 OSC1 Oscillator Circuit

Crystal Oscillator

This oscillator circuit includes a gain-controlled oscillation inverter and a variable gate capacitor allowing use of various crystal resonators (32.768 kHz typ.) with ranges from cylinder type through surface-mount type. The oscillator circuit also includes a feedback resistor and a drain resistor, so no external parts are required except for a crystal resonator. The embedded oscillation stop detector, which detects oscillation stop and restarts the oscillator, allows the system to operate in safety under adverse environments that may stop the oscillation. The oscillation startup control circuit operates for a set period of time after the oscillation is enabled to assist the oscillator in initiating, this makes it possible to use a low-power resonator that is difficult to start up.

Note: Depending on the circuit board or the crystal resonator type used, an external gate capacitor CG1 and a drain capacitor CD1 may be required.

Internal Oscillator

This 32 kHz oscillator circuit operates without any external parts. When the internal oscillator circuit is used, set the OSC1 pin level to VSS and leave the OSC2 pin open.

Oscillation start procedure for the OSC1 oscillator circuit

Follow the procedure shown below to start oscillation of the OSC1 oscillator circuit.

1. Write 1 to the CLGINTF.OSC1STAIF bit. (Clear interrupt flag)
2. Write 1 to the CLGINTE.OSC1STAIE bit. (Enable interrupt)
3. Write 0x0096 to the SYSPROT.PROT[15:0] bits. (Remove system protection)
4. Configure the following CLGOSC1 register bits:

- CLGOSC1.OSC1SELCR bit (Select oscillator type)
- CLGOSC1.OSC1WT[1:0] bits (Set oscillation stabilization waiting time)

In addition to the above, configure the following bits when using the crystal oscillator:

- CLGOSC1.INV1N[1:0] bits (Set oscillation inverter gain)
- CLGOSC1.CGI1[2:0] bits (Set internal gate capacitor)
- CLGOSC1.INV1B[1:0] bits (Set oscillation inverter gain for startup boosting period)
- CLGOSC1.OSC1BUP bit (Enable/disable oscillation startup control circuit)

5. When using the internal oscillator, set the CLGTRIM2.OSC1SAJ[5:0] bits as necessary.
   (Finely adjust oscillation frequency)
6. Write a value other than 0x0096 to the SYSPROT.PROT[15:0] bits. (Set system protection)
7. Write 1 to the CLGOSC.OSC1EN bit. (Start oscillation)
8. OSC1CLK can be used if the CLGINTF.OSC1STAIF bit = 1 after an interrupt occurs.

The setting values of the CLGOSC1.INV1N[1:0], CLGOSC1.CGI1[2:0], CLGOSC1.OSC1WT[1:0], CLGOSC1.INV1B[1:0], and CLGTRIM2.OSC1SAJ[5:0] bits should be determined after performing evaluation using the populated circuit board.

Note: Make sure the CLGOSC.OSC1EN bit is set to 0 (while the OSC1 oscillation is halted) when setting the CLGTRIM2.OSC1SAJ[5:0] bits.

7.1.3 FOUT Pin

The FOUT pin can output the clock generated by a clock source or its divided clock to outside the IC. This allows monitoring the oscillation frequency of the oscillator circuit or supplying an operating clock to external ICs. Follow the procedure shown below to start clock external output:

1. Assign the FOUT function to GPIO port P11.
2. Configure the following CLGFOUT register bits:
   - CLGFOUT.FOUTSRC[1:0] bits (Select clock source)
   - CLGFOUT.FOUTDIV[2:0] bits (Set clock division ratio)
   - Set the CLGFOUT.FOUTEN bit to 1. (Enable clock external output)

7.1.4 Interrupts

CLG has a function to generate the interrupts shown in Table 7.1.

Table 7.1 CLG Interrupt Function

Interrupt	Interrupt flag	Set condition	Clear condition
OSC1 oscillation stabilization waiting completion	CLGINTF.OSC1STAIF	When the OSC1 oscillation stabilization waiting operation has completed after the oscillation starts	Writing 1
OSC1 oscillation stop	CLGINTF.OSC1STPIF	When OSC1CLK is stopped, or when the CLGOSC.OSC1EN or CLGOSC1.OSDEN bit setting is altered from 1 to 0.	Writing 1

CLG provides interrupt enable (..IE) bits corresponding to each interrupt flag. An interrupt request is sent to the host MCU only when the interrupt flag, of which interrupt has been enabled by the interrupt enable bit, is set.

Figure 7.4 shows a diagram of the CLGINT interrupt signal. The CLGINT signal is logical ORed with interrupt signals from other peripherals to generate the HIFIRQ interrupt signal to the host MCU.
See Section 22 (“Interrupts”) for more details on the HIFIRQ output.

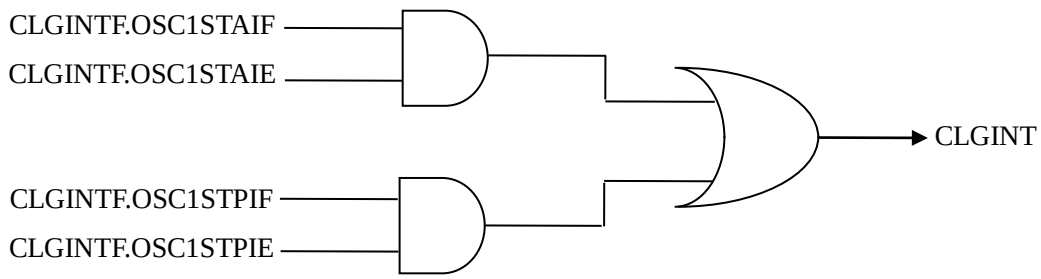


Figure 7.4 CLGINT Interrupt Circuit

7.1.5 Control Registers

See Section 10.2 for descriptions of the control registers for CLG.

7.2 System Resets

The S1D13C00 has three reset sources which resets the registers and internal circuits:

- Power-on Reset (POR)
- Hardware reset pin, #RESET
- Software reset from the Host with SYSCTRL bit 15 (SOFTRST)

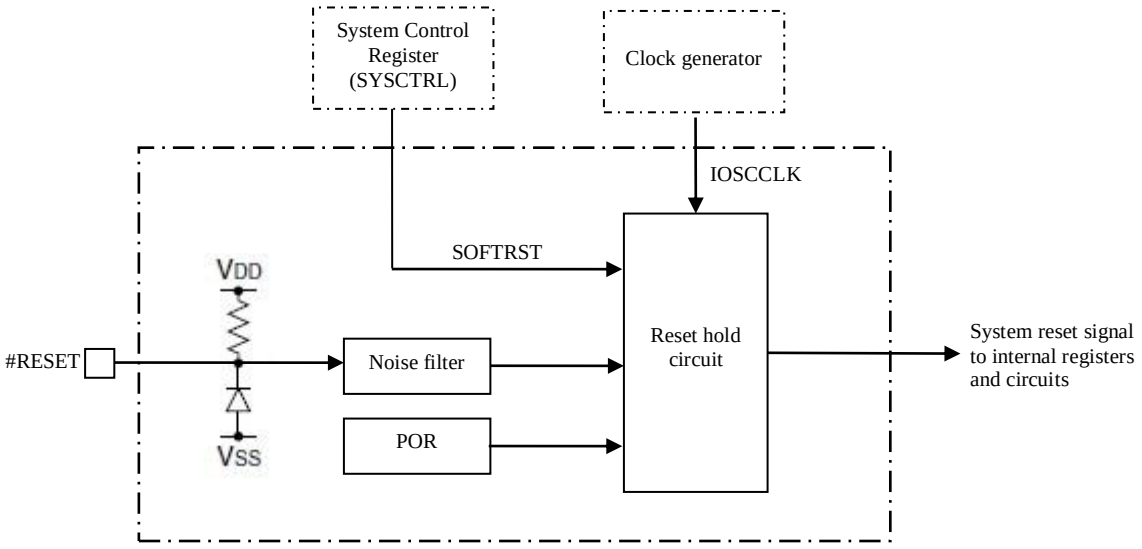


Figure 7.5 System Reset Circuit

7.2.1 #RESET Pin

Inputting a reset signal with a certain low level period to the #RESET pin issues a reset request. There is a noise filter which filters out noise or glitches on the #RESET pin.

7.2.2 Software Reset from Host

The Host MCU can issue a reset by writing 1 to the SOFTRST bit (15) of the asynchronous SYSCTRL register. A short pulse on SOFTRST is generated to trigger the reset hold circuit and reset the system. The SOFTRST bit is self-clearing.

7.2.3 POR – Power On Reset

POR (Power On Reset) issues a reset request when the rise of VDD is detected. Reset requests from this circuit ensure that the system will be reset properly when the power is turned on. Shows an example of POR internal reset operation according to variations in VDD.

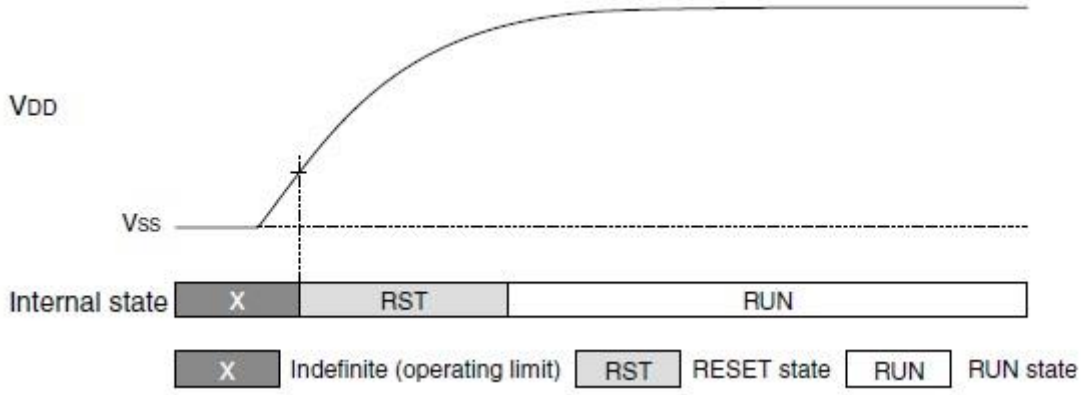


Figure 7.6 Example of Internal Reset by POR

For the #RESET and POR electrical specification, refer to Section 8.4 – Reset Characteristics.

7.3 Operating States

Two operating states are defined for the S1D13C00: IDLE or ACTIVE. In IDLE state, the IOSC is off and the chip is not doing anything. In ACTIVE state (default state after reset), the IOSC is on and the host MCU can access all the registers and RAM to perform display tasks. After display tasks are completed, the chip can go back into IDLE state to minimize power consumption.

After a hardware reset (#RESET pin) or a software reset (writing 1 to SOFTRST bit of SYSCTRL register), the IOSC oscillator is on (ACTIVE state) with a frequency of 8MHz (SYSCTRL.IOSCFQ[1:0] = 00b) and the OSC1 oscillator is off. The internal RAM and all of the registers (except SYSCTRL) in the S1D13C00 are synchronous. To access the synchronous memory and registers, the IOSC oscillator needs to be on and running at maximum frequency (20MHz, SYSCTRL.IOSCFQ[1:0] = 11b) for fastest access. The SYSCTRL which controls the IOSC oscillator can be accessed asynchronously.

ACTIVE-to-IDLE Procedure

1. Write 0 to the IOSCEN bit of SYSCTRL register to turn off IOSC.

IDLE-to-ACTIVE Procedure

1. Set the IOSCFQ[1:0] bits of the SYSCTRL register to desired frequency (normally 11b to select 20MHz for fastest access time and processing) and the IOSCEN bit 1 to turn on the IOSC oscillator.
2. Before accessing synchronous memory and registers, wait 3us, or poll the IOSCSTA bit of the SYSCTRL register to wait for it to go to 1 indicating that IOSC is stable.

OSC1 State and Selection

OSC1 is used mainly for the VCOM/XFRP output to the panel and for the RTC (real-time clock) to maintain time/date. Normally, for “always on” display operation, the OSC1 clock should be turned on and kept on to generate the VCOM/XFRP output. OSC1 is turned off only if the display is turned off (VCOM/XFRP outputs turned off). See Section 21.4 for the procedure to turn on/off the VCOM/XFRP outputs, and Section 7.1.2 for the procedure to turn on/off the OSC1 oscillator.

If the RTC is used, an external 32.768kHz crystal circuit connected to OSC1/OSC2 is needed to provide the accuracy and tolerance needed for the RTC time-keeping. If the RTC is not used, the external 32.768kHz crystal circuit is not needed and the internal 32kHz oscillator can be used as the OSC1 clock source.

8. D.C. Characteristics

8.1 Absolute Maximum Ratings

Table 8.1 Absolute Maximum Ratings

Item	Symbol	Condition	Rated value	Unit
Power supply voltage	VDD		-0.3 to 7.0	V
Host interface I/O supply voltage	HIFVDD		-0.3 to 7.0	V
Panel interface I/O supply voltage	DISPVDD		-0.3 to 7.0	V
VCOM/XFRP I/O supply voltage	VCOMVDD		-0.3 to 7.0	V
External supply voltage for display	VC3		-0.3 to 7.0	V
Input voltage	V_I	QSPICLK, #QSPISS, QSPIDx, TEST, #RESET, P0x, P1x	-0.3 to VDD + 0.5	V
		HIFx, #HIFx	-0.3 to HIFVDD + 0.5	V
		XRST, VST, VCK, ENB, HST, HCK, REDx, GRNx, BLUx	-0.3 to DISPVDD + 0.5	V
		VCOM, XFRP	-0.3 to VCOMVDD + 0.5	V
Output voltage	V_O	QSPICLK, #QSPISS, QSPIDx, TEST, #RESET, P0x, P1x	-0.3 to VDD + 0.5	V
		HIFx, #HIFx	-0.3 to HIFVDD + 0.5	V
		XRST, VST, VCK, ENB, HST, HCK, REDx, GRNx, BLUx	-0.3 to DISPVDD + 0.5	V
		VCOM, XFRP	-0.3 to VCOMVDD + 0.5	V
High level output current	I_{OH}	1 pin	-10	mA
		Total of all pins	-20	mA
Low level output current	I_{OL}	1 pin	10	mA
		Total of all pins	20	mA
Operating temperature	Ta		-40 to 85	°C
Storage temperature	Tstg		-65 to 125	°C

Note
V_{SS} = 0V

8.2 Recommended Operating Conditions

Table 8.2 Recommended Operating Conditions

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Power supply voltage	VDD	For normal operation, Voltage Booster/Regulator off	1.8	-	5.5	V
		Voltage Booster/Regulator on	2.0	-	5.5	V
I/O power supply voltage	HIFVDD	For host interface	1.8	-	5.5	V
	DISPVDD	For panel interface	1.8	-	5.5	V
	VCOMVDD	For VCOM and XFRP	1.8	-	5.5	V
External power supply voltage	VC3	*2	5.2	-	5.6	V
OSC1 oscillator oscillation frequency	fOSC1	Crystal oscillator	-	32.768	-	kHz
Bypass capacitor between VSS and VDD	C_{PW1}		-	3.3	-	uF
Capacitor between Vss and VD1	C_{PW2}		-	1	1.2	uF
Capacitor between Vss and DISPVDD	C_{PW3}	*3	-	1	-	uF
Capacitor between Vss and VCOMVDD	C_{PW4}	*4	-	1	-	uF
Capacitor between Vss and HIFVDD	C_{PW5}	*5	-	1	-	uF
Capacitor between Vss and VMDL	C_{MDC1}		-	1	-	uF
Capacitor between Vss and VMDH	C_{MDC2}		-	1	-	uF
Capacitor between Vss and VC1	C_{MDC3}	*2	-	1	-	uF
Capacitor between Vss and VC2, Vss and VC3	C_{MDC4-5}	*2	-	1	-	uF
Capacitor between CP1 and CP2, CP1 and CP3	C_{MDC6-7}	*2	-	1	-	uF
Gate capacitor for OSC1 oscillator	C_{G1}	When crystal oscillator is used *1	0	-	25	pF
Drain capacitor for OSC1 oscillator	C_{D1}	When crystal oscillator is used *1	-	0	-	pF

V_{SS} = 0V

*1 The component values should be determined after performing matching evaluation of the resonator mounted on the printed.

*2 C_{MDC3}, C_{MDC4}, C_{MDC6}, C_{MDC7} are not needed if external VC3 supply is applied. See Sections 4.5 and 21.3.4 for more details.

*3 If DISPVDD is supplied by VMDL, it can share the same bypass capacitor as VMDL (C_{MDC1}) and C_{PW3} can be removed (is not needed).

*4 If VCOMVDD is supplied by VMDL, it can share the same bypass capacitor as VMDL (C_{MDC1}) and C_{PW4} can be removed (is not needed).

*5 If HIFVDD is supplied by VDD, it can share the same bypass capacitor as VDD (C_{PW1}) and C_{PW5} can be removed (is not needed).

8.3 Current Consumption

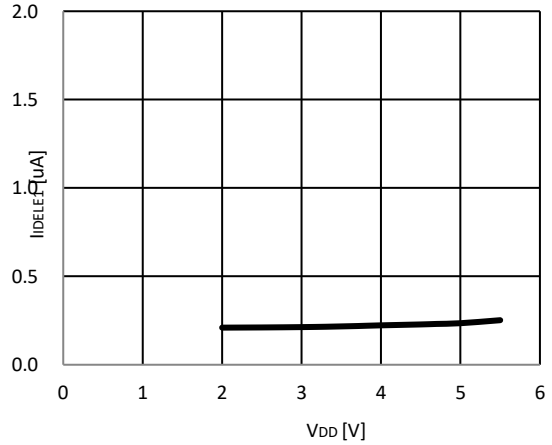
Unless otherwise specified: VDD = 1.8 to 5.5V,VSS = 0V, Ta = 25 °C

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Current consumption in IDLE mode (IOSCEN=0)	I_{IDLE1}	OSC1=Off, RTC Off, VCOM=Off, IOSC=Off, Voltage booster/regulator Off, Display not connected	-	0.26	3.0	µA
	I_{IDLE2}	OSC1=**external 32.768kHz**, RTC On, VCOM=Off, IOSC=Off, Voltage booster/regulator Off, Display not connected	-	0.8	4.0	µA
	I_{IDLE3}	OSC1=**internal 32kHz**, RTC Off, VCOM=Off, IOSC=Off, Voltage booster/regulator Off, Display not connected	-	1.7	5.0	µA
	I_{IDLE4}	OSC1=external 32.768kHz, RTC On, VCOM=60Hz, IOSC=Off, **Voltage booster/regulator On (OSC1/32),** REGECO=1, VMDBUP=0, Display not connected	-	2.8	6.0	µA
Current consumption in ACTIVE mode (IOSCEN=1)	$I_{ACTIVE1}$	OSC1=external 32.768kHz, RTC On, VCOM=60Hz, IOSC=20MHz, Voltage booster/regulator (OSC1/1), Display not connected, No activity	-	650	750	µA
	$I_{ACTIVE2}$	OSC1=external 32.768kHz, RTC On, VCOM=60Hz, IOSC=20MHz, Voltage booster/regulator On, Display not connected, **Filled Rectangle drawing**	-	2.4	2.9	mA
	$I_{ACTIVE3}$	OSC1=external 32.768kHz, RTC On, VCOM=60Hz, IOSC=20MHz, Voltage booster/regulator On, Display not connected, **Image copy with rotation and scaling RAM-to-RAM**	-	1.9	2.3	mA
	$I_{ACTIVE4}$	OSC1=external 32.768kHz, RTC On, VCOM=60Hz, IOSC=20MHz, Voltage booster/regulator On, Display not connected, **Display panel update (checkered)**	-	1.6	1.9	mA

*1 When voltage booster/regulator is on, VDD = 2.0 to 5.5V
*2 Measurement condition: VMDL is connected to DISPVDD and VCOMVDD, current is measured at VDD supply

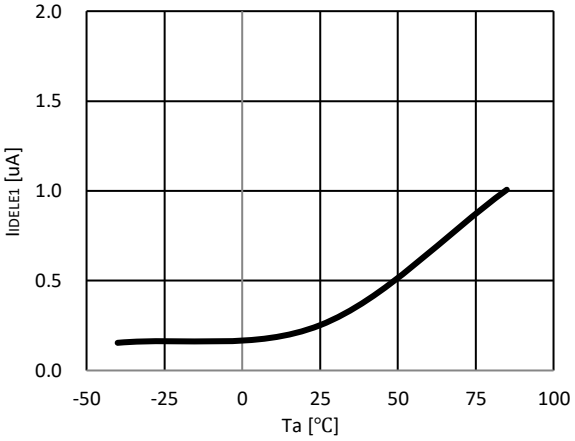
Current consumption - power supply voltage characteristic in IDLE1

IOSC = OFF, OSC1 = OFF, Typ. value



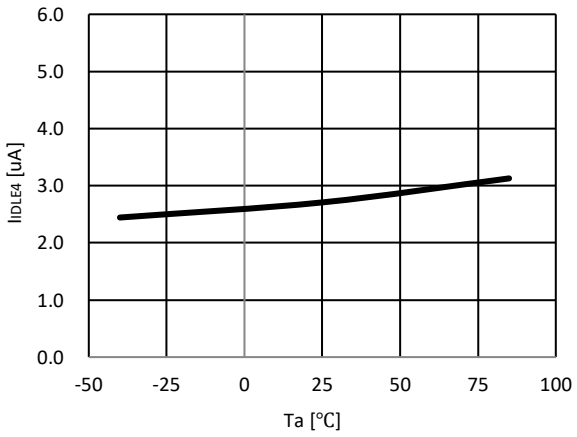
Current consumption - temperature characteristic in IDLE1

IOSC = OFF, OSC1 = OFF, Typ. value



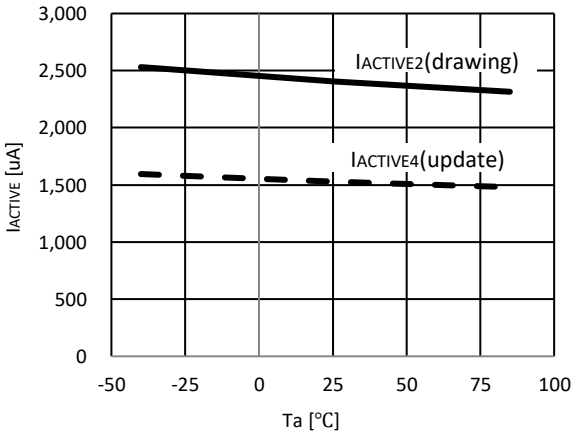
Current consumption - temperature characteristic in IDLE4

IOSC=OFF, OSC1=32.768kHz, Voltage booster=ON, Typ. value



Current consumption - temperature characteristic in ACTIVE2,4

IOSC=20MHz, OSC1=32.768kHz, Voltage booster=ON, Typ. value

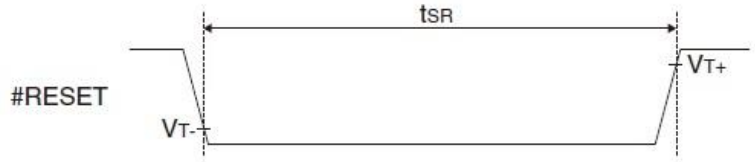


8.4 Reset Characteristics

#RESET pin characteristics

Unless otherwise specified: VDD = 1.8 to 5.5V,VSS = 0V, Ta = -40 to 85 °C

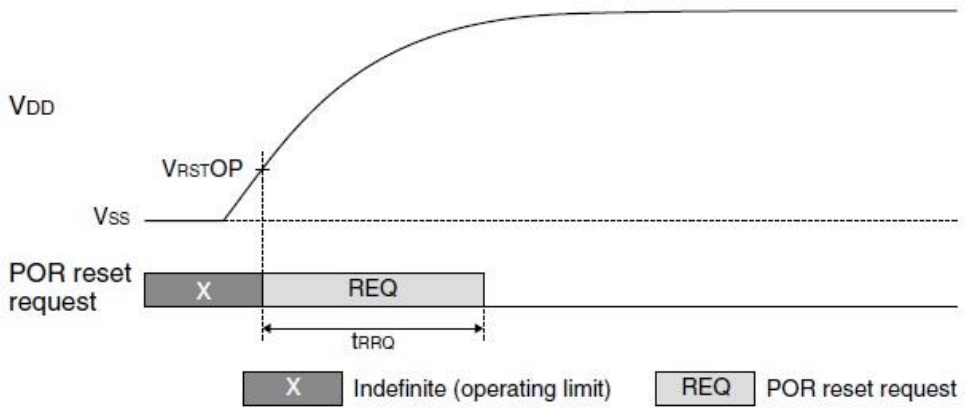
Item	Symbol	Condition	Min.	Typ.	Max.	Unit
High level Schmitt input threshold voltage	V_{T+}		0.5 x VDD	-	0.8 x VDD	V
Low level Schmitt input threshold voltage	V_{T-}		0.2 x VDD	-	0.5 x VDD	V
Schmitt input hysteresis voltage	ΔV_T		180	-	-	mV
Input pull-up resistance	R_{IN}		100	270	500	kΩ
Pin capacitance	C_{IN}		-	-	15	pF
Reset Low pulse width	t_{SR}		5	-	-	µs



POR characteristics

Unless otherwise specified: VDD = 1.8 to 5.5V,VSS = 0V, Ta = -40 to 85 °C

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
POR operating limit voltage	$V_{RST}OP$		-	0.5	0.95	V
POR reset request hold time	t_{RRQ}		0.01	-	4	ms



Note: When performing a power-on-reset again after the power is turned off, ensure that the VDD voltage has dropped below $V_{RST}OP$ before powering up.

Reset hold circuit characteristics

Unless otherwise specified: VDD = 1.8 to 5.5V,VSS = 0V, Ta = -40 to 85 °C

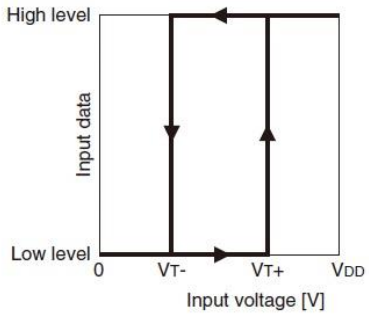
Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Reset hold time*1	t_{RSTR}		-	-	150	us

*1 Time until the internal reset signal is negated after the reset request is cancelled (#RESET de-assertion).

8.5 Input/Output Port (GPIO) Characteristics

Unless otherwise specified: VDD = 1.8 to 5.5V,VSS = 0V, Ta = -40 to 85 °C

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
High level Schmitt input threshold voltage	V_{T+}		0.5 x VDD	-	0.8 x VDD	V
Low level Schmitt input threshold voltage	V_{T-}		0.2 x VDD	-	0.5 x VDD	V
Schmitt input hysteresis voltage	ΔV_T		180	-	-	mV
High level output current	I_{OH}	V_{OH} = 0.9 x VDD	-	-	-0.4	mA
Low level output current	I_{OL}	V_{OL} = 0.1 x VDD	0.4	-	-	mA
Leakage current	I_{LEAK}		-150	-	150	nA
Input pull-up resistance	R_{INU}		75	150	300	kΩ
Input pull-down resistance	R_{IND}		75	150	300	kΩ
Pin capacitance	C_{IN}		-	-	15	pF



8.6 Voltage Booster/Regulator Characteristics

The voltage booster/regulator characteristics varies depending on the panel load (panel size, drive duty, number of display pixels and display contents), so evaluate them by connecting to the actual panel used.

Unless otherwise specified: VDD = 2.0 to 5.5V,VSS = 0V, Ta = 25 °C, MDCBSTCLK.CLKSRC[1:0] bits = 0x1, MDCBSTCLK.CLKDIV[2:0] = 0x0 (voltage booster clock = OSC1 frequency), MDCBSTPWR.REGON = 1, MDCBSTPWR.BSTON = 1, MDCBSTPWR.REGECO bit = 0, MDCBSTPWR.VMDBUP bit = 0, no panel load

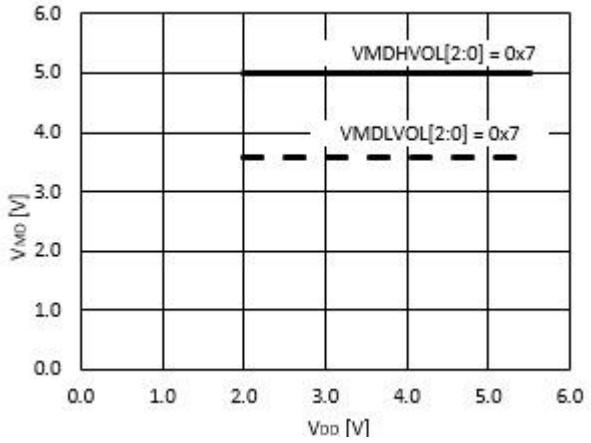
Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Panel drive voltage	V_{MDH}*2	MDCBSTVMD.VMDHVOL[2:0] bits = 0x0	4.2	4.3	4.4	V
		MDCBSTVMD.VMDHVOL[2:0] bits = 0x1	4.3	4.4	4.5	V
		MDCBSTVMD.VMDHVOL[2:0] bits = 0x2	4.4	4.5	4.6	V
		MDCBSTVMD.VMDHVOL[2:0] bits = 0x3	4.5	4.6	4.7	V
		MDCBSTVMD.VMDHVOL[2:0] bits = 0x4	4.6	4.7	4.8	V
		MDCBSTVMD.VMDHVOL[2:0] bits = 0x5	4.7	4.8	4.9	V
		MDCBSTVMD.VMDHVOL[2:0] bits = 0x6	4.8	4.9	5.0	V
		MDCBSTVMD.VMDHVOL[2:0] bits = 0x7	4.9	5.0	5.1	V
	V_{MDL}*2	MDCBSTVMD.VMDLVOL[2:0] bits = 0x0	2.2	2.3	2.4	V
		MDCBSTVMD.VMDLVOL[2:0] bits = 0x1	2.6	2.7	2.8	V
		MDCBSTVMD.VMDLVOL[2:0] bits = 0x2	2.9	3.0	3.1	V
		MDCBSTVMD.VMDLVOL[2:0] bits = 0x3	3.0	3.1	3.2	V
		MDCBSTVMD.VMDLVOL[2:0] bits = 0x4	3.1	3.2	3.3	V
		MDCBSTVMD.VMDLVOL[2:0] bits = 0x5	3.2	3.3	3.4	V
		MDCBSTVMD.VMDLVOL[2:0] bits = 0x6	3.3	3.4	3.5	V
		MDCBSTVMD.VMDLVOL[2:0] bits = 0x7	3.5	3.6	3.7	V
V_{MD} load current*1	L_{VMD}	MDCBSTVMD.VMD*VOL[2:0] bits = 0x7 MDCBSTCLK.CLKDIV[2:0] bits = 0x0	-	-	2	mA
		MDCBSTVMD.VMD*VOL[2:0] bits = 0x7 MDCBSTCLK.CLKDIV[2:0] bits = 0x5	-	-	100	µA
	L_{VMDE}*2	MDCBSTVMD.VMD*VOL[2:0] bits = 0x7	-	-	2	mA
Boosted voltage output stabilization time	t_{BST}	C_{MDC*} = 1µF	-	-	5	ms
VMD voltage output stabilization time	t_{VMD}*2	MDCBSTVMD.VMD*VOL[2:0] bits = 0x7 C_{MDC1}, C_{MDC2} = 1µF	-	-	5	ms

*1 VMDL and VMDH total load current

*2 When applying an external power source through the VC3 pin: VDD=1.8 to 5.5V, MDCBSTPWR.REGON and MDCBSTPWR.BSTON bits = 0.

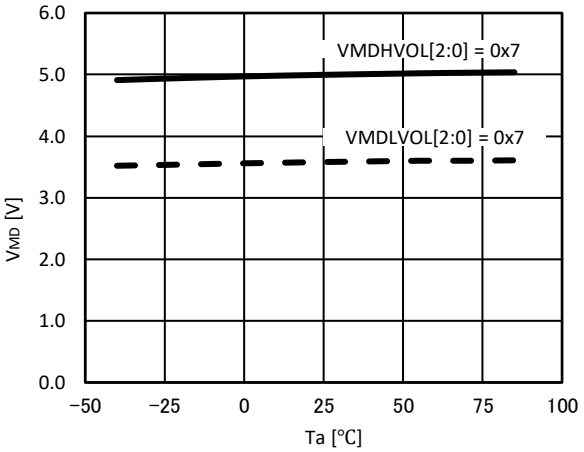
Panel drive voltage – power supply voltage characteristic

L_{VMD}=1mA, Typ. value



Panel drive voltage - temperature characteristic

L_{VMD}=1mA, Typ. value



9. A.C. Characteristics

9.1 Clock Generator (CLG) Characteristics

Oscillator circuit characteristics including resonators change depending on conditions (board pattern, components used, etc.). Use these characteristic values as a reference and perform matching evaluation using the actual printed circuit board.

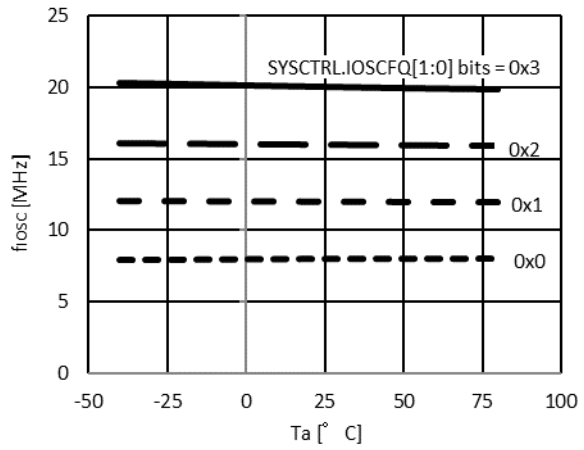
IOSC oscillator circuit characteristics

Unless otherwise specified: VDD = 1.8 to 5.5 V, VSS = 0 V, Ta = -40 to 85 °C

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Oscillation start time	t_{sta1}		-	-	3	us
Oscillation frequency	f_{IOSC}	SYSCTRL.IOSCFQ[1:0] = 0x3	18	20	21	MHz
		SYSCTRL.IOSCFQ[1:0] = 0x2	14.4	16	16.8	MHz
		SYSCTRL.IOSCFQ[1:0] = 0x1	10.8	12	12.6	MHz
		SYSCTRL.IOSCFQ[1:0] = 0x0	7.2	8	8.4	MHz

IOSC oscillation frequency-temperature characteristic

VDD = 1.8 to 5.5 V, SYSCTRL.IOSCAJ[5:0] = 0x00



OSC1 oscillator circuit characteristics

Unless otherwise specified: VDD = 1.8 to 5.5 V, VSS = 0 V, Ta = 25 °C

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Crystal oscillator oscillation start time*1	t_{sta1c}	CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.INV1N[1:0] bits = 0x1, CLGOSC1.INV1B[1:0] bits = 0x2, CLGOSC1.OSC1BUP bit = 1	-	-	3	s
Crystal oscillator internal gate capacitance	C_{GI1C}	CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.CGI1[2:0] bits = 0x0	-	12	-	pF
		CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.CGI1[2:0] bits = 0x1	-	14	-	pF
		CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.CGI1[2:0] bits = 0x2	-	16	-	pF
		CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.CGI1[2:0] bits = 0x3	-	18	-	pF
		CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.CGI1[2:0] bits = 0x4	-	19	-	pF
		CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.CGI1[2:0] bits = 0x5	-	21	-	pF
		CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.CGI1[2:0] bits = 0x6	-	23	-	pF
		CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.CGI1[2:0] bits = 0x7	-	24	-	pF
Crystal oscillator internal drain capacitance	C_{DI1C}	CLGOSC1.OSC1SELCR bit = 0	-	6	-	pF
Crystal oscillator oscillator circuit current – oscillation inverter drivability ratio*1	I_{OSC1C}	CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.INV1N/INV1B[1:0] bits = 0x0	-	70	-	%
		CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.INV1N/INV1B[1:0] bits = 0x1 (reference)	-	100	-	%
		CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.INV1N/INV1B[1:0] bits = 0x2	-	130	-	%
		CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.INV1N/INV1B[1:0] bits = 0x3	-	300	-	%
Crystal oscillator oscillation stop detector current	I_{OSD1C}	CLGOSC1.OSC1SELCR bit = 0, CLGOSC1.OSDEN bit = 1	-	0.025	0.1	uA
Internal oscillator oscillation start time	t_{sta1I}	CLGOSC1.OSC1SELCR bit = 1	-	-	100	us
Internal oscillator oscillation frequency	f_{OSC1I}	CLGOSC1.OSC1SELCR bit = 1	31.04	32	32.96	kHz

*1 CLGOSC1.CGI1[2:0] bits = 0x0, Crystal resonator = C-002RX (manufactured by Seiko Epson Corporation, R1 = 50 kΩ (Max.), CL = 7 pF)

9.2 Host Interface Characteristics

9.2.1 Indirect 8-bit parallel interface

Unless otherwise specified: VSS = 0V, HIFVDD = 1.8 to 5.5V, Ta = -40 to 85 °C

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
#HIFDE setup time	t_{DES1}		0	-	-	ns
#HIFDE hold time (at write)	t_{DEHW}		t_{WRH2}, t_{WRH3}	-	-	ns
#HIFDE hold time (at read)	t_{DEHR}		t_{RDH2}, t_{RDH3}	-	-	ns
#HIFDE High pulse width	t_{DWH}		5	-	-	ns
#HIFCS setup time (at write)	t_{CSWS}		0	-	-	ns
#HIFCS hold time (at write)	t_{WCSH}		0	-	-	ns
#HIFWR Low pulse width (command/address byte)	t_{WRL1}		7	-	-	ns
#HIFWR Low pulse width (asynchronous register data byte[*1])	t_{WRL2}		14	-	-	ns
#HIFWR Low pulse width (synchronous memory/register write[*2])	t_{WRL3}		[*3] t_{SYSCLK} + 6	-	-	ns
#HIFCS High pulse width (at write) (command/address byte)	t_{WRH1}		9	-	-	ns
#HIFCS High pulse width (at write) (asynchronous register data byte[*1])	t_{WRH2}		9	-	-	ns
#HIFCS High pulse width (at write) (synchronous memory/register write[*2])	t_{WRH3}		[*3] t_{SYSCLK} x (3 + 2 x nACTIVE[*4,5])	-	-	ns
Write data setup time	t_{WDS1}		7	-	-	ns
Write data hold time	t_{WDH1}		9	-	-	ns
#HIFCS setup time (at read)	t_{CSRS}		0	-	-	ns
#HIFCS hold time (at read)	t_{RCSH}		0	-	-	ns
#HIFRD Low pulse width (asynchronous register data byte[*1])	t_{RDL2}		60	-	-	ns
#HIFRD Low pulse width (synchronous memory/register read[*2])	t_{RDL3}		[*3] t_{SYSCLK} + 5	-	-	ns
#HIFRD low to HIFD[7:0] low impedance	t_{RDO1}		-	-	59	ns
#HIFRD high to HIF[7:0] high impedance	t_{RDDH1}		6	-	33	ns
#HIFCS High pulse width (at read) (asynchronous register data byte[*1])	t_{RDH2}		33	-	-	ns
#HIFCS High pulse width (at read) (synchronous memory/register data byte[*2])	t_{RDH3}		[*3] t_{SYSCLK} x (7 + 2 x nACTIVE[*4,5])	-	-	ns

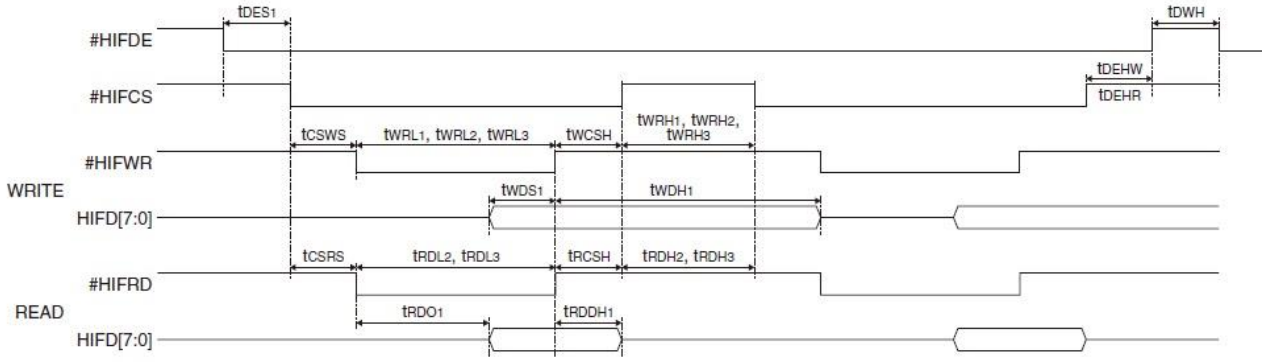
*1 Asynchronous registers are SYSCTRL and SYSINTS.

*2 "Synchronous memory/register" refers to the internal RAM and synchronous registers. The timing specification does not include host reads from memory-mapped serial flash. Host reads from memory-mapped serial flash area (0x00040000 – 0x0013FFFF) is not recommended because it takes many system clock cycles and is unreliable. For host reading of memory-mapped serial flash data, it is recommended to use DMAC to transfer blocks to RAM and then reading from RAM.

*3 t_{SYSCLK}: System clock period [ns]. Typically 50ns (system clock = 20MHz).

*4 nACTIVE: Number of other active internal bus masters (0 to 4) accessing the same memory map area.
The internal bus has 4 bus masters: DMAC, MDC Graphics Engine, Event Processor and MDC Display Updater.
Internal bus switch matrix allows parallel access by each bus master to different areas. However, when two bus masters simultaneously access the same area, there is arbitration logic to let only one master access at a time and the other master(s) have to wait until the current access-granted master is finished. If the host is accessing an area in the memory map and there are no other bus masters accessing the same area, then nACTIVE=0. Refer to Section 11.3.4 for more details.

*5 Host read from Memory-Mapped Serial Flash is not recommended. Please refer to Section 11.3.3 for more details.
Read and write access by Host of RAM or registers is not recommended while either one of the MDC Graphics Engine or MDC Event Processor is actively reading from Memory-Mapped Serial Flash because the internal bus can be held for many system clock cycles resulting in unreliable access from Host MCU. Refer to Section 11.3.4 for more details.


tDES1
tDWH
#HIFDE
tDEHW
#HIFCS
tDEHR
tCSWS
tWRL1, tWRL2, tWRL3
tWCSH
tWRH1, tWRH2, tWRH3
#HIFWR
WRITE
tWDS1
tWDH1
HIFD[7:0]
tCSRS
tRDL2, tRDL3
tRCSH
tRDH2, tRDH3
#HIFRD
READ
tRDO1
tRDDH1
HIFD[7:0]

9.2.2 SPI/QSPI interface

Unless otherwise specified: VSS = 0V, HIFVDD = 1.8 to 5.5V, Ta = -40 to 85 °C

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
#HSPISS setup time	t_{CSS}		9	-	-	ns
#HSPISS hold time	t_{CSH}		13	-	-	ns
#HSPISS High pulse width	t_{CWH}		28	-	-	ns
HSPID write data setup time	t_{WDS}		3	-	-	ns
HSPID write data hold time	t_{WDH}		11	-	-	ns
HSPID low impedance waiting time	t_{RLOZ}		12	-	60	ns
HSPID valid data waiting time	t_{RDO}		9	-	51	ns
HSPID high impedance waiting time	t_{RHIZ}		6	-	34	ns
HSPICLK High pulse width (command/address/ asynchronous register data byte[*1])	t_{CLKH1}		11	-	-	ns
HSPICLK Low pulse width (command/address/ asynchronous register data byte[*1])	t_{CLKL1}		51	-	-	ns
HSPICLK High pulse width (at write) (synchronous memory/register data byte[*2])	t_{CLKH2}		[*3] t_{SYSCLK} x 0.25 x (5 + 2 x nACTIVE[*4,5])	-	-	ns
HSPICLK Low pulse width (at write) (synchronous memory/register data byte[*2])	t_{CLKL2}		[*3] t_{SYSCLK} x 0.25 x (5 + 2 x nACTIVE[*4,5])	-	-	ns
HSPICLK High pulse width (at read) (synchronous memory/register data byte[*2])	t_{CLKH3}		[*3] t_{SYSCLK} x (8 + 2 x nACTIVE[*4,5])	-	-	ns
HSPICLK Low pulse width (at read) (synchronous memory/register data byte[*2])	t_{CLKL3}		51	-	-	ns

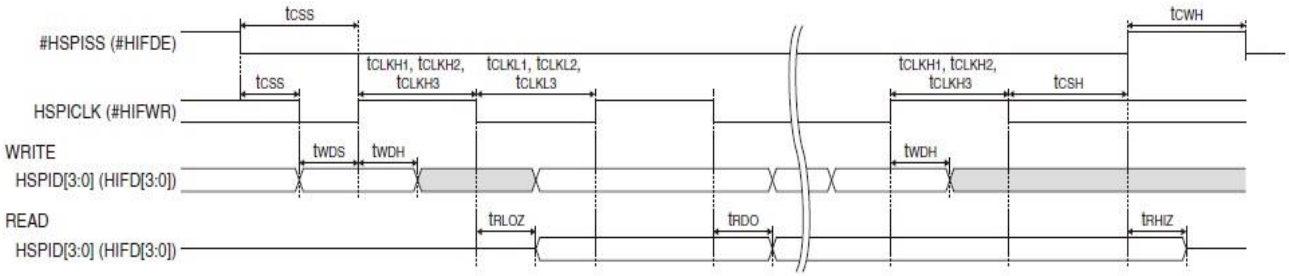
*1 Asynchronous registers are SYSCTRL and SYSINTS.

*2 "Synchronous memory/register" refers to the internal RAM and synchronous registers. The timing specification does not include host reads from memory-mapped serial flash. Host reads from memory-mapped serial flash area (0x00040000 – 0x0013FFFF) is not recommended because it takes many system clock cycles and is unreliable. For host reading of memory-mapped serial flash data, it is recommended to use DMAC to transfer blocks to RAM and then reading from RAM.

*3 t_{SYSCLK}: System clock period [ns]. Typically 50ns (system clock = 20MHz).

*4 nACTIVE: Number of other active internal bus masters (0 to 4) accessing the same memory map area.
The internal bus has 4 bus masters: DMAC, MDC Graphics Engine, Event Processor and MDC Display Updater.
Internal bus switch matrix allows parallel access by each bus master to different areas. However, when two bus masters simultaneously access the same area, there is arbitration logic to let only one master access at a time and the other master(s) have to wait until the current access-granted master is finished. If the host is accessing an area in the memory map and there are no other bus masters accessing the same area, then nACTIVE=0. Refer to Section 11.3.4 for more details.

*5 Host read from Memory-Mapped Serial Flash is not recommended. Please refer to Section 11.3.3 for more details.
Read and write access by Host of RAM or registers is not recommended while either one of the MDC Graphics Engine or MDC Event Processor is actively reading from Memory-Mapped Serial Flash because the internal bus can be held for many system clock cycles resulting in unreliable access from Host MCU. Refer to Section 11.3.4 for more details.



9.3 Panel Interface Characteristics

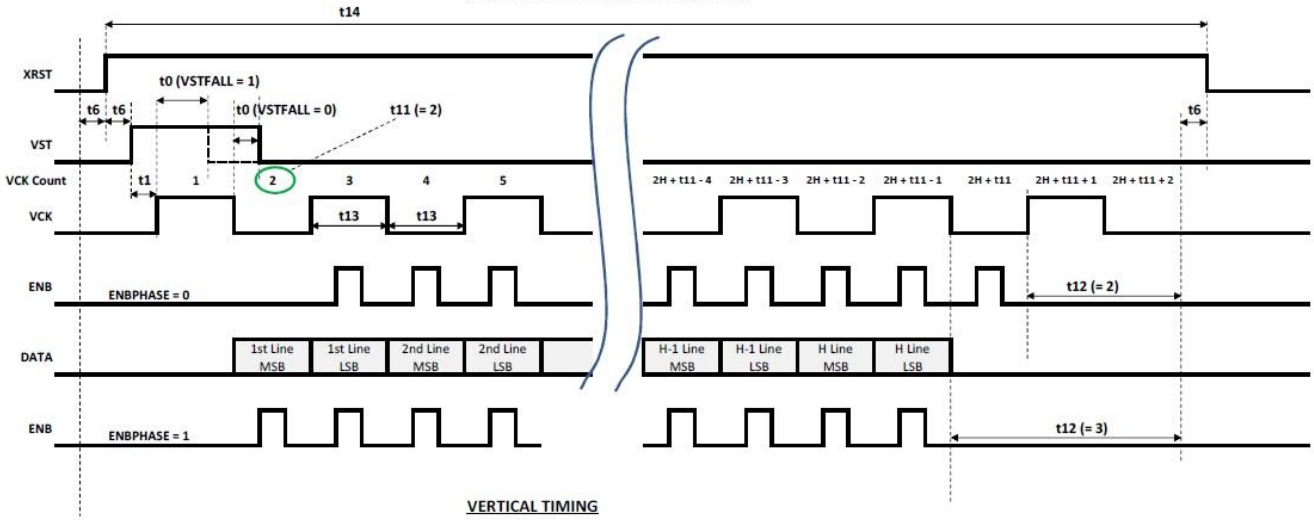
9.3.1 6-Bit Color Panel

Unless otherwise specified: VSS = 0V, DISPVDD=1.8 to 5.5V, VCOMVDD=1.8 to 5.5V, Ta = -40 to 85 °C

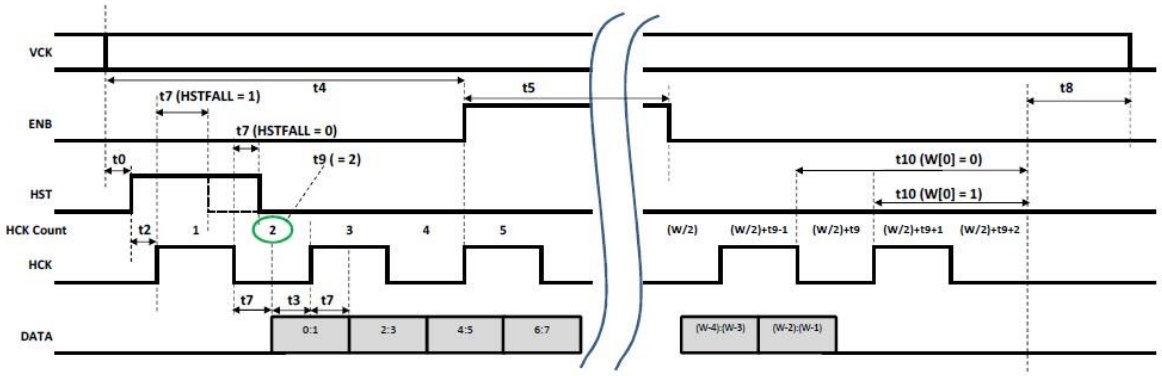
Item	Symbol	Nominal[*3]	Unit	Variation from Nominal[*3]		
				Min	Max	Unit
Panel Timing unit	T	(MDCDISPCLKDIV.CLKDIV[7:0] + 1) x t_{SYSCLK} [*1]	ns			
VCK rise/fall to HST rise	t0	(MDCDISPCLKDIV.TIM0[7:0] + 1)	T	-9	4	ns
VCK rise/fall to VST fall				-3	0	
VST rise to VCK rise	t1	(MDCDISPPRM21.TIM1[7:0] + 1)	T	1	4	ns
HST rise to HCK rise	t2	(MDCDISPPRM21.TIM2[7:0] + 1)	T	1	7	ns
Data to HCK rise/fall	t3	(MDCDISPPRM43.TIM3[7:0] + 1)	T	-10	2	ns
VCK rise/fall to ENB rise	t4	t0 + t2 + (MDCDISPPRM43.TIM4[7:0] x (t3 + t7))	T	-5	7	ns
ENB high width	t5	(MDCDISPPRM65.TIM5[7:0] x (t3 + t7))	T	-9	0	ns
Register trigger to XRST rise	t6	(MDCDISPPRM65.TIM6[7:0] + 1)	T	-	-	-
XRST rise to VST rise				0	2	ns
Last data end to XRST fall				-	-	-
HCK rise/fall to data	t7	(MDCDISPPRM87.TIM7[7:0] + 1)	T	-2	10	ns
HCK rise/fall to HST fall				-3	0	ns
End of line to next VCK rise/fall	t8	(MDCDISPPRM87.TIM8[7:0] + 1) x (t3 + t7)	T	-	-	-
HCK count for start of pixel data	t9	MDCDISPPRM109.TIM9[7:0]	-	-	-	-
Number of extra HCK counts after end of line data	t10	MDCDISPPRM109.TIM10[7:0]	-	-	-	-
VCK count for start of pixel data	t11	MDCDISPPRM1211.TIM11[7:0]	-	-	-	-
Number of extra VCK counts after last line of data	t12	MDCDISPPRM1211.TIM12[7:0]	-	-	-	-
VCK high/low width for normal display update	t13a	t0 + t2 + [(t3 + t7) x (t9 + (W/2) + t10 + W[0] [*4] - 1)] + t8	T	-	-	-
VCK high width for fast VCK partial update	t13b	MDCDISPPRM1413.TIM13[7:0]	T	-12	-1	ns
VCK low width for fast VCK partial update				1	12	ns
XRST high width for normal display update and partial update with normal VCK	t14a	t6 + t1 + (t13a x (t11 + 2H + t12 - ENBPHASE[*5])) + t6	T	-	-	-
XRST high width for partial update with Fast VCK	t14b	t6 + t1 + [t13a x (t2 + 2x(E-S+1) + F)] + [*6] [t13b x (2x(H+S-E-1) + t12 - F)] + t6	T	-	-	-
VCOM frequency (50% duty cycle) [*2]	f_{VCOM}	f_{OSC1} / (4 x (MDCDISPVCOMDIV[15:0] + 1))	Hz	-	-	-
VCOM high width	t_{VCOMH}	(4 x (MDCDISPVCOMDIV[15:0] + 1)) / (2 x f_{OSC1})	s	-11	-1	ns
VCOM low width	t_{VCOML}	(4 x (MDCDISPVCOMDIV[15:0] + 1)) / (2 x f_{OSC1})	s	1	11	ns

*1 t_{SYSCLK}: System clock period [ns]
*2 f_{OSC1}: OSC1 frequency [Hz]
*3 Output timing of two signals is relative to each other. The Nominal timing depends on register programming and is in units of T, and the T value changes with the system clock frequency (t_{SYSCLK}) variations. Assuming no variation in T, the "Variation from Nominal" is the relative gate delay between two output signals which can vary over operating conditions.
*4 W is the panel width in pixels. W[0] is bit 0 of the panel width value.
*5 H is the panel height in pixels. ENBPHASE = 0 if first ENB pulse starts on LSB. ENBPHASE = 1 if first ENB pulse starts on MSB.
*6 H is the panel height in pixels
S is the value of MDCDISPSTARTY[9:0]
E is the value of MDCDISPENDY[9:0]
F is 1 if t11 is odd and 2 if t11 is even

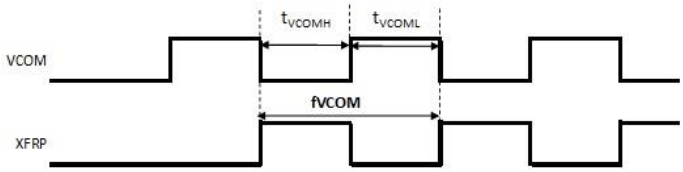
6-BIT COLOR PANEL (W x H Pixels)

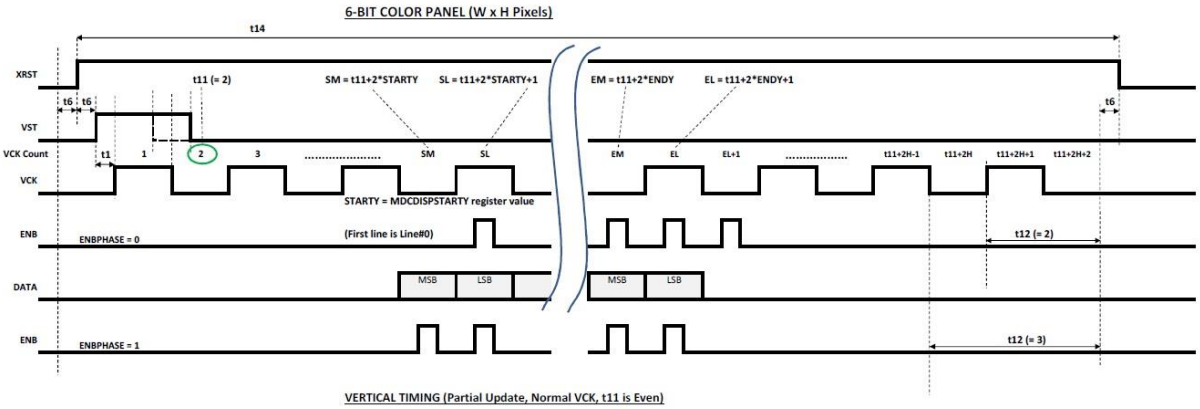


VERTICAL TIMING

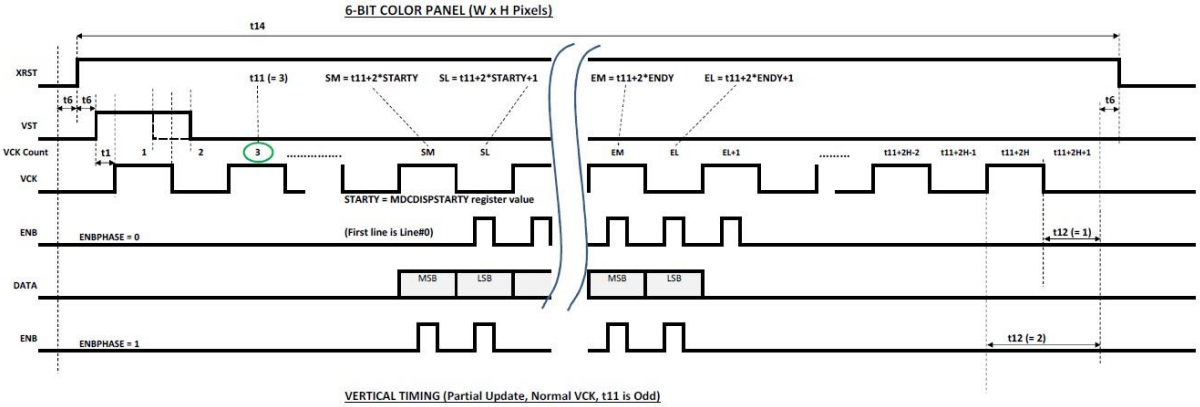


HORIZONTAL TIMING

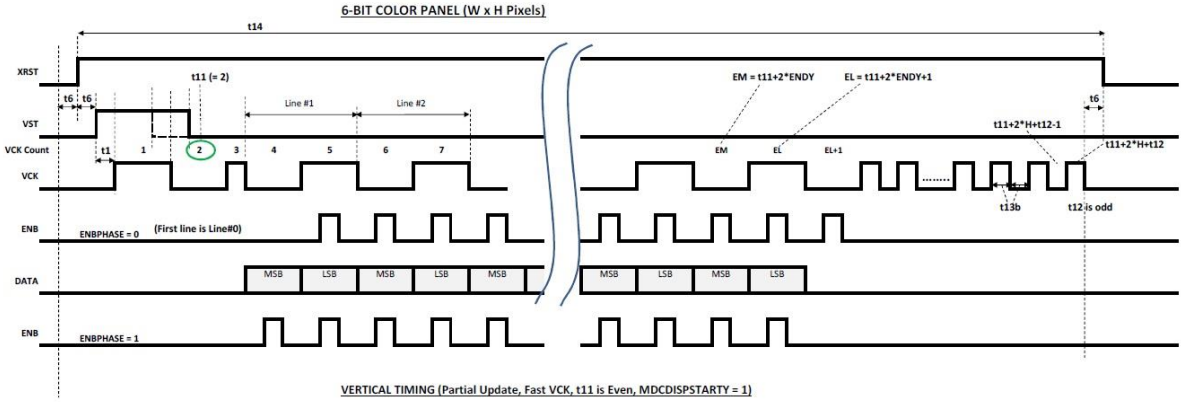




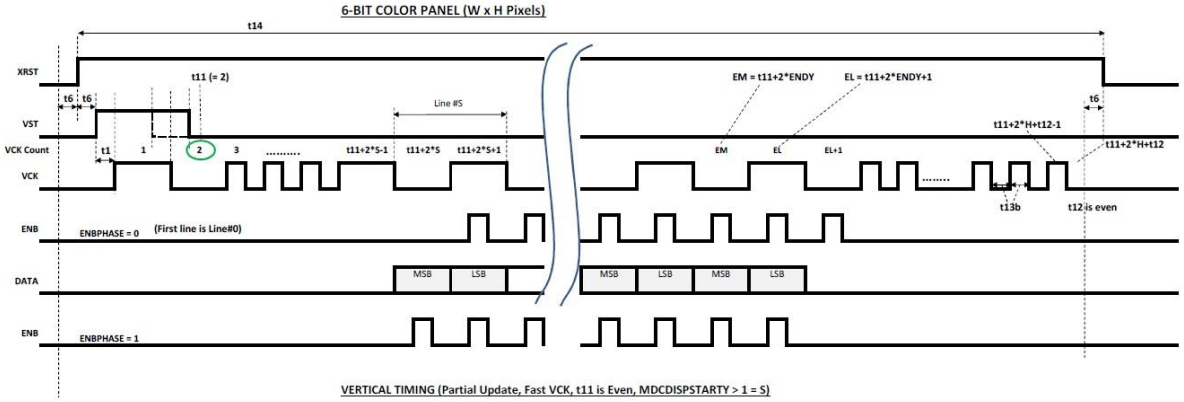
VERTICAL TIMING (Partial Update, Normal VCK, t11 is Even)



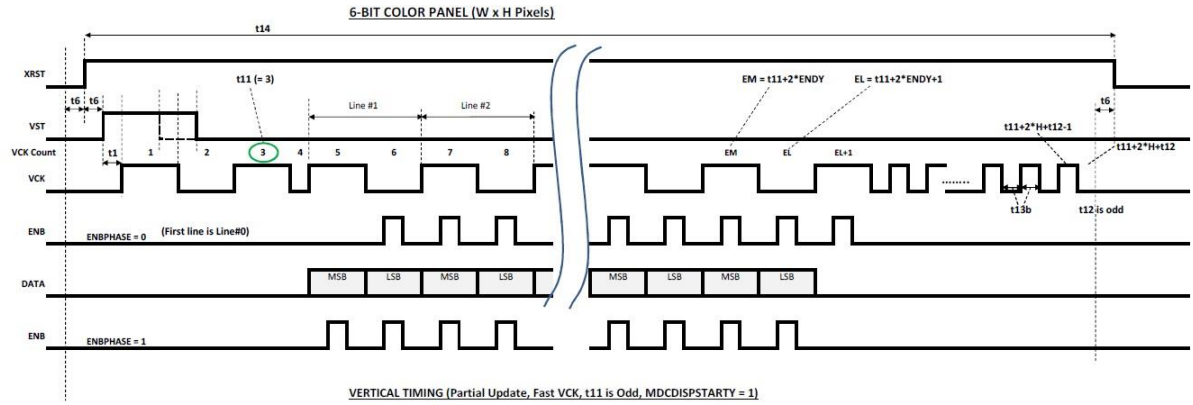
VERTICAL TIMING (Partial Update, Normal VCK, t11 is Odd)



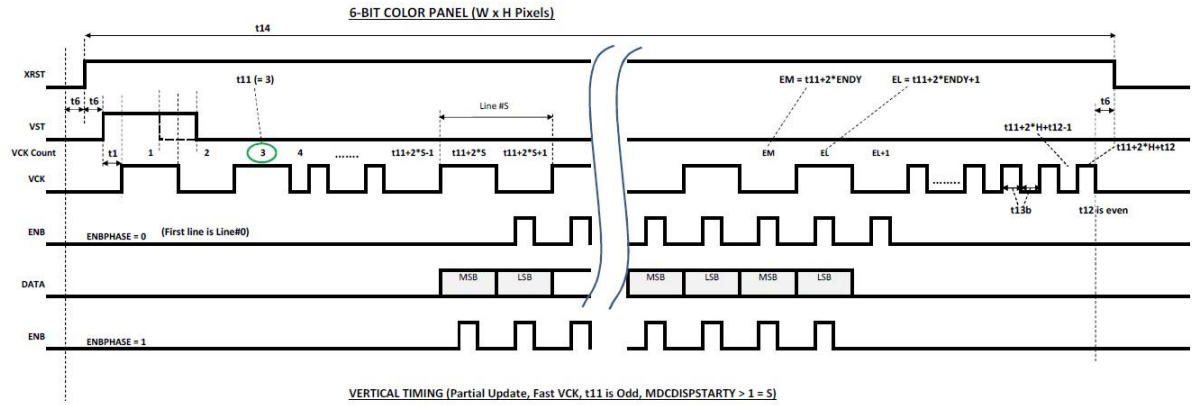
VERTICAL TIMING (Partial Update, Fast VCK, t11 is Even, MDCDISPSTARTY = 1)



VERTICAL TIMING (Partial Update, Fast VCK, t11 is Even, MDCDISPSTARTY > 1 = S)



VERTICAL TIMING (Partial Update, Fast VCK, t11 is Odd, MDCDISPSTARTY = 1)



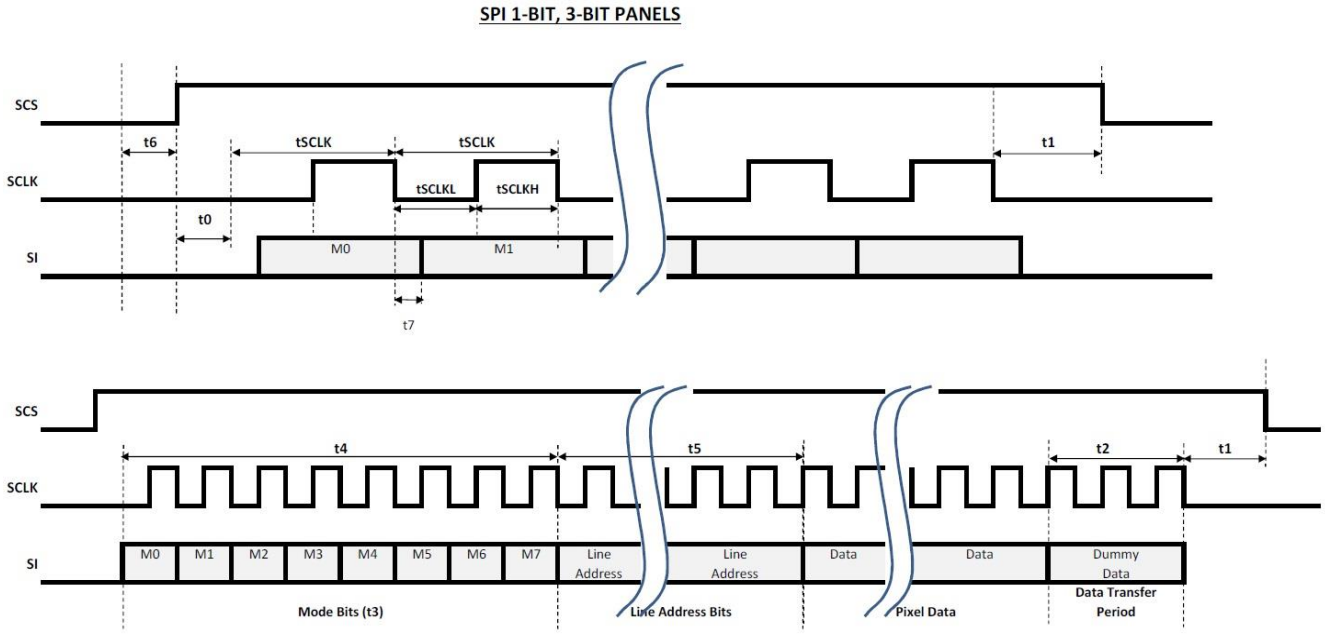
VERTICAL TIMING (Partial Update, Fast VCK, t11 is Odd, MDCDISPSTARTY > 1 = S)

9.3.2 SPI 1-Bit/3-Bit Panel

Unless otherwise specified: VSS = 0V, DISPVDD=1.8 to 5.5V, VCOMVDD=1.8 to 5.5V, Ta = -40 to 85 °C

Item	Symbol	Nominal	Unit	Variation from Nominal		
				Min	Max	Unit
SCLK period	t_{SCLK}	(MDCDISPCLKDIV.CLKDIV[7:0] + 1) x t_{SYSCLK} *1	ns			
SCLK high time	t_{SCLKH}	(((MDCDISPCLKDIV.CLKDIV[7:0] + 1)/2) + 1) x t_{SYSCLK} *1	ns	-11	0	ns
SCLK low time	t_{SCLKL}	= (t_{SCLK} - t_{SCLKH})	ns	0	11	ns
SCS rise to first SI data	t0	(MDCDISPCLKDIV.TIM0[7:0] + 1)	t_{SCLK}	0	2	ns
Last SI data (SCLK falling edge) to SCS fall	t1	(MDCDISPPRM21.TIM1[7:0] + 1)	t_{SCLK}	0	2	ns
Number of "dummy" clocks for data transfer period	t2	(MDCDISPPRM21.TIM2[7:0] + 1)	-	-	-	-
MODE bits value: M[7:0], M[0] (LSB) is sent first	-	MDCDISPPRM43.TIM3[7:0]	-	-	-	-
Number of MODE bits sent	t4	(MDCDISPPRM43.TIM4[7:0] + 1)	bits	-	-	-
Number of Line Address bits sent	t5	(MDCDISPPRM65.TIM5[7:0] + 1)	bits	-	-	-
Register trigger to SCS high	t6	(MDCDISPPRM65.TIM6[7:0] + 1)	t_{SCLK}	-	-	-
SCLK fall to next SI data	t7	-	ns	0	10	ns
COM frequency (50% duty cycle) *2	f_{COM}	f_{OSC1} / (4 x (MDCDISPVCOMDIV + 1))	Hz	-	-	-

*1 t_{SYSCLK}: System clock period [ns]
*2 f_{OSC1}: OSC1 frequency [Hz]

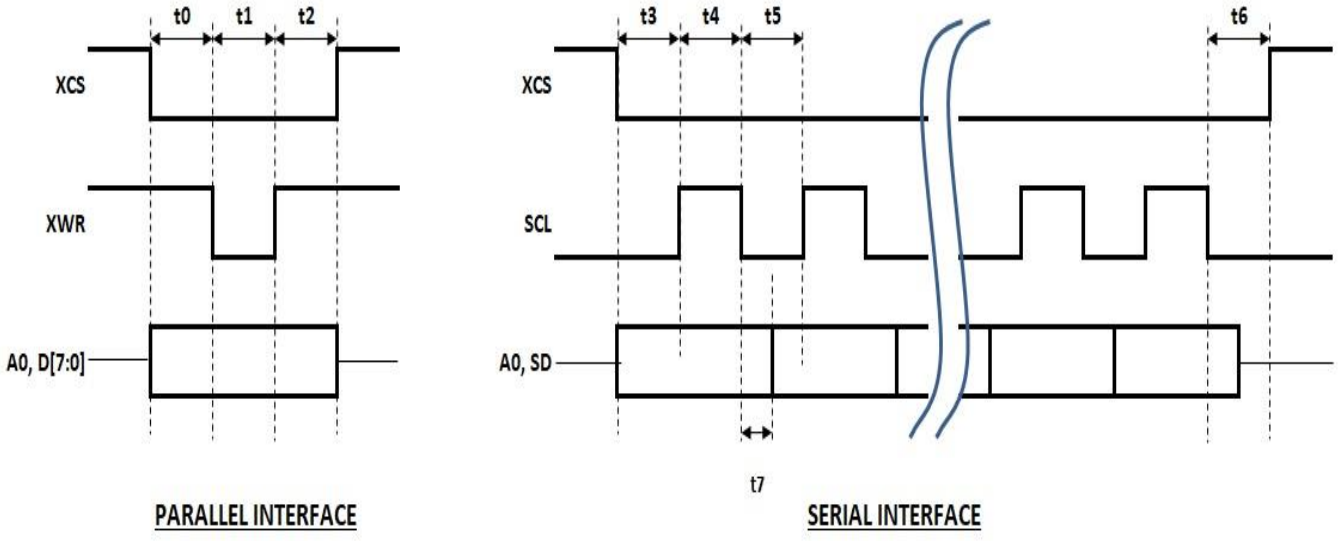


9.3.3 Grayscale Panel

Unless otherwise specified: VSS = 0V, DISPVDD=1.8 to 5.5V, VCOMVDD=1.8 to 5.5V, Ta = -40 to 85 °C

Item	Symbol	Nominal	Unit	Variation from Nominal		
				Min	Max	Unit
Panel Timing unit	T	(MDCDISPCLKDIV.CLKDIV[7:0] + 1) x t_{SYSCLK} *1	ns			
XCS/A0/D[7:0] before XWR fall setup	t0	1	T	-14	1	ns
XWR low width	t1	1	T	0	9	ns
XWR rise to XCS/A0/D[7:0] hold	t2	1	T	-2	10	ns
XCS/A0/SD before SCL rise setup	t3	1	T	0	10	ns
SCL high time	t4	1	T	-10	-1	ns
SCL low time	t5	1	T	1	10	ns
Last SCL fall to XCS rise	t6	-	-	0	8	ns
SCL fall to next data	t7	-	-	0	9	ns

*1 t_{SYSCLK}: System clock period [ns]

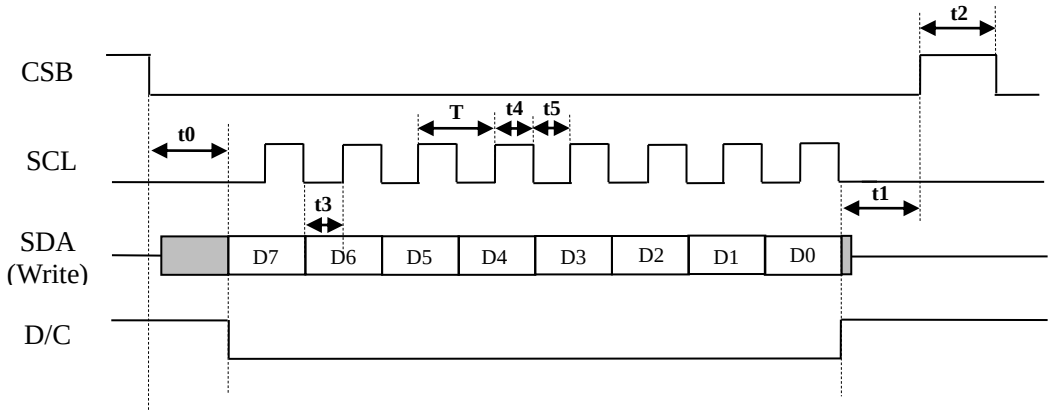


9.3.4 EPD Panel

Unless otherwise specified: VSS = 0V, DISPVDD=1.8 to 5.5V, VCOMVDD=1.8 to 5.5V, Ta = -40 to 85 °C

Item	Symbol	Nominal	Unit	Variation from Nominal		
				Min	Max	Unit
Panel Timing unit	T	(MDCDISPCLKDIV.CLKDIV[7:0] + 1) x t_{SYSCLK} *1	ns			
CSB fall, SDA low-impedance setup before first data, D/C falling edge	t0	MDCDISPPRM21.TIM2[7:0] + 1	T	0	9	ns
CSB hold after last data, D/C rising edge, SDA high-impedance	t1	MDCDISPPRM43.TIM3[7:0] + 1	T	-7	7	ns
CSB high pulse width between bytes	t2	MDCDISPPRM21.TIM1[7:0] + 1	T	-7	0	ns
Write data setup before SCL rising edge	t3	((MDCDISPCLKDIV.CLKDIV[7:0]+1) / 2) x t_{SYSCLK} *1	ns	0	9	ns
SCL high time	t4	((MDCDISPCLKDIV.CLKDIV[7:0] / 2) + 1) x t_{SYSCLK} *1	ns	-10	0	ns
SCL low time	t5	((MDCDISPCLKDIV.CLKDIV[7:0]+1) / 2) x t_{SYSCLK} *1	ns	0	10	ns

*1 t_{SYSCLK}: System clock period [ns]



9.4 QSPI Characteristics

Master Mode

Unless otherwise specified: Master mode, VSS = 0 V, Ta = -40 to 85 °C

Item	Symbol	Condition	VDD	Min.	Typ.	Max.	Unit
QSPICLK cycle time	t_{SCYC}		3.0 to 5.5V	100	-	-	ns
			1.8 to 3.0V	160	-	-	ns
QSPICLK High pulse width	t_{SCKH}		3.0 to 5.5V	40	-	-	ns
			1.8 to 3.0V	64	-	-	ns
QSPICLK Low pulse width	t_{SCKL}		3.0 to 5.5V	40	-	-	ns
			1.8 to 3.0V	64	-	-	ns
QSDIO[3:0] setup time	t_{SDS}		3.0 to 5.5V	30	-	-	ns
			1.8 to 3.0V	50	-	-	ns
QSDIO[3:0] hold time	t_{SDH}		3.0 to 5.5V	10	-	-	ns
			1.8 to 3.0V	10	-	-	ns
QSDIO[3:0] output delay time	t_{SDO}	C_L = 15 pF *1	3.0 to 5.5V	-	-	20	ns
			1.8 to 3.0V	-	-	30	ns

*1 C_L = Pin load

Slave Mode

Unless otherwise specified: Slave mode, VSS = 0 V, Ta = -40 to 85 °C

Item	Symbol	Condition	VDD	Min.	Typ.	Max.	Unit
QSPICLK cycle time	t_{SCYC}		3.0 to 5.5V	120	-	-	ns
			1.8 to 3.0V	160	-	-	ns
QSPICLK High pulse width	t_{SCKH}		3.0 to 5.5V	48	-	-	ns
			1.8 to 3.0V	64	-	-	ns
QSPICLK Low pulse width	t_{SCKL}		3.0 to 5.5V	48	-	-	ns
			1.8 to 3.0V	64	-	-	ns
QSDIO[3:0] setup time	t_{SDS}		3.0 to 5.5V	15	-	-	ns
			1.8 to 3.0V	20	-	-	ns
QSDIO[3:0] hold time	t_{SDH}		3.0 to 5.5V	10	-	-	ns
			1.8 to 3.0V	10	-	-	ns
QSDIO[3:0] output delay time	t_{SDO}	C_L = 15 pF *1	3.0 to 5.5V	-	-	43	ns
			1.8 to 3.0V	-	-	56	ns
#QSPISS setup time	t_{SSS}		3.0 to 5.5V	10	-	-	ns
			1.8 to 3.0V	10	-	-	ns
#QSPISS High pulse width	t_{SSH}		3.0 to 5.5V	48	-	-	ns
			1.8 to 3.0V	64	-	-	ns
QSDIO[3:0] output start time	t_{SDD}	C_L = 15 pF *1	3.0 to 5.5V	-	-	50	ns
			1.8 to 3.0V	-	-	60	ns
QSDIO[3:0] output stop time	t_{SDZ}	C_L = 15 pF *1	3.0 to 5.5V	-	-	50	ns
			1.8 to 3.0V	-	-	60	ns

*1 C_L = Pin load

9.5 SPI Characteristics

Master Mode

Unless otherwise specified: Master mode, VSS = 0V, Ta = -40 to 85 °C

Item	Symbol	Condition	VDD	Min.	Typ.	Max.	Unit
SPICLK cycle time	t_{SCYC}		3.0 to 5.5V	100	-	-	ns
			1.8 to 3.0V	160	-	-	ns
SPICLK High pulse width	t_{SCKH}		3.0 to 5.5V	40	-	-	ns
			1.8 to 3.0V	64	-	-	ns
SPICLK Low pulse width	t_{SCKL}		3.0 to 5.5V	40	-	-	ns
			1.8 to 3.0V	64	-	-	ns
SDI setup time	t_{SDS}		3.0 to 5.5V	32	-	-	ns
			1.8 to 3.0V	55	-	-	ns
SDI hold time	t_{SDH}		3.0 to 5.5V	10	-	-	ns
			1.8 to 3.0V	10	-	-	ns
SDO output delay time	t_{SDO}	C_L = 15 pF *1	3.0 to 5.5V	-	-	20	ns
			1.8 to 3.0V	-	-	30	ns

*1 C_L = Pin load

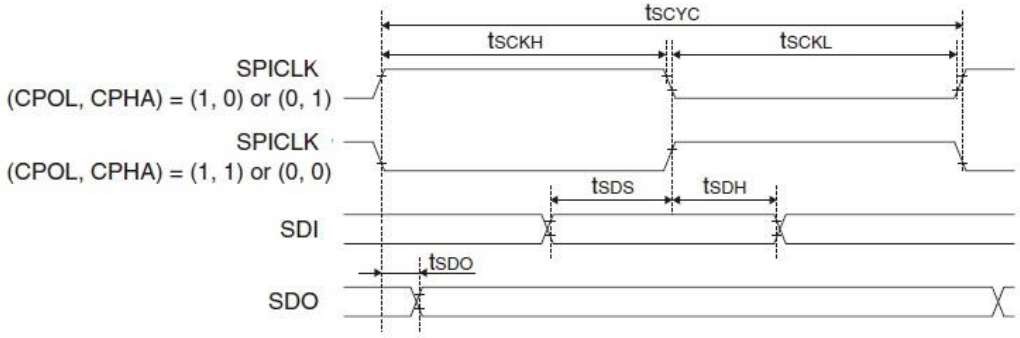
Slave Mode

Unless otherwise specified: Slave mode, VSS = 0 V, Ta = -40 to 85 °C

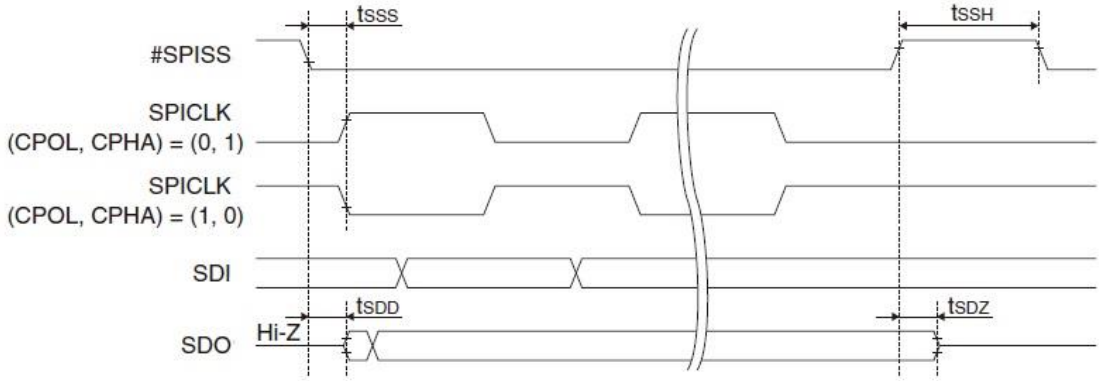
Item	Symbol	Condition	VDD	Min.	Typ.	Max.	Unit
SPICLK cycle time	t_{SCYC}		3.0 to 5.5V	100	-	-	ns
			1.8 to 3.0V	160	-	-	ns
SPICLK High pulse width	t_{SCKH}		3.0 to 5.5V	40	-	-	ns
			1.8 to 3.0V	64	-	-	ns
SPICLK Low pulse width	t_{SCKL}		3.0 to 5.5V	40	-	-	ns
			1.8 to 3.0V	64	-	-	ns
SDI setup time	t_{SDS}		3.0 to 5.5V	10	-	-	ns
			1.8 to 3.0V	10	-	-	ns
SDI hold time	t_{SDH}		3.0 to 5.5V	10	-	-	ns
			1.8 to 3.0V	10	-	-	ns
SDO output delay time	t_{SDO}	C_L = 15 pF *1	3.0 to 5.5V	-	-	42	ns
			1.8 to 3.0V	-	-	60	ns
#SPISS setup time	t_{SSS}		3.0 to 5.5V	10	-	-	ns
			1.8 to 3.0V	10	-	-	ns
#SPISS High pulse width	t_{SSH}		3.0 to 5.5V	40	-	-	ns
			1.8 to 3.0V	64	-	-	ns
SDO output start time	t_{SDD}	C_L = 15 pF *1	3.0 to 5.5V	-	-	50	ns
			1.8 to 3.0V	-	-	60	ns
SDO output stop time	t_{SDZ}	C_L = 15 pF *1	3.0 to 5.5V	-	-	50	ns
			1.8 to 3.0V	-	-	60	ns

*1 C_L = Pin load

Master and slave modes



Slave mode



9.6 I2C Characteristics

Unless otherwise specified: VDD = 1.8 to 5.5 V, VSS = 0 V, Ta = -40 to 85 °C

Item	Symbol	Condition	Standard mode			Fast mode			Unit
			Min.	Typ.	Max.	Min.	Typ.	Max.	
SCL frequency	f_{SCL}		0	-	100	0	-	400	kHz
Hold time (repeated) START condition *	$t_{HD:STA}$		4.0	-	-	0.6	-	-	us
SCL Low pulse width	t_{LOW}		4.7	-	-	1.3	-	-	us
SCL High pulse width	t_{HIGH}		4.0	-	-	0.6	-	-	us
Repeated START condition setup time	$t_{SU:STA}$		4.7	-	-	0.6	-	-	us
Data hold time	$t_{HD:DAT}$		0	-	-	0	-	-	us
Data setup time	$t_{SU:DAT}$		250	-	-	100	-	-	us
SDA, SCL rise time	t_r		-	-	1,000	-	-	300	us
SDA, SCL fall time	t_f		-	-	300	-	-	300	us
STOP condition setup time	$t_{SU:STO}$		4.0	-	-	0.6	-	-	us
Bus free time	t_{BUF}		4.7	-	-	1.3	-	-	us

* After this period, the first clock pulse is generated.

10. Registers

Registers for the peripherals in the S1D13C00 are located at base address 0x4000_0000 in the Memory Map (address range of 0x4000_0000 to 0x4000_3FFF). Registers are either asynchronous or synchronous. All of the registers, except for the Asynchronous System Registers, are synchronous and require the IOSC oscillator to be turned on to access them. The Asynchronous System Registers (see table below) are accessible from the Host and used to control IOSC. Table 10.1 shows the Registers Set.

The CLGOSC1 and PORTCLK registers have write-protection which is controlled by the SYSPROT register. Please refer to the SYSPROT register for details on how to unprotect before writing to these registers.

All bits marked n/a have no effect when written and always read back 0. All bits marked reserved should not be written.

Table 10.1 Registers Set

10.1 Asynchronous System Registers

10.1.1 System Control

REG[30E0h] SYSCTRL Register							
Default = XX01h							R/W
SOFTRST	n/a	IOSCAJ[5:0]					
15	14	13	12	11	10	9	8
n/a		HCKINV	IRQPOL	IOSCSTA	IOSCFQ bits 1-0		IOSCEN
7	6	5	4	3	2	1	0

bit 15 Soft Reset
This bit performs a software reset of the chip. It has the same effect as a hardware reset. All of the chip's circuits are reset except the Host Interface circuit.
Writing 1 will reset the chip. Writing 0 has no effect.
This bit is self-clearing and there is no need to write 0 after writing 1.

bits 13:8 IOSC Trim Adjust Value
These bits set the frequency trimming value for the IOSC internal oscillator circuit. This setting affects the oscillation frequencies (8MHz to 20MHz).

IOSCAJ[5:0]	**IOSC oscillation frequency (20/16/12/8MHz)**
0x00	Low
.	.
.	.
0x3F	High

Note: The initial value of IOSCAJ[5:0] is adjusted so that the IOSC oscillator circuit characteristics described in the "AC Characteristics" chapter can be guaranteed. Be aware that the frequency characteristics may not be satisfied when this setting is altered. When altering this setting, always make sure that the IOSC oscillator circuit is inactive.

bit 5 HCK Invert
When this bit = 0, HCK is normal (default).
When this bit = 1, HCK is inverted.

bit 4 HIFRQ Polarity
When this bit = 0, HIFRQ polarity is active high (default).
When this bit = 1, HIFRQ polarity is active low.

bit 3 IOSC Stable (read-only)
When this bit = 0, IOSC is not stable.
When this bit = 1, IOSC is stable.

bits 2:1 IOSC Frequency Select bits [1:0]

Table 10.2 IOSC Frequency Setting

IOSC Frequency Select bits [1:0]	Frequency
00	8MHz (default)
01	12MHz
10	16MHz
11	20MHz

bit 0 IOSC Enable
When this bit = 0, IOSC is disabled.
When this bit = 1, IOSC is enabled (default).

10.1.2 System Interrupts Status

REG[30E4h] SYSINTS Register (Read-Only)							
Default = 0000h							RO
n/a				RTCINT	SNDINT	DMAINT	REMCINT
15	14	13	12	11	10	9	8
PORTINT	CLGINT	I2CINT	MDCINT	T16_1INT	SPIINT	T16_2INT	QSPIINT
7	6	5	4	3	2	1	0

bit 11 RTC Interrupt Status
When this bit = 0, the RTC interrupt is not active.
When this bit = 1, the RTC interrupt is active.

bit 10 Sound Generator (SND) Interrupt Status
When this bit = 0, the SND interrupt not active.
When this bit = 1, the SND interrupt is active.

bit 9 DMA Controller Interrupt Status
When this bit = 0, the DMAC interrupt not active.
When this bit = 1, the DMAC interrupt is active.

bit 8 IR Remote (REMC) Interrupt Status
When this bit = 0, the REMC interrupt is not active.
When this bit = 1, the REMC interrupt is active.

bit 7 GPIO Ports Interrupt Status
When this bit = 0, the GPIO ports interrupt is not active.
When this bit = 1, the GPIO ports interrupt is active.

bit 6 Clock Generator (CLG) Interrupt Status
When this bit = 0, the CLG interrupt is not active.
When this bit = 1, the CLG interrupt is active.

bit 5 I2C Interrupt Status
When this bit = 0, the I2C interrupt is not active.
When this bit = 1, the I2C interrupt is active.

bit 4 Memory Display Controller (MDC) Interrupt Status
When this bit = 0, the MDC interrupt is not active.
When this bit = 1, the MDC interrupt is active.

bit 3 16-bit Timer Channel 1 (SPI) Interrupt Status
When this bit = 0, the 16-bit Timer Channel 1 (SPI) interrupt is not active.
When this bit = 1, the 16-bit Timer Channel 1 (SPI) interrupt is active.

bit 2 SPI Interrupt Status
When this bit = 0, the SPI interrupt is not active.
When this bit = 1, the SPI interrupt is active.

bit 1 16-bit Timer Channel 2 (QSPI) Interrupt Status
When this bit = 0, the 16-bit Timer Channel 2 (QSPI) interrupt is not active.
When this bit = 1, the 16-bit Timer Channel 2 (QSPI) interrupt is active.

bit 0 QSPI Interrupt Status
When this bit = 0, the QSPI interrupt is not active.
When this bit = 1, the QSPI interrupt is active.

10.2 Clock Generator (CLG) Registers

10.2.1 System Protection

REG[0000h] SYSPROT Register
Default = 0000h R/W

PROT bits 15-8							
15	14	13	12	11	10	9	8
PROT bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 PROT bits [15:0]
These bits protect some control registers against writing. Write 0x0096 to this register to disable protection. Registers with R/WP designation are protected from writes.

When this register = 0x0096, System protection is disabled.
When this register = any value other than 0x0096, System protection is enabled (default).

10.2.2 Oscillation Control

REG[0042h] CLGOSC Register
Default = 0000h R/W

n/a							
15	14	13	12	11	10	9	8
n/a						OSC1EN	n/a
7	6	5	4	3	2	1	0

bit 1 OSC1 enable
When this bit = 0, OSC1 is disabled (default).
When this bit = 1, OSC1 is enabled.

10.2.3 OSC1 Control

REG[0046h] CLGOSC1 Register
Default = 5092h R/WP

n/a	OSDRB	OSDEN	OSC1BUP	OSC1SELCR	CGI1 bits 2-0		
15	14	13	12	11	10	9	8
INV1B bits 1-0		INV1N bits 1-0		n/a		OSC1WT bits 1-0	
7	6	5	4	3	2	1	0

bit 14 OSDRB – OSC1 Restart After Stop Detected
This bit enables the OSC1 oscillator circuit restart function by the oscillation stop detector when OSC1 oscillation stop is detected.
When this bit = 0, OSC1 restart after stop detected is disabled.
When this bit = 1, OSC1 restart after stop detected is enabled (default: restart the OSC1 oscillator circuit when oscillation stop is detected).

bit 13 OSDEN – OSC1 stop detection
This bit controls the oscillation stop detector in the OSC1 oscillator circuit.
When this bit = 0, OSC1 oscillation stop detector is off (default).
When this bit = 1, OSC1 oscillation stop detector is on.

NOTE: Do not write 1 to the CLGOSC1.OSDEN bit before stabilized OSC1CLK is supplied. Furthermore, the CLGOSC1.OSDEN bit should be set to 0 when the CLGOSC.OSC1EN bit is set to 0.

bit 12 OSC1BUP – OSC1 Startup Booster
When this bit = 0, OSC1 startup booster is disabled.
When this bit = 1, OSC1 startup booster is enabled (default: activate booster operation at startup).

bit 11 OSC1SELCR – OSC1 Oscillator Select
When this bit = 0, OSC1 is external 32.768kHz crystal oscillator (default).
When this bit = 1, OSC1 is internal 32kHz CR oscillator.

bits 10:8 CGI1 bits [2:0] – OSC1 internal gate capacitance
These bits set the internal gate capacitance in the OSC1 oscillator circuit.
0x7 is the maximum capacitance value and 0x0 is the minimum capacitance value.

bits 7:6 INV1B bits [1:0] – OSC1 boost startup inverter gain
These bits set the oscillation inverter gain that will be applied at boost startup of the OSC1 oscillator circuit. 0x3 is the maximum gain and 0x0 is the minimum gain.

NOTE: The CLGOSC1.INV1B[1:0] bits must be set to a value equal to or larger than the CLGOSC1.INV1N[1:0] bits.

bits 5:4 INV1N bits [1:0] – OSC1 normal startup inverter gain
These bits set the oscillation inverter gain applied at normal operation of the OSC1 oscillator circuit. 0x3 is the maximum gain and 0x0 is the minimum gain.

bits 1:0 OSC1WT bits [1:0] – OSC1 stabilization wait
These bits set the oscillation stabilization waiting time for the OSC1 oscillator circuit.
0x3: 65,536 clocks
0x2: 16,384 clocks
0x1: 4,096 clocks
0x0: Reserved

10.2.4 CLG Interrupt Flag

REG[004Ch] CLGINTF Register
Default = 0000h R/W

n/a							
15	14	13	12	11	10	9	8
n/a		OSC1STPIF	n/a			OSC1STAIF	n/a
7	6	5	4	3	2	1	0

bit 5 OSC1 Oscillation Stop Interrupt
When this bit reads 0, no interrupt has occurred
When this bit reads 1, Oscillation Stop Interrupt has occurred
When this bit is written 0, there is no effect
When this bit is written 1, the Oscillation Stop Interrupt flag is cleared

bit 1 OSC1 Stabilization Waiting Completion Interrupt
When this bit reads 0, no interrupt has occurred
When this bit reads 1, Stabilization Waiting Completion Interrupt has occurred
When this bit is written 0, there is no effect
When this bit is written 1, the Stabilization Waiting Completion Interrupt flag is cleared

10.2.5 CLG Interrupt Enable

REG[004Eh] CLGINTE Register
Default = 0000h R/W

n/a							
15	14	13	12	11	10	9	8
n/a		OSC1STPIE	n/a			OSC1STAIE	n/a
7	6	5	4	3	2	1	0

bit 5 OSC1 Oscillation Stop Interrupt Enable
When this bit = 0, Oscillation Stop Interrupt is disabled
When this bit = 1, Oscillation Stop Interrupt is enabled

bit 1 OSC1 Stabilization Waiting Completion Interrupt Enable
When this bit = 0, Stabilization Waiting Completion Interrupt is disabled
When this bit = 1, Stabilization Waiting Completion Interrupt is enabled

10.2.6 CLG FOUT Control

REG[0050h] CLGFOUT Register
Default = 0000h R/W

n/a							
15	14	13	12	11	10	9	8
n/a	FOUTDIV bits 2-0			FOUTSRC bits 1-0		n/a	FOUTEN
7	6	5	4	3	2	1	0

bits 6:4 FOUTDIV bits [2:0] – FOUT Clock Division Ratio
These bits set the FOUT clock division ratio.

bits 3:2 FOUTSRC bits [1:0] – FOUT Clock Source
These bits select the FOUT clock source.

Table 10.3 FOUT Clock Source and Division Ratio Settings

CLGFOUT. FOUTDIV[2:0] bits	CLGFOUT.FOUTSRC[1:0] bits			
	0x0	0x1	0x2	0x3
	IOSCCLK	OSC1CLK	Reserved	SYSCLK
0x7	1/128	1/32,768	-	Reserved
0x6	1/64	1/4,096	-	Reserved
0x5	1/32	1/1,024	-	Reserved
0x4	1/16	1/256	-	Reserved
0x3	1/8	1/8	-	Reserved
0x2	1/4	1/4	-	Reserved
0x1	1/2	1/2	-	Reserved
0x0	1/1	1/1	-	1/1

bit 0 FOUTEN – FOUT Enable
This bit controls the FOUT clock external output.
When this bit = 0, FOUT output is disabled
When this bit = 1, FOUT output is enabled

10.2.7 OSC1 Internal Oscillator Trimming

REG[0054h] CLGOSC1TRM Register
Default = 00xxh R/WP

n/a							
15	14	13	12	11	10	9	8
n/a		OSC1SAJ bits 5-0					
7	6	5	4	3	2	1	0

bits 5:0 OSC1SAJ bits [5:0] – OSC1 Internal Oscillator Frequency Trimming Adjust
These bits set the frequency trimming value for the OSC1 internal 32kHz oscillator circuit. This setting does not affect the OSC1 crystal oscillation frequency.

OSC1SAJ[5:0]	**OSC1 oscillation frequency (32kHz)**
0x00	Low
.	.
.	.
0x3F	High

Note: The initial value of OSC1SAJ[5:0] is adjusted so that the OSC1 internal oscillator circuit characteristics described in the "AC Characteristics" chapter can be guaranteed. Be aware that the frequency characteristics may not be satisfied when this setting is altered. When altering this setting, always make sure that the OSC1 internal oscillator circuit is inactive.

10.3 Real-Time Clock (RTC) Registers

10.3.1 RTC Control (Low Byte)

REG[00C0h] RTCCTLL Register
Default = 00h R/W

n/a	RTCBSY	RTCHLD	RTC24H	n/a	RTCADJ	RTCRST	RTCRUN
7	6	5	4	3	2	1	0

bit 6 RTCBSY (read-only) – RTC Busy
This bit indicates whether the counter is performing count-up operation or not.
When this bit reads 0, the RTC is idle (ready to rewrite real-time clock counter)
When this bit reads 1, the RTC performing a count-up operation

This bit goes to 1 when performing 1-second count-up, +1 second correction, or 30-second correction. It retains 1 for 1/256 second and then reverts to 0.

bit 5 RTCHLD – RTC Halt
This bit halts the count-up operation of the real-time clock counter.
When this bit = 0, the RTC is in normal operation
When this bit = 1, real-time clock counter count-up operation is halted

Writing 1 to this bit halts the count-up operation of the RTC counter. This makes it possible to read the counter value correctly without changing the counter. Write 0 to this bit to resume count-up operation immediately after the counter has been read. Depending on these operation timings, the +1 second correction may be executed after the count-up operation resumes.

NOTE: When the RTCACTLH.RTCTRMBSY bit = 1, the RTCACTLL.RTCHLD bit cannot be rewritten to 1 (is fixed at 0).

bit 4 RTC24H – RTC 24-Hour Mode
This bit sets the hour counter to 24H mode or 12H mode.
When this bit = 0, the hour counter is set to 12H mode
When this bit = 1, the hour counter is set to 24H mode

This selection changes the count range of the hour counter. Note, however, that the counter value is not updated automatically; therefore, it must be programmed again.

NOTE: Be sure to avoid writing to this bit when the RTCACTLL.RTCRUN bit = 1.

bit 2 RTCADJ – RTC 30-Second Correction Time Adjustment
This bit executes the 30-second correction time adjustment function.
When this bit is written 1, Execute 30-second correction
When this bit is written 0, there is no effect
When this bit reads 1, the 30-second correction is executing
When this bit reads 0, the 30-second correction has finished (Normal operation)

Writing 1 to this bit executes 30-second correction and an enabled interrupt occurs even if the RTCACTLL.RTCRUN bit = 0. The correction takes up to 2/256 seconds. The RTCACTLL.RTCADJ bit is automatically cleared to 0 when the correction has finished.

NOTES:
Be sure to avoid writing to this bit when the RTCACTLL.RTCBSY bit = 1.
Do not write 1 to this bit again while the RTCACTLL.RTCADJ bit = 1.

bit 1 RTCRST – RTC Reset
This bit resets the 1 Hz counter, the RTCACTLL.RTCADJ bit, and the RTCACTLL.RTCHLD bit.
When this bit is written 1, the 1 Hz counter, the RTCACTLL.RTCADJ bit and the RTCACTLL.RTCHLD bit are all reset
When this bit is written 0, there is no effect
When this bit reads 1, the reset is being executed
When this bit reads 0, the reset has finished. (Normal operation)

This bit is automatically cleared to 0 after reset has finished.

bit 0 RTCRUN – RTC Run
This bit starts/stops the real-time clock counter.
When this bit is written 1, the RTC is started
When this bit is written 0, the RTC is stopped
When this bit reads 1, the RTC is running
When this bit reads 0, the RTC is idle

When the real-time clock counter stops counting by writing 0 to this bit, the counter retains the value when it stopped. Writing 1 to this bit again resumes counting from the value retained.

10.3.2 RTC Control (High Byte)

REG[00C1h] RTCCTLH Register
Default = 00h R/W

RTCTRMBSY (RO)	RTCTRM bits 6-0						
7	6	5	4	3	2	1	0

bit 7 RTCTRMBSY (read-only) – RTC Trimming Busy
This bit indicates whether the theoretical regulation is currently executed or not.

When this bit = 0, theoretical regulation has finished (or is not executing)
When this bit = 1, theoretical regulation is executing

This bit goes 1 when a value is written to the RTCACTLH.RTCTRM[6:0] bits. The theoretical regulation takes up to 1 second for execution. This bit reverts to 0 automatically after the theoretical regulation has finished execution.

bits 6:0 RTCTRM bits [6:0] – RTC Trim Value for Theoretical Regulation
Write the correction value for adjusting the 1 Hz frequency to these bits to execute theoretical regulation.

NOTES:
When the RTCACTLH.RTCTRMBSY bit = 1, the RTCACTLH.RTCTRM[6:0] bits cannot be rewritten.
Writing 0x00 to the RTCACTLH.RTCTRM[6:0] bits sets the RTCACTLH.RTCTRMBSY bit to 1 as well. However, no correcting operation is performed.

10.3.3 RTC Seconds Alarm

REG[00C2h] RTCALM1 Register
Default = 0000h R/W

n/a	RTCSHA bits 2-0			RTCSLA bits 3-0			
15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0

bits 14:12 RTCSHA bits [2:0]
bits 11:8 RTCSLA bits [3:0]
The RTCAALM1.RTCSHA[2:0] bits and the RTCAALM1.RTCSLA[3:0] bits set the 10-second digit and 1-second digit of the alarm time, respectively. A value within 0 to 59 seconds can be set in BCD code.

10.3.4 RTC Hour/Minute Alarm

REG[00C4h] RTCALM2 Register
Default = 0000h R/W

n/a	RTCPA	RTCHHA bits 1-0		RTCHLA bits 3-0			
15	14	13	12	11	10	9	8
n/a	RTCMIHA bits 2-0			RTCMILA bits 3-0			
7	6	5	4	3	2	1	0

bit 14 RTCPA
This bit sets A.M. or P.M. of the alarm time in 12H mode (RTCACTLL.RTC24H bit = 0).
When this bit = 0, P.M. is set
When this bit = 1, A.M. is set

This setting is ineffective in 24H mode (RTCACTLL.RTC24H bit = 1).

bits 13:12 RTCHHA bits [1:0]
bits 11:8 RTCHLA bits [3:0]
The RTCAALM2.RTCHHA[1:0] bits and the RTCAALM2.RTCHLA[3:0] bits set the 10-hour digit and 1-hour digit of the alarm time, respectively. A value within 1 to 12 o'clock in 12H mode or 0 to 23 in 24H mode can be set in BCD code.

bits 6:4 RTCMIHA bits [2:0]
bits 3:0 RTCMILA bits [3:0]
The RTCAALM2.RTCMIHA[2:0] bits and the RTCAALM2.RTCMILA[3:0] bits set the

10-minute digit and 1-minute digit of the alarm time, respectively. A value within 0 to 59 minutes can be set in BCD code.

10.3.5 RTC Stopwatch Control

REG[00C6h] RTCSWCTL Register							
Default = 0000h							R/W
BCD10 bits 3-0				BCD100 bits 3-0			
15	14	13	12	11	10	9	8
n/a			SWRST	n/a			SWRUN
7	6	5	4	3	2	1	0

bits 15:12 BCD10 bits [3:0] (read-only)

bits 11:8 BCD100 bits [3:0] (read-only)

The 1/10-second and 1/100-second digits of the stopwatch counter can be read as a BCD code from the RTCASWCTL.BCD10[3:0] bits and the RTCASWCTL.BCD100[3:0] bits, respectively.

NOTE: The counter value may not be read correctly while the stopwatch counter is running. The RTCASWCTL.BCD10[3:0]/BCD100[3:0] bits must be read twice and assume the counter value was read successfully if the two read results are the same.

bit 4 SWRST – Stopwatch Reset

This bit resets the stopwatch counter to 0x00.
When this bit is written 1, Reset
When this bit is written 0, Ineffective
This bit always reads 0

When the stopwatch counter in running status is reset, it continues counting from count 0x00. The stopwatch counter retains 0x00 if it is reset in idle status.

bit 0 SWRUN – Stopwatch Run

This bit starts/stops the stopwatch counter.
When this bit is written 1, the stopwatch is started
When this bit is written 0, the stopwatch is stopped
When this bit reads 1, the stopwatch is running
When this bit reads 0, the stopwatch is idle

When the stopwatch counter stops counting by writing 0 to this bit, the counter retains the value when it stopped. Writing 1 to this bit again resumes counting from the value retained.

NOTE: The stopwatch counter stops in sync with the stopwatch clock after 0 is written to the RTCASWCTL.SWRUN bit. Therefore, the counter value may be incremented (+1) from the value at writing 0.

10.3.6 RTC Seconds

REG[00C8h] RTCSEC Register							
Default = 0000h							R/W
n/a	RTCSH bits 2-0			RTCSL bits 3-0			
15	14	13	12	11	10	9	8
RTC1HZ	RTC2HZ	RTC4HZ	RTC8HZ	RTC16HZ	RTC32HZ	RTC64HZ	RTC128HZ
7	6	5	4	3	2	1	0

bits 14:12 RTCSH bits [2:0]

bits 11:8 RTCSL bits [3:0]

The RTCASEC.RTCSH[2:0] bits and the RTCASEC.RTCSL[3:0] bits are used to set and read

the 10-second digit and the 1-second digit of the second counter, respectively. The setting/read values are a BCD code within the range from 0 to 59.

NOTE: Be sure to avoid writing to the RTCASEC.RTCSH[2:0]/RTCSL[3:0] bits while the RTCACTLL.RTCBSY bit = 1.

bit 7 RTC1HZ (read-only)
bit 6 RTC2HZ (read-only)
bit 5 RTC4HZ (read-only)
bit 4 RTC8HZ (read-only)
bit 3 RTC16HZ (read-only)
bit 2 RTC32HZ (read-only)
bit 1 RTC64HZ (read-only)
bit 0 RTC128HZ (read-only)

1 Hz counter data can be read from these bits.
The following shows the correspondence between the bit and frequency:
RTCASEC.RTC1HZ bit: 1 Hz
RTCASEC.RTC2HZ bit: 2 Hz
RTCASEC.RTC4HZ bit: 4 Hz
RTCASEC.RTC8HZ bit: 8 Hz
RTCASEC.RTC16HZ bit: 16 Hz
RTCASEC.RTC32HZ bit: 32 Hz
RTCASEC.RTC64HZ bit: 64 Hz
RTCASEC.RTC128HZ bit: 128 Hz

NOTE: The counter value may not be read correctly while the 1 Hz counter is running. These bits must be read twice and assume the counter value was read successfully if the two read results are the same.

10.3.7 RTC Hour/Minute

REG[00CAh] RTCHUR Register
Default = 1200h R/W

n/a	RTCAP	RTCHH bits 1-0		RTCHL bits 3-0			
15	14	13	12	11	10	9	8
n/a	RTCMIH bits 2:0			RTCMIL bits 3-0			
7	6	5	4	3	2	1	0

bit 14 RTCAP – RTC AM/PM Setting

This bit is used to set and read A.M. or P.M. data in 12H mode (RTCACTLL.RTC24H bit = 0).
When this bit = 0, A.M. is selected
When this bit = 1, P.M. is selected

In 24H mode (RTCACTLL.RTC24H bit = 1), this bit is fixed at 0 and writing 1 is ignored. However, if the RTCAHUR.RTCAP bit = 1 when changed to 24H mode, it goes 0 at the next count-up timing of the hour counter.

bits 13:12 RTCHH bits [1:0]
bits 11:8 RTCHL bits [3:0]

The RTCAHUR.RTCHH[1:0] bits and the RTCAHUR.RTCHL[3:0] bits are used to set and read the 10-hour digit and the 1-hour digit of the hour counter, respectively. The setting/read values are a BCD code within the range from 1 to 12 in 12H mode or 0 to 23 in 24H mode.

NOTE: Be sure to avoid writing to the RTCAHUR.RTCHH[1:0]/RTCHL[3:0] bits while the RTCACTLL.RTCBSY bit = 1.

bits 6:4 RTCMIH bits[2:0]
bits 3:0 RTCMIL bits [3:0]
The RTCAHUR.RTCMIH[2:0] bits and the RTCAHUR.RTCMIL[3:0] bits are used to set and read the 10-minute digit and the 1-minute digit of the minute counter, respectively. The setting/read values are a BCD code within the range from 0 to 59.

NOTE: Be sure to avoid writing to the RTCAHUR.RTCMIH[2:0]/RTCMIL[3:0] bits while the RTCACTLL.RTCBSY bit = 1.

10.3.8 RTC Month/Day

REG[00CCh] RTCMON Register
Default = 0101h R/W

n/a			RTCMOH	RTCMOL bits 3-0			
15	14	13	12	11	10	9	8
n/a		RTCDH bits 1-0		RTCDL bits 3-0			
7	6	5	4	3	2	1	0

bit 12 RTCMOH
bits 11:8 RTCMOL bits [3:0]
The RTCAMON.RTCMOH bit and the RTCAMON.RTCMOL[3:0] bits are used to set and read the 10-month digit and the 1-month digit of the month counter, respectively. The setting/read values are a BCD code within the range from 1 to 12.

NOTE: Be sure to avoid writing to the RTCAMON.RTCMOH/RTCMOL[3:0] bits while the RTCACTLL.RTCBSY bit = 1.

bits 5:4 RTCDH bits [1:0]
bits 11:8 RTCDL bits [3:0]
The RTCAMON.RTCDH[1:0] bits and the RTCAMON.RTCDL[3:0] bits are used to set and read the 10-day digit and the 1-day digit of the day counter, respectively. The setting/read values are a BCD code within the range from 1 to 31 (to 28 for February in a common year, to 29 for February in a leap year, or to 30 for April/June/September/November).

NOTE: Be sure to avoid writing to the RTCAMON.RTCDH[1:0]/RTCDL[3:0] bits while the RTCACTLL.RTCBSY bit = 1.

10.3.9 RTC Year/Week

REG[00CEh] RTCYAR Register
Default = 0000h R/W

n/a					RTCWK bits 2-0		
15	14	13	12	11	10	9	8
RTCYH bits 3-0				RTCYL bits 3-0			
7	6	5	4	3	2	1	0

bits 10:8 RTCWK[2:0]
These bits are used to set and read day of the week.
0x0: Sunday
0x1: Monday
0x2: Tuesday
0x3: Wednesday
0x4: Thursday
0x5: Friday
0x6: Saturday

bits 7:4 RTCYH bits [3:0]
bits 3:0 RTCYL bits [3:0]
The RTCAYAR.RTCYH[3:0] bits and the RTCAYAR.RTCYL[3:0] bits are used to set and read the 10-year digit and the 1-year digit of the year counter, respectively. The setting/read values are a BCD code within the range from 0 to 99.

NOTE: Be sure to avoid writing to the RTCAYAR.RTCYH[3:0]/RTCYL[3:0] bits while the RTCACTLL.RTCBSY bit = 1.

10.3.10 RTC Interrupt Flag

REG[00D0h] RTCINTF Register
Default = 0000h R/W

RTFTRMIF	SW1IF	SW10IF	SW100IF	n/a			ALARMIF
15	14	13	12	11	10	9	8
T1DAYIF	T1HURIF	T1MINIF	T1SECIF	T1_2SECIF	T1_4SECIF	T1_8SECIF	T1_32SECIF
7	6	5	4	3	2	1	0

bit 15 RTCTRMIF
bit 14 SW1IF
bit 13 SW10IF
bit 12 SW100IF
These bits indicate the real-time clock interrupt cause occurrence status.
When a bit reads 1, an interrupt has occurred
When a bit reads 0, no interrupt has occurred
When a bit is written 1, the flag is cleared
When a bit is written 0, there is no effect

The following shows the correspondence between the bit and interrupt:
RTCAINTF.RTCTRMIF bit: Theoretical regulation completion interrupt
RTCAINTF.SW1IF bit: Stopwatch 1 Hz interrupt
RTCAINTF.SW10IF bit: Stopwatch 10 Hz interrupt
RTCAINTF.SW100IF bit: Stopwatch 100 Hz interrupt

bit 8 ALARMIF
bit 7 T1DAYIF
bit 6 T1HURIF
bit 5 T1MINIF
bit 4 T1SECIF
bit 3 T1_2SECIF
bit 2 T1_4SECIF
bit 1 T1_8SECIF
bit 0 T1_32SECIF
These bits indicate the real-time clock interrupt cause occurrence status.
When a bit reads 1, an interrupt has occurred
When a bit reads 0, no interrupt has occurred
When a bit is written 1, the flag is cleared
When a bit is written 0, there is no effect

The following shows the correspondence between the bit and interrupt:
RTCAINTF. ALARMIF bit: Alarm interrupt
RTCAINTF.T1DAYIF bit: 1-day interrupt
RTCAINTF.T1HURIF bit: 1-hour interrupt
RTCAINTF.T1MINIF bit: 1-minute interrupt
RTCAINTF.T1SECIF bit: 1-second interrupt
RTCAINTF.T1_2SECIF bit: 1/2-second interrupt

RTCAINTF.T1_4SECIF bit: 1/4-second interrupt
RTCAINTF.T1_8SECIF bit: 1/8-second interrupt
RTCAINTF.T1_32SECIF bit: 1/32-second interrupt

10.3.11 RTC Interrupt Enable

REG[00D2h] RTCINTE Register							
Default = 0000h							R/W
RTCTRMIE	SW1IE	SW10IE	SW100IE	n/a			ALARMIE
15	14	13	12	11	10	9	8
T1DAYIE	T1HURIE	T1MINIE	T1SECIE	T1_2SECIE	T1_4SECIE	T1_8SECIE	T1_32SECIE
7	6	5	4	3	2	1	0

bit 15 RTCTRMIE
bit 14 SW1IE
bit 13 SW10IE
bit 12 SW100IE
These bits enable real-time clock interrupts.
When a bit = 1, the corresponding interrupt is enabled
When a bit = 0, the corresponding interrupt is disabled

The following shows the correspondence between the bit and interrupt:
RTCAINTE.RTCTRMIE bit: Theoretical regulation completion interrupt
RTCAINTE.SW1IE bit: Stopwatch 1 Hz interrupt
RTCAINTE.SW10IE bit: Stopwatch 10 Hz interrupt
RTCAINTE.SW100IE bit: Stopwatch 100 Hz interrupt

bit 8 ALARMIE
bit 7 T1DAYIE
bit 6 T1HURIE
bit 5 T1MINIE
bit 4 T1SECIE
bit 3 T1_2SECIE
bit 2 T1_4SECIE
bit 1 T1_8SECIE
bit 0 T1_32SECIE
These bits enable real-time clock interrupts.
When a bit = 1, the corresponding interrupt is enabled
When a bit = 0, the corresponding interrupt is disabled

The following shows the correspondence between the bit and interrupt:
RTCAINTE.ALARMIE bit: Alarm interrupt
RTCAINTE.T1DAYIE bit: 1-day interrupt
RTCAINTE.T1HURIE bit: 1-hour interrupt
RTCAINTE.T1MINIE bit: 1-minute interrupt
RTCAINTE.T1SECIE bit: 1-second interrupt
RTCAINTE.T1_2SECIE bit: 1/2-second interrupt
RTCAINTE.T1_4SECIE bit: 1/4-second interrupt
RTCAINTE.T1_8SECIE bit: 1/8-second interrupt
RTCAINTE.T1_32SECIE bit: 1/32-second interrupt

10.4 GPIO Port Registers

10.4.1 Port P0 Data

REG[0200h] PORTP0DAT Register							
Default = 0000h							R/W
P0OUT[7] 15	P0OUT[6] 14	P0OUT[5] 13	P0OUT[4] 12	P0OUT[3] 11	P0OUT[2] 10	P0OUT[1] 9	P0OUT[0] 8
P0IN[7] 7	P0IN[6] 6	P0IN[5] 5	P0IN[4] 4	P0IN[3] 3	P0IN[2] 2	P0IN[1] 1	P0IN[0] 0

bits 15:8 P0OUT[7:0]
These bits are used to set data to be output from the GPIO port pins.
When a bit = 1, output a high level from the corresponding port pin
When a bit = 0, output a low level from the corresponding port pin

When output is enabled, the port pin outputs the data set here. Although data can be written when output is disabled, it does not affect the pin status. These bits do not affect the outputs when the port is used as a peripheral I/O function.

bits 7:0 P0IN[7:0] (read-only)
The GPIO port pin status can be read out from these bits.
When a bit reads 1, the corresponding port pin = high level
When a bit reads 0, the corresponding port pin = low level

The port pin status can be read out when input is enabled. When input is disabled, these bits are always read as 0. When the port is used for a peripheral I/O function, the input value cannot be read out from these bits.

10.4.2 Port P0 I/O Enable

REG[0202h] PORTP0IOEN Register							
Default = 0000h							R/W
P0IEN[7] 15	P0IEN[6] 14	P0IEN[5] 13	P0IEN[4] 12	P0IEN[3] 11	P0IEN[2] 10	P0IEN[1] 9	P0IEN[0] 8
P0OEN[7] 7	P0OEN[6] 6	P0OEN[5] 5	P0OEN[4] 4	P0OEN[3] 3	P0OEN[2] 2	P0OEN[1] 1	P0OEN[0] 0

bits 15:8 P0IEN[7:0]
These bits enable/disable the GPIO port input.
When a bit = 1, the corresponding GPIO port input is enabled (the port pin status is input.)
When a bit = 0, the corresponding GPIO port input is disabled (input data is fixed at 0.)

When both data output and data input are enabled, the pin output status controlled by this IC can be read. These bits do not affect the input control when the port is used as a peripheral I/O function.

bits 7:0 P0OEN[7:0]
These bits enable/disable the GPIO port output.
When a bit = 1, the corresponding GPIO port output is enabled (data is output from the port pin.)
When a bit = 0, the corresponding GPIO port output is disabled (the port is placed into Hi-Z.)

These bits do not affect the output control when the port is used as a peripheral I/O function.

10.4.3 Port P0 Pull-up/down Control

REG[0204h] PORTP0RCTL Register							
Default = 0000h							R/W
P0PDPU[7] 15	P0PDPU[6] 14	P0PDPU[5] 13	P0PDPU[4] 12	P0PDPU[3] 11	P0PDPU[2] 10	P0PDPU[1] 9	P0PDPU[0] 8
P0REN[7] 7	P0REN[6] 6	P0REN[5] 5	P0REN[4] 4	P0REN[3] 3	P0REN[2] 2	P0REN[1] 1	P0REN[0] 0

bits 15:8 P0PDPU[7:0]

These bits select either the pull-up resistor or the pull-down resistor when using a resistor built into the port.
When a bit = 1, the corresponding port pull-up resistor is selected
When a bit = 0, the corresponding port pull-down resistor is selected

The selected pull-up/down resistor is enabled when the PORTP0RCTL.P0REN[x] bit = 1.

bits 7:0 P0REN[7:0]

These bits enable/disable the port pull-up/down control.
When a bit = 1, the corresponding port pull-up/down is enabled (the built-in pull-up/down resistor is used)
When a bit = 0, the corresponding port pull-up/down is disabled (no pull-up/down control is performed)

Enabling this function pulls up or down the port when output is disabled (PORTP0IOEN.P0OEN[x] bit = 0). When output is enabled (PORTP0IOEN.P0OEN[x] bit = 1), the PORTP0RCTL.P0REN[x] bit setting is ineffective regardless of how the PORTP0IOEN.P0IE[x] bit is set and the port is not pulled up/down. These bits do not affect the pull-up/down control when the port is used as a peripheral I/O function.

10.4.4 Port P0 Interrupt Flag

REG[0206h] PORTP0INTF Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
P0IF[7] 7	P0IF[6] 6	P0IF[5] 5	P0IF[4] 4	P0IF[3] 3	P0IF[2] 2	P0IF[1] 1	P0IF[0] 0

bits 7:0 P0IF[7:0]

These bits indicate the port input interrupt cause occurrence status.
When a bit reads 1, an interrupt has occurred
When a bit reads 0, no interrupt has occurred
When a bit is written 1, the interrupt flag is cleared
When a bit is written 0, there is no effect

10.4.5 Port P0 Interrupt Control

REG[0208h] PORTP0INTCTL Register							
Default = 0000h							R/W
P0EDGE[7]	P0EDGE[6]	P0EDGE[5]	P0EDGE[4]	P0EDGE[3]	P0EDGE[2]	P0EDGE[1]	P0EDGE[0]
15	14	13	12	11	10	9	8
P0IE[7]	P0IE[6]	P0IE[5]	P0IE[4]	P0IE[3]	P0IE[2]	P0IE[1]	P0IE[0]
7	6	5	4	3	2	1	0

bits 15:8 P0EDGE[7:0]
These bits select the input signal edge to generate a port input interrupt.
When a bit = 1, the corresponding interrupt will occur at a falling edge
When a bit = 0, the corresponding interrupt will occur at a rising edge

bits 7:0 P0IE[7:0]
These bits enable port input interrupts.
When a bit = 1, the corresponding interrupt is enabled
When a bit = 0, the corresponding interrupt is disabled

NOTE: To prevent generating unnecessary interrupts, the corresponding interrupt flag should be cleared before enabling interrupts.

10.4.6 Port P0 Chattering Filter Enable

REG[020Ah] PORTP0CHATEN Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
P0CHATEN[7]	P0CHATEN[6]	P0CHATEN[5]	P0CHATEN[4]	P0CHATEN[3]	P0CHATEN[2]	P0CHATEN[1]	P0CHATEN[0]
7	6	5	4	3	2	1	0

bits 7:0 P0CHATEN[7:0]
These bits enable/disable the chattering filter function.
When a bit = 1, the corresponding chattering filter is enabled
When a bit = 0, the corresponding chattering filter is bypassed (disabled)

10.4.7 Port P0 Mode Select

REG[020Ch] PORTP0MODSEL Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
P0SEL[7]	P0SEL[6]	P0SEL[5]	P0SEL[4]	P0SEL[3]	P0SEL[2]	P0SEL[1]	P0SEL[0]
7	6	5	4	3	2	1	0

bits 7:0 P0SEL[7:0]
These bits select whether each port is used for the GPIO function or a peripheral I/O function.
When a bit = 1, the corresponding port uses peripheral I/O function
When a bit = 0, the corresponding port uses GPIO function

10.4.8 Port P0 Function Select

REG[020Eh] PORTP0FNCSEL Register								R/W
Default = 0000h								
P07MUX bits 1-0		P06MUX bits 1-0		P05MUX bits 1-0		P04MUX bits 1-0		
15	14	13	12	11	10	9	8	
P03MUX bits 1-0		P02MUX bits 1-0		P01MUX bits 1-0		P00MUX bits 1-0		
7	6	5	4	3	2	1	0	

bits 15:14 P07MUX bits [1:0]
bits 13:12 P06MUX bits [1:0]
bits 11:10 P05MUX bits [1:0]
bits 9:8 P04MUX bits [1:0]
bits 7:6 P03MUX bits [1:0]
bits 5:4 P02MUX bits [1:0]
bits 3:2 P01MUX bits [1:0]
bits 1:0 P00MUX bits [1:0]

These bits select the peripheral I/O function to be assigned to each port pin according to the following table:

Table 10.4 P0 Port Group Function Assignment

Port name	P0SEL[x] = 0	P0SEL[x] = 1							
		P0xMUX = 0x0		P0xMUX = 0x1		P0xMUX = 0x2		P0xMUX = 0x3	
	GPIO	Peripheral	Pin	Peripheral	Pin	Peripheral	Pin	Peripheral	Pin
P00	P00	SPI	#SPISS	-	-	-	-	–	–
P01	P01	SPI	SPICLK	-	-	-	-	–	–
P02	P02	SPI	SDI	-	-	-	-	–	–
P03	P03	SPI	SDO	-	-	-	-	–	–
P04	P04	I2C	SCL	-	-	-	-	–	–
P05	P05	I2C	SDA	-	-	-	-	–	–
P06	P06	REMC	REMO	-	-	-	-	–	–
P07	P07	REMC	CLPLS	-	-	-	-	–	–

10.4.9 Port P1 Data

REG[0210h] PORTP1DAT Register								R/W
Default = 0000h								
n/a						P1OUT[1]		P1OUT[0]
15	14	13	12	11	10	9		8
n/a						P1IN[1]		P1IN[0]
7	6	5	4	3	2	1		0

bits 9:8 P1OUT[1:0]

These bits are used to set data to be output from the GPIO port pins.
When a bit = 1, output a high level from the corresponding port pin
When a bit = 0, output a low level from the corresponding port pin

When output is enabled, the port pin outputs the data set here. Although data can be written when output is disabled, it does not affect the pin status. These bits do not affect the outputs when the port is used as a peripheral I/O function.

bits 1:0 P0IN[1:0] (read-only)
The GPIO port pin status can be read out from these bits.
When a bit reads 1, the corresponding port pin = high level
When a bit reads 0, the corresponding port pin = low level

The port pin status can be read out when input is enabled. When input is disabled, these bits are always read as 0. When the port is used for a peripheral I/O function, the input value cannot be read out from these bits.

10.4.10 Port P1 I/O Enable

REG[0212h] PORTP1IOEN Register
Default = 0003h R/W

15	14	13	12	11	10	9	8
n/a						P1IEN[1]	P1IEN[0]
7	6	5	4	3	2	1	0
n/a						P1OEN[1]	P1OEN[0]

bits 9:8 P1IEN[1:0]
These bits enable/disable the GPIO port input.
When a bit = 1, the corresponding GPIO port input is enabled (the port pin status is input.)
When a bit = 0, the corresponding GPIO port input is disabled (input data is fixed at 0.)

When both data output and data input are enabled, the pin output status controlled by this IC can be read. These bits do not affect the input control when the port is used as a peripheral I/O function.

bits 1:0 P1OEN[1:0]
These bits enable/disable the GPIO port output.
When a bit = 1, the corresponding GPIO port output is enabled (data is output from the port pin.)
When a bit = 0, the corresponding GPIO port output is disabled (the port is placed into Hi-Z.)

These bits do not affect the output control when the port is used as a peripheral I/O function.

10.4.11 Port P1 Pull-up/down Control

REG[0214h] PORTP1RCTL Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						P1PDPU[1]	P1PDPU[0]
7	6	5	4	3	2	1	0
n/a						P1REN[1]	P1REN[0]

bits 9:8 P1PDPU[1:0]
These bits select either the pull-up resistor or the pull-down resistor when using a resistor built into the port.
When a bit = 1, the corresponding port pull-up resistor is selected
When a bit = 0, the corresponding port pull-down resistor is selected

The selected pull-up/down resistor is enabled when the PORTP1RCTL.P1REN[x] bit = 1.

bits 1:0 P1REN[1:0]
These bits enable/disable the port pull-up/down control.
When a bit = 1, the corresponding port pull-up/down is enabled (the built-in pull-up/down resistor is used.)
When a bit = 0, the corresponding port pull-up/down is disabled (no pull-up/down control is

performed.)

Enabling this function pulls up or down the port when output is disabled (PORTP1IOEN.P1OEN[x] bit = 0). When output is enabled (PORTP1IOEN.P1OEN[x] bit = 1), the PORTP1RCTL.P1REN[x] bit setting is ineffective regardless of how the PORTP1IOEN.P1IE[x] bit is set and the port is not pulled up/down. These bits do not affect the pull-up/down control when the port is used as a peripheral I/O function.

10.4.12 Port P1 Interrupt Flag

REG[0216h] PORTP1INTF Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
n/a						P1IF[1]	P1IF[0]

bits 1:0 P1IF[1:0]
These bits indicate the port input interrupt cause occurrence status.
When a bit reads 1, an interrupt has occurred
When a bit reads 0, no interrupt has occurred
When a bit is written 1, the interrupt flag is cleared
When a bit is written 0, there is no effect

10.4.13 Port P1 Interrupt Control

REG[0218h] PORTP1INTCTL Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						P1EDGE[1]	P1EDGE[0]
7	6	5	4	3	2	1	0
n/a						P1IE[1]	P1IE[0]

bits 9:8 P1EDGE[1:0]
These bits select the input signal edge to generate a port input interrupt.
When a bit = 1, the corresponding interrupt will occur at a falling edge
When a bit = 0, the corresponding interrupt will occur at a rising edge

bits 1:0 P1IE[1:0]
These bits enable port input interrupts.
When a bit = 1, the corresponding interrupt is enabled
When a bit = 0, the corresponding interrupt is disabled

NOTE: To prevent generating unnecessary interrupts, the corresponding interrupt flag should be cleared before enabling interrupts.

10.4.14 Port P1 Chattering Filter Enable

REG[021Ah] PORTP1CHATEN Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
n/a						P1CHATEN[1]	P1CHATEN[0]

bits 1:0 P1CHATEN[1:0]
These bits enable/disable the chattering filter function.

When a bit = 1, the corresponding chattering filter is enabled
When a bit = 0, the corresponding chattering filter is bypassed (disabled)

10.4.15 Port P1 Mode Select

REG[021Ch] PORTP1MODSEL Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a							

7	6	5	4	3	2	1	0
n/a						P1SEL[1]	P1SEL[0]

bits 1:0 P1SEL[1:0]
These bits select whether each port is used for the GPIO function or a peripheral I/O function.
When a bit = 1, the corresponding port uses peripheral I/O function
When a bit = 0, the corresponding port uses GPIO function

10.4.16 Port P1 Function Select

REG[021Eh] PORTP1FNCSEL Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a							

7	6	5	4	3	2	1	0
n/a				P11MUX[1:0]		P10MUX[1:0]	

bits 3:2 P11MUX[1:0]
bits 1:0 P10MUX[1:0]
These bits select the peripheral I/O function to be assigned to each port pin according to the following table:

Table 10.5 P1 Port Group Function Assignment

Port name	P1SEL[x] = 0	P1SEL[x] = 1							
		P1xMUX = 0x0		P1xMUX = 0x1		P1xMUX = 0x2		P1xMUX = 0x3	
	GPIO	Peripheral	Pin	Peripheral	Pin	Peripheral	Pin	Peripheral	Pin
P10	P10	SND	BZOUT	RTC	RTC1S	-	-	–	–
P11	P11	SND	#BZOUT	CLG	FOUT	-	-	–	–

10.4.17 Ports Clock Control

REG[02E0h] PORTCLK Register							
Default = 0000h							R/WP
n/a							
15	14	13	12	11	10	9	8
CLKDIV bits 3-0				n/a			CLKSRC
7	6	5	4	3	2	1	0

bits 7:4 CLKDIV bits [3:0]

These bits select the division ratio of the GPIO operating clock (chattering filter clock) source.

0x0: 1/1
0x1: 1/2
0x2: 1/4
0x3: 1/8
0x4: 1/16
0x5: 1/32
0x6: 1/64
0x7: 1/128
0x8: 1/256
0x9: 1/512
0xA: 1/1,024
0xB: 1/2,048
0xC: 1/4,096
0xD: 1/8,192
0xE: 1/16,384
0xF: 1/32,768

bit 0 CLKSRC

This bit select the clock source of GPIO (chattering filter).
When this bit = 0, IOSC is the clock source
When this bit = 1, OSC1 is the clock source

10.4.18 Ports Interrupt Flag Group

REG[02E2h] PORTINTFGRP Register							
Default = 0000h							RO
n/a							
15	14	13	12	11	10	9	8
n/a						P1INT	P0INT
7	6	5	4	3	2	1	0

bit 1 P1INT (read only)
bit 0 P0INT (read only)

These bits indicate that Px port group includes a port that has generated an interrupt.
When this bit reads 1, a port has generated an interrupt
When this bit reads 0, no port has generated an interrupt

The PORTINTFGRP.PxINT bit is cleared when the interrupt flag for the port that has generated an interrupt is cleared.

10.5 SPI Registers

10.5.1 T16 CH1 Clock Control

REG[03A0h] T16_1CLK Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
CLKDIV bits 3-0				n/a			CLKSRC
7	6	5	4	3	2	1	0

bits 7:4 CLKDIV bits [3:0]

These bits select the division ratio of the clock source for T16 CH1 (timer for SPI) as follows:

IOSC (CLKSRC = 0)
0x0: 1/1
0x1: 1/2
0x2: 1/4
0x3: 1/8
0x4: 1/16
0x5: 1/32
0x6: 1/64
0x7: 1/128
0x8: 1/256
0x9: 1/512
0xA: 1/1,024
0xB: 1/2,048
0xC: 1/4,096
0xD: 1/8,192
0xE: 1/16,384
0xF: 1/32,768

OSC1 (CLKSRC = 1)
0x0: 1/1
0x1: 1/2
0x2: 1/4
0x3: 1/8
0x4: 1/16
0x5: 1/32
0x6: 1/64
0x7: 1/128
0x8: 1/256
0x9-0xF: 1/1

bit 0 CLKSRC

This bit select the clock source of T16 CH1 (timer for SPI).
When this bit = 0, IOSC is the clock source
When this bit = 1, OSC1 is the clock source

10.5.2 T16 CH1 Mode

REG[03A2h] T16_1MOD Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a							TRMD
7	6	5	4	3	2	1	0

bit 0 TRMD

This bit selects the T16 operation mode.
When this bit = 1, T16 is in one-shot mode
When this bit = 0, T16 is in repeat mode

10.5.3 T16 CH1 Control

REG[03A4h] T16_1CTL Register							
Default = 0000h							R/W
n/a							PRUN
15	14	13	12	11	10	9	8
n/a						PRESET	MODEN
7	6	5	4	3	2	1	0

bit 8 PRUN

This bit starts/stops the timer.
When this bit is written 1, starts the timer
When this bit is written 0, stops the timer
When this bit reads 1, the timer is running
When this bit reads 0, the timer is idle

By writing 1 to this bit, the timer starts count operations. However, the T16_1CTL.MODEN bit must be set to 1 in conjunction with this bit or it must be set in advance. While the timer is running, writing 0 to this bit stops count operations. When the counter stops due to a counter underflow in one-shot mode, this bit is automatically cleared to 0.

bit 1 PRESET

This bit presets the reload data stored in the T16_1TR register to the counter.
When this bit is written 1, preset data is loaded to the timer
When this bit is written 0, there is no effect
When this bit reads 1, loading preset data is in progress
When this bit reads 0, loading of preset data has finished or normal operation

By writing 1 to this bit, the timer presets the T16_1TR register value to the counter. However, the T16_1CTL.MODEN bit must be set to 1 in conjunction with this bit or it must be set in advance. This bit retains 1 during presetting and is automatically cleared to 0 after presetting has finished.

bit 0 MODEN

This bit enables the T16 CH1 operations.
When this bit = 1, T16 CH1 operations are enabled (start supplying operating clock)
When this bit = 0, T16 CH1 operations are disabled (stop supplying operating clock)

10.5.4 T16 CH1 Reload Data

REG[03A6h] T16_1TR Register							
Default = FFFFh							R/W
TR bits 15-8							
15	14	13	12	11	10	9	8
TR bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 TR bits [15:0]

These bits are used to set the initial value to be preset to the counter.
The value set to this register will be preset to the counter when 1 is written to the T16_1CTL.PRESET bit or when the counter underflows.

10.5.5 T16 CH1 Counter Data

REG[03A8h] T16_1TC Register
Default = FFFFh RO

TC bits 15-8							
15	14	13	12	11	10	9	8
TC bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 TC bits [15:0] (read-only)
The current counter value can be read out from these bits.

10.5.6 T16 CH1 Interrupt Flag

REG[03AAh] T16_1INTF Register
Default = 0000h R/W

n/a							
15	14	13	12	11	10	9	8
n/a							UFIF
7	6	5	4	3	2	1	0

bit 0 UFIF
This bit indicates the T16 CH1 underflow interrupt cause occurrence status.
When this bit reads 1, a T16 CH1 underflow interrupt has occurred
When this bit reads 0, no T16 CH1 underflow interrupt has occurred
When this bit is written 1, the T16 CH1 underflow interrupt flag is cleared
When this bit is written 0, there is no effect

10.5.7 T16 CH1 Interrupt Enable

REG[03ACh] T16_1INTE Register
Default = 0000h R/W

n/a							
15	14	13	12	11	10	9	8
n/a							UFIE
7	6	5	4	3	2	1	0

bit 0 UFIE
This bit enables T16 CH1 underflow interrupt.
When this bit = 1, the T16 CH1 underflow interrupt is enabled
When this bit = 0, the T16 CH1 underflow interrupt is disabled

NOTE: To prevent generating unnecessary interrupts, the corresponding interrupt flag should be cleared before enabling interrupts.

10.5.8 SPI Mode

REG[03B0h] SPIMOD Register							
Default = 0700h							R/W
n/a				CHLN bits 3-0			
15	14	13	12	11	10	9	8
n/a		PUEN	NOCLKDIV	LSBFST	CPHA	CPOL	MST
7	6	5	4	3	2	1	0

bits 11:8 CHLN bits [3:0]
These bits set the bit length of transfer data as follows:
0x0: Setting prohibited
0x1-0xF: data bit length = (CHLN[3:0] + 1)

bit 5 PUEN
This bit enables pull-up/down of the input pins.
When this bit = 1, the input pins pull-up/down are enabled
When this bit = 0, the input pins pull-up/down are disabled

bit 4 NOCLKDIV
This bit selects SPICLK in master mode. This setting is ineffective in slave mode.
When this bit = 1, the SPICLK frequency = CLK_SPI frequency (= 16-bit timer operating clock frequency)
When this bit = 0, the SPICLK frequency = 16-bit timer output frequency / 2

bit 3 LSBFST
This bit configures the data format (input/output permutation).
When this bit = 1, the data format is LSB first
When this bit = 0, the data format is MSB first

bit 2 CPHA
bit 1 CPOL
These bits set the SPI clock phase and polarity.

bit 0 MST
This bit sets the SPI operating mode (master mode or slave mode).
When this bit = 1, the SPI operating mode is Master
When this bit = 0, the SPI operating mode is Slave

NOTE: The SPIMOD register settings can be altered only when the SPICTL.MODEN bit = 0.

10.5.9 SPI Control

REG[03B2h] SPICTL Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a						SFTRST	MODEN
7	6	5	4	3	2	1	0

bit 1 SFTRST
This bit issues software reset to SPI.
When this bit is written 1, a software reset is issued
When this bit is written 0, there is no effect
When this bit reads 1, software reset is executing.
When this bit reads 0, software reset has finished. (during normal operation)

Setting this bit resets the SPI shift register and transfer bit counter. This bit is automatically cleared after the reset processing has finished.

bit 0 MODEN
This bit enables the SPI operations.
When this bit = 1, SPI operations are enabled (in master mode, the operating clock is supplied.)
When this bit = 0, SPI operations are disabled (in master mode, the operating clock is stopped.)

NOTE: If the SPICTL.MODEN bit is altered from 1 to 0 while sending/receiving data, the data being sent/received cannot be guaranteed. When setting the SPICTL.MODEN bit to 1 again after that, be sure to write 1 to the SPICTL.SFTRST bit as well.

10.5.10 SPI Transmit Data

REG[03B4h] SPITXD Register
Default = 0000h R/W

TXD bits 15-8							
15	14	13	12	11	10	9	8
TXD bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 TXD bits [15:0]
Data can be written to the transmit data buffer through these bits.

In master mode, writing to the upper byte (TXD[15:8], REG[03B5h]) starts data transfer. For a data bit length of less than 9 bits, the upper byte still needs to be written even though only the lower byte value is transmitted.

Transmit data can be written when the SPIINTF.TBEIF bit = 1 regardless of whether data is being output from the SDO pin or not.

Note that the upper data bits that exceed the data bit length configured by the SPIMOD.CHLN[3:0] bits will not be output from the SDO pin.

NOTE: Be sure to avoid writing to the SPITXD register when the SPIINTF.TBEIF bit = 0. Otherwise, transfer data cannot be guaranteed.

10.5.11 SPI Receive Data

REG[03B6h] SPIRXD Register
Default = 0000h RO

RXD bits 15-8							
15	14	13	12	11	10	9	8
RXD bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 RXD bits [15:0] (read-only)
The receive data buffer can be read through these bits. Received data can be read when the SPIINTF.RBFIF bit = 1 regardless of whether data is being input from the SDI pin or not. Note that the upper bits that exceed the data bit length configured by the SPIMOD.CHLN[3:0] bits become 0.

For data bit length less than 9 bits, the upper byte (RXD[15:8], REG[03B7h]) still needs to be read even though only the lower byte value is used to clear the SPIINTF.RBFIF bit.

10.5.12 SPI Interrupt Flag

REG[03B8h] SPIINTF Register							
Default = 0001h							R/W
n/a							
15	14	13	12	11	10	9	8
BSY	n/a			OEIF	TENDIF	RBFIF	TBEIF
7	6	5	4	3	2	1	0

bit 7 BSY (read-only)
This bit indicates the SPI operating status.
When this bit reads 1, in master mode transmit/receive is busy, in slave mode #SPISS = low level)
When this bit reads 0, the SPI is idle

bit 3 OEIF
bit 2 TENDIF
bit 1 RBFIF
bit 0 TBEIF
These bits indicate the SPI interrupt cause occurrence status.
When a bit reads 1, the corresponding interrupt has occurred
When a bit reads 0, no interrupt has occurred
When a bit is written 1, the corresponding interrupt flag is cleared
When a bit is written 0, there is no effect

The following shows the correspondence between the bit and interrupt:
SPIINTF.OEIF bit: Overrun error interrupt
SPIINTF.TENDIF bit: End-of-transmission interrupt
SPIINTF.RBFIF bit: Receive buffer full interrupt
SPIINTF.TBEIF bit: Transmit buffer empty interrupt

10.5.13 SPI Interrupt Enable

REG[03BAh] SPIINTE Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a				OEIE	TENDIE	RBFIE	TBEIE
7	6	5	4	3	2	1	0

bit 3 OEIE
bit 2 TENDIE
bit 1 RBFIE
bit 0 TBEIE
These bits enable SPI interrupts.
When a bit = 1, the corresponding interrupt is enabled
When a bit = 0, the corresponding interrupt is disabled

The following shows the correspondence between the bit and interrupt:
SPIINTF.OEIE bit: Overrun error interrupt
SPIINTF.TENDIE bit: End-of-transmission interrupt
SPIINTF.RBFIE bit: Receive buffer full interrupt
SPIINTF.TBEIE bit: Transmit buffer empty interrupt

10.5.14 SPI Transmit Buffer Empty DMA Request Enable

REG[03BCh] SPITBEDMAEN Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
n/a				TBEDMAEN bits 3-0			

bits 3:0 TBEDMAEN bits [3:0]
These bits enable the SPI to issue a DMA transfer request to the corresponding DMA channel (Ch.0–Ch.3) when a transmit buffer empty state has occurred.
When a bit = 1, DMA transfer request for the corresponding channel is enabled
When a bit = 0, DMA transfer request for the corresponding channel is disabled

Each bit corresponds to a DMA controller channel.

10.5.15 SPI Receive Buffer Full DMA Request Enable

REG[03BEh] SPIRBFDMAEN Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
n/a				RBFDMAEN bits 3-0			

bits 3:0 RBFDMAEN bits [3:0]
These bits enable the SPI to issue a DMA transfer request to the corresponding DMA channel (Ch.0–Ch.3) when a receive buffer full state has occurred.
When a bit = 1, DMA transfer request for the corresponding channel is enabled
When a bit = 0, DMA transfer request for the corresponding channel is disabled

Each bit corresponds to a DMA controller channel.

10.6 I2C Registers

10.6.1 I2C Clock Control

REG[03C0h] I2CCLK Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
n/a		CLKDIV bits 1-0		n/a			CLKSRC

bits 5:4 CLKDIV bits [1:0]
These bits select the division ratio of the clock source for the I2C as follows:

IOSC (CLKSRC = 0)
0x0: 1/1
0x1: 1/2
0x2: 1/4
0x3: 1/8

OSC1 (CLKSRC = 1)
0x0-0x3: 1/1

bit 0 CLKSRC
This bit select the clock source of the I2C.

When this bit = 0, IOSC is the clock source
When this bit = 1, OSC1 is the clock source

10.6.2 I2C Mode

REG[03C2h] I2CMOD Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
n/a					OADR10	GCEN	n/a

bit 2 OADR10
This bit sets the number of own address bits for slave mode.
When this bit = 1, 10-bit address is set
When this bit = 0, 7-bit address is set

bit 1 GCEN
This bit sets whether to respond to master general calls in slave mode or not.
When this bit = 1, respond to general calls.
When this bit = 0, do not respond to general calls.

NOTE: The I2CMOD register settings can be altered only when the I2CCTL.MODEN bit = 0.

10.6.3 I2C Baud-Rate

REG[03C4h] I2CBR Register
Default = 007Fh R/W

15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
n/a	BRT bits 6-0						

bits 6:0 BRT bits [6:0]
These bits set the I2C transfer rate for master mode.

NOTE: The I2CBR register settings can be altered only when the I2CCTL.MODEN bit = 0.
Be sure to avoid setting the I2CBR register to 0.

10.6.4 I2C Own Address

REG[03C8h] I2COADR Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						OADR bits 9-8	
7	6	5	4	3	2	1	0
OADR bits 7-0							

bits 9:0 OADR bits [9:0]
These bits set the own address for slave mode.
The I2COADR.OADR[9:0] bits are effective in 10-bit address mode (I2CMOD.OADR10 bit = 1), or the I2COADR.OADR[6:0] bits are effective in 7-bit address mode (I2CMOD.OADR10 bit = 0).

NOTE: The I2COADR register settings can be altered only when the I2CCTL.MODEN bit = 0.

10.6.5 I2C Control

REG[03CAh] I2CCTL Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a		MST	TXNACK	TXSTOP	TXSTART	SFTRST	MODEN
7	6	5	4	3	2	1	0

bit 5 MST
This bit selects the I2C operating mode.
When this bit = 1, the I2C mode is master
When this bit = 0, the I2C mode is slave

bit 4 TXNACK
This bit issues a request for sending a NACK at the next responding.
When this bit is written 1, I2C issues a NACK
When this bit is written 0, there is no effect
When this bit reads 1, I2C is on standby or sending a NACK
When this bit reads 0, a NACK has been sent.

This bit is automatically cleared after a NACK has been sent.

bit 3 TXSTOP
This bit issues a STOP condition in master mode. This bit is ineffective in slave mode.
When this bit is written 1, I2C issues a STOP condition.
When this bit is written 0, there is no effect
When this bit reads 1, I2C is on standby or generating a STOP condition
When this bit reads 0, a STOP condition has been generated.

This bit is automatically cleared when the bus free time (tBUF defined in the I2C Specifications) has elapsed after the STOP condition has been generated.

bit 2 TXSTART
This bit issues a START condition in master mode. This bit is ineffective in slave mode.
When this bit is written 1, I2C issues a START condition.
When this bit is written 0, there is no effect
When this bit reads 1, I2C is on standby or generating a START condition
When this bit reads 0, a START condition has been generated.

This bit is automatically cleared when a START condition has been generated.

bit 1 SFTRST
This bit issues software reset to the I2C.
When this bit is written 1, an I2C software reset is issued
When this bit is written 0, there is no effect
When this bit reads 1, an I2C software reset is executing.
When this bit reads 0, I2C software reset has finished (during normal operation)

Setting this bit resets the I2C transmit/receive control circuit and interrupt flags. This bit is automatically cleared after the reset processing has finished.

bit 0 MODEN
This bit enables the I2C operations.
When this bit = 1, I2C operations are enabled (the operating clock is supplied.)
When this bit = 0, I2C operations are disabled (the operating clock is stopped.)

NOTE: If the I2CCTL.MODEN bit is altered from 1 to 0 while sending/receiving data, the data being sent/received cannot be guaranteed. When setting the I2CCTL.MODEN bit to 1 again after that, be sure to write 1 to the I2CCTL.SFTRST bit as well.

10.6.6 I2C Transmit Data

REG[03CCh] I2CTXD Register
Default = 0000h R/W

n/a							
15	14	13	12	11	10	9	8
TXD bits 7-0							
7	6	5	4	3	2	1	0

bits 7:0 TXD bits [7:0]
Data can be written to the transmit data buffer through these bits. Make sure the I2CINTF.TBEIF bit is set to 1 before writing data.

NOTE: Be sure to avoid writing to the I2CTXD register when the I2CINTF.TBEIF bit = 0, otherwise transmit data cannot be guaranteed.

10.6.7 I2C Receive Data

REG[03CEh] I2CRXD Register
Default = 0000h RO

n/a							
15	14	13	12	11	10	9	8
RXD bits 7-0							
7	6	5	4	3	2	1	0

bits 7:0 RXD bits [7:0] (read-only)
The receive data buffer can be read through these bits.

10.6.8 I2C Status and Interrupt Flag

REG[03D0h] I2CINTF Register
Default = 0000h R/W

n/a			SDALOW	SCLLOW	BSY	TR	n/a
15	14	13	12	11	10	9	8
BYTEENDIF	GCIF	NACKIF	STOPIF	STARTIF	ERRIF	RBFIF	TBEIF
7	6	5	4	3	2	1	0

bit 12 SDALOW (read-only)
This bit indicates that SDA is set to low level.
When this bit reads 1, SDA is set to low level
When this bit reads 0, SDA is set to high level

bit 11 SCLLOW (read-only)
This bit indicates that SCL is set to low level.
When this bit reads 1, SCL is set to low level
When this bit reads 0, SCL is set to high level

bit 10 BSY (read-only)
This bit indicates that the I2C bus is placed into busy status.
When this bit reads 1, the I2C bus is busy
When this bit reads 0, the I2C bus is free

bit 9 TR (read-only)
This bit indicates whether the I2C is set in transmission mode or not.
When this bit reads 1, I2C is in transmission mode
When this bit reads 0, I2C is in reception mode

bit 7 BYTEENDIF
bit 6 GCIF
bit 5 NACKIF
bit 4 STOPIF
bit 3 STARTIF
bit 2 ERRIF
bit 1 RBFIF
bit 0 TBEIF
These bits indicate the I2C interrupt cause occurrence status.
When a bit reads 1, the corresponding interrupt has occurred
When a bit reads 0, no interrupt has occurred
When a bit is written 1, the corresponding interrupt flag is cleared
When a bit is written 0 , there is no effect

The following shows the correspondence between the bit and interrupt:
I2CINTF.BYTEENDIF bit: End of transfer interrupt
I2CINTF.GCIF bit: General call address reception interrupt
I2CINTF.NACKIF bit: NACK reception interrupt
I2CINTF.STOPIF bit: STOP condition interrupt
I2CINTF.STARTIF bit: START condition interrupt
I2CINTF.ERRIF bit: Error detection interrupt
I2CINTF.RBFIF bit: Receive buffer full interrupt
I2CINTF.TBEIF bit: Transmit buffer empty interrupt

10.6.9 I2C Interrupt Enable

REG[03D2h] I2CINTE Register
Default = 0000h R/W

n/a							
15	14	13	12	11	10	9	8
BYTEENDIE 7	GCIE 6	NACKIE 5	STOPIE 4	STARTIE 3	ERRIE 2	RBFIE 1	TBEIE 0

bit 7 BYTEENDIE
bit 6 GCIE
bit 5 NACKIE
bit 4 STOPIE
bit 3 STARTIE
bit 2 ERRIE
bit 1 RBFIE
bit 0 TBEIE
These bits enable I2C interrupts.
When a bit = 1, the corresponding interrupt is enabled
When a bit = 0, the corresponding interrupt is disabled

The following shows the correspondence between the bit and interrupt:
I2CINTF.BYTEENDIE bit: End of transfer interrupt
I2CINTF.GCIEF bit: General call address reception interrupt
I2CINTF.NACKIE bit: NACK reception interrupt
I2CINTF.STOPIE bit: STOP condition interrupt
I2CINTF.STARTIE bit: START condition interrupt

I2CINTF.ERRIE bit: Error detection interrupt
I2CINTF.RBFIE bit: Receive buffer full interrupt
I2CINTF.TBEIE bit: Transmit buffer empty interrupt

10.6.10 I2C Transmit Buffer Empty DMA Request Enable

REG[03D4h] I2CTBEDMAEN Register
Default = 0000h R/W

n/a							
15	14	13	12	11	10	9	8
n/a				TBEDMAEN bits 3-0			
7	6	5	4	3	2	1	0

bits 3:0 TBEDMAEN bits [3:0]
These bits enable the I2C to issue a DMA transfer request to the corresponding DMA channel (Ch.0–Ch.3) when a transmit buffer empty state has occurred.
When this bit = 1, DMA transfer request is enabled
When this bit = 0, DMA transfer request is disabled

Each bit corresponds to a DMA controller channel.

10.6.11 I2C Receive Buffer Full DMA Request Enable

REG[03D6h] I2CRBFDMAEN Register
Default = 0000h R/W

n/a							
15	14	13	12	11	10	9	8
n/a				RBFDMAEN bits 3-0			
7	6	5	4	3	2	1	0

bits 3:0 RBFDMAEN bits [3:0]
These bits enable the I2C to issue a DMA transfer request to the corresponding DMA channel (Ch.0–Ch.3) when a receive buffer full state has occurred.
When this bit = 1, DMA transfer request is enabled
When this bit = 0, DMA transfer request is disabled

Each bit corresponds to a DMA controller channel.

10.7 QSPI Registers

10.7.1 T16 CH2 Clock Control

REG[0680h] T16_2CLK Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
CLKDIV bits 3-0				n/a			CLKSRC
7	6	5	4	3	2	1	0

bits 7:4 CLKDIV bits [3:0]

These bits select the division ratio of the clock source for T16 CH2 (timer for QSPI) as follows:

IOSC (CLKSRC = 0)
0x0: 1/1
0x1: 1/2
0x2: 1/4
0x3: 1/8
0x4: 1/16
0x5: 1/32
0x6: 1/64
0x7: 1/128
0x8: 1/256
0x9: 1/512
0xA: 1/1,024
0xB: 1/2,048
0xC: 1/4,096
0xD: 1/8,192
0xE: 1/16,384
0xF: 1/32,768

OSC1 (CLKSRC = 1)
0x0: 1/1
0x1: 1/2
0x2: 1/4
0x3: 1/8
0x4: 1/16
0x5: 1/32
0x6: 1/64
0x7: 1/128
0x8: 1/256
0x9-0xF: 1/1

bit 0 CLKSRC

This bit select the clock source of T16 CH2 (timer for QSPI).
When this bit = 1, OSC1 is the clock source
When this bit = 0, IOSC is the clock source

10.7.2 T16 CH2 Mode

REG[0682h] T16_2MOD Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a							TRMD
7	6	5	4	3	2	1	0

bit 0 TRMD

This bit selects the T16 operation mode.
When this bit = 1, one-shot mode is selected
When this bit = 0, repeat mode is selected

10.7.3 T16 CH2 Control

REG[0684h] T16_2CTL Register							
Default = 0000h							R/W
n/a							PRUN
15	14	13	12	11	10	9	8
n/a						PRESET	MODEN
7	6	5	4	3	2	1	0

bit 8 PRUN

This bit starts/stops the timer.
When this bit is written 1, the timer is started
When this bit is written 0, the timer is stopped
When this bit reads 1, the timer is running
When this bit reads 0, the timer is idle

By writing 1 to this bit, the timer starts count operations. However, the T16_2CTL.MODEN bit must be set to 1 in conjunction with this bit or it must be set in advance. While the timer is running, writing 0 to this bit stops count operations. When the counter stops due to a counter underflow in one-shot mode, this bit is automatically cleared to 0.

bit 1 PRESET

This bit presets the reload data stored in the T16_2TR register to the counter.
When this bit is written 1, preset data is loaded to the counter
When this bit is written 0, there is no effect
When this bit reads 1, loading preset data is in progress
When this bit reads 0, loading of preset data has finished or normal operation

By writing 1 to this bit, the timer presets the T16_2TR register value to the counter. However, the T16_2CTL.MODEN bit must be set to 1 in conjunction with this bit or it must be set in advance. This bit retains 1 during presetting and is automatically cleared to 0 after presetting has finished.

bit 0 MODEN

This bit enables the T16 CH2 operations.
When this bit = 1, T16 CH2 operations are enabled (start supplying operating clock)
When this bit = 0, T16 CH2 operations are disabled (stop supplying operating clock)

10.7.4 T16 CH2 Reload Data

REG[0686h] T16_2TR Register							
Default = FFFFh							R/W
TR bits 15-8							
15	14	13	12	11	10	9	8
TR bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 TR bits [15:0]

These bits are used to set the initial value to be preset to the counter.
The value set to this register will be preset to the counter when 1 is written to the T16_2CTL.PRESET bit or when the counter underflows.

10.7.5 T16 CH2 Counter Data

REG[0688h] T16_2TC Register							
Default = FFFFh							RO
TC bits 15-8							
15	14	13	12	11	10	9	8
TC bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 TC bits [15:0] (read-only)
The current counter value can be read out from these bits.

10.7.6 T16 CH2 Interrupt Flag

REG[068Ah] T16_2INTF Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a							UFIF
7	6	5	4	3	2	1	0

bit 0 UFIF
This bit indicates the T16 CH2 underflow interrupt cause occurrence status.
When this bit reads 1, an underflow interrupt has occurred
When this bit reads 0, no interrupt has occurred
When this bit is written 1, the underflow interrupt flag is cleared
When this bit is written 0, there is no effect

10.7.7 T16 CH2 Interrupt Enable

REG[068Ch] T16_2INTE Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a							UFIE
7	6	5	4	3	2	1	0

bit 0 UFIE
This bit enables T16 CH2 underflow interrupt.
When this bit =1, the underflow interrupt is enabled
When this bit =0, the underflow interrupt is disabled

NOTE: To prevent generating unnecessary interrupts, the corresponding interrupt flag should be cleared before enabling interrupts.

10.7.8 QSPI Mode

REG[0690h] QSPIMOD Register							
Default = 7700h							R/W
CHDL bits 3-0				CHLN bits 3-0			
15	14	13	12	11	10	9	8
TMOD bits 1-0		PUEN	NOCLKDIV	LSBFST	CPHA	CPOL	MST
7	6	5	4	3	2	1	0

bits 15:12 CHDL bits [3:0]
These bits set the number of clocks to drive the serial output data lines. This setting is required to output the XIP confirmation bit to Micron Flash memories or to output the mode byte to

Spansion Flash memories. The data line drive length settings are as follows:
Data line drive bit length = (CHDL[3:0] + 1)

These bits must be set to a value smaller than or equal to the QSPIMOD.CHLN[3:0] bit setting.

NOTE: When using the QSPI in slave mode, the QSPIMOD.CHDL[3:0] bits should be set to the same value as the QSPIMOD.CHLN[3:0] bits.

bits 11:8 CHLN bits [3:0]
These bits set the number of clocks for data transfer as follows:
0x0: Setting prohibited
0x1-0xF: number data transfer clocks = (CHLN[3:0] + 1)

bits 7:6 TMOD bits [1:0]
These bits select a transfer mode as follows:
0x0: Single transfer mode
The QSPID[1:0] pins are configured as an output pin and input pin, respectively. The QSPID[3:2] pins are not used.
0x1: Dual transfer mode
The QSPID[1:0] pins are configured as input or output pins according to the QSPIMOD.DIR bit setting. The QSPID[3:2] pins are not used.
0x2: Quad transfer mode
The QSDIOn[3:0] pins are configured as input or output pins according to the QSPI_OD.DIR bit setting.
0x3: Reserved

bit 5 PUEN
This bit enables pull-up/down of the QSPI pins.
When this bit = 1, pull-up/down is enabled
When this bit = 0, pull-up/down is disabled

For the QSPID[3:0] pins, weak pull-ups are enabled by this bit.
For the QSPICLK pin, the pull-up or pull-down is enabled by this bit only if QSPI is in slave mode (MST=0); otherwise pull-up/pull-down is disabled in master mode (MST=1).
CPOL=1 selects pull-up and CPOL=0 selects pull-down.
For the #QSPISS pin, pull-up is enabled by this bit only if QSPI in in slave mode (MST=0); otherwise pull-up is disabled in master mode (MST=1).

bit 4 NOCLKDIV
This bit selects QSPICLK in master mode. This setting is ineffective in slave mode.
When this bit = 1, QSPICLK frequency = CLK_QSPI frequency (= 16-bit timer operating clock frequency)
When this bit = 0, QSPICLK frequency = 16-bit timer output frequency / 2

bit 3 LSBFST
This bit configures the data format (input/output permutation).
When this bit = 1, the data format is LSB first
When this bit = 0, the data format is MSB first

bit 2 CPHA
bit 1 CPOL
These bits set the QSPI clock phase and polarity.

bit 0 MST
This bit sets the QSPI operating mode (master mode or slave mode).

When this bit = 1, the QSPI operating mode is master
When this bit = 0, the QSPI operating mode is slave

NOTE: The QSPIMOD register settings can be altered only when the QSPICTL.MODEN bit = 0.

10.7.9 QSPI Control

REG[0692h] QSPICTL Register
Default = 0004h R/W

n/a							
15	14	13	12	11	10	9	8
n/a				DIR	MSTSSO	SFTRST	MODEN
7	6	5	4	3	2	1	0

bit 3 DIR
This bit sets the data transfer direction on the QSPID[3:0] lines when the QSPIMOD.TMOD[1:0] bits are set to 1 or 2.
When this bit = 1, the data transfer direction is input
When this bit = 0, the data transfer direction is output

bit 2 MSTSSO
This bit controls and indicates the #QSPISS pin status.
When this bit = 1, #QSPISS = high (the device is deselected)
When this bit = 0, #QSPISS = low (the device is selected)

In memory mapped access mode, the #QSPISS pin is automatically controlled by the internal state machine. Reading this bit allows monitoring of the current #QSPISS pin status at any time.

bit 1 SFTRST
This bit issues software reset to QSPI.
When this bit is written 1, a software reset is issued
When this bit is written 0, there is no effect
When this bit reads 1, the software reset is executing.
When this bit reads 0, the software reset has finished (during normal operation)

Setting this bit resets the QSPI shift register and transfer bit counter. This bit is automatically cleared after the reset processing has finished.

bit 0 MODEN
This bit enables the QSPI operations.
When this bit = 1, QSPI operations are enabled (the operating clock is supplied)
When this bit = 0, QSPI operations are disabled (the operating clock is stopped)

NOTE: If the QSPICTL.MODEN bit is altered from 1 to 0 while sending/receiving data, the data being sent/received cannot be guaranteed. When setting the QSPICTL.MODEN bit to 1 again after that, be sure to write 1 to the QSPICTL.SFTRST bit as well.

10.7.10 QSPI Transmit Data

REG[0694h] QSPITXD Register							
Default = 0000h							R/W
TXD bits 15-8							
15	14	13	12	11	10	9	8
TXD bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 TXD bits [15:0]

Data can be written to the transmit data buffer through these bits.

In master mode, writing to the upper byte (TXD[15:8], REG[0695h]) starts data transfer. For data bit length of less than 9 bits, the upper byte still needs to be written even though only the lower byte value is transmitted.

Transmit data can be written when the QSPIINTF.TBEIF bit = 1 regardless of whether data is being output from the QSPID[3:0] pins or not.

Note that the upper data bits that exceed the data bit length configured by the QSPIMOD.CHLN[3:0] bits will not be output from the QSPID[3:0] pins.

NOTE: Be sure to avoid writing to the QSPITXD register when the QSPIINTF.TBEIF bit = 0. Otherwise, transfer data cannot be guaranteed.

10.7.11 QSPI Receive Data

REG[0696h] QSPIRXD Register							
Default = 0000h							RO
RXD bits 15-8							
15	14	13	12	11	10	9	8
RXD bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 RXD bits [15:0] (read-only)

The receive data buffer can be read through these bits. Received data can be read when the QSPIINTF.RBFIF bit = 1 regardless of whether data is being input from the QSPID[3:0] pins or not. Note that the upper bits that exceed the data bit length configured by the QSPIMOD.CHLN[3:0] bits become 0.

For data bit length less than 9 bits, the upper byte (RXD[15:8], REG[0697h]) still needs to be read even though only the lower byte value is used to clear the QSPIINTF.RBFIF bit.

10.7.12 QSPI Interrupt Flag

REG[0698h] QSPIINTF Register							
Default = 0001h							R/W
n/a							
15	14	13	12	11	10	9	8
BSY	MMABSY	n/a		OEIF	TENDIF	RBFIF	TBEIF
7	6	5	4	3	2	1	0

bit 7 BSY (read-only)

This bit indicates the QSPI operating status.
When this bit reads 1, QSPI transmit/receive is busy
When this bit reads 0, QSPI is idle

bit 6 MMABSY (read-only)
This bit indicates the QSPI memory mapped access operating status.
When this bit reads 1, the memory mapped access state machine is busy
When this bit reads 0, the memory mapped access state machine is idle

bit 3 OEIF
bit 2 TENDIF
bit 1 RBFIF
bit 0 TBEIF
These bits indicate the QSPI interrupt cause occurrence status.
When a bit reads 1, the corresponding interrupt has occurred
When a bit reads 0, no interrupt has occurred
When a bit is written 1, the corresponding interrupt flag is cleared
When a bit is written 0, there is no effect

The following shows the correspondence between the bit and interrupt:
QSPIINTF.OEIF bit: Overrun error interrupt
QSPIINTF.TENDIF bit: End-of-transmission interrupt
QSPIINTF.RBFIF bit: Receive buffer full interrupt
QSPIINTF.TBEIF bit: Transmit buffer empty interrupt

10.7.13 QSPI Interrupt Enable

REG[069Ah] QSPIINTE Register
Default = 0000h R/W

n/a							
15	14	13	12	11	10	9	8
n/a				OEIE	TENDIE	RBFIE	TBEIE
7	6	5	4	3	2	1	0

bit 3 OEIE
bit 2 TENDIE
bit 1 RBFIE
bit 0 TBEIE
These bits enable QSPI interrupts.
When a bit = 1, the corresponding interrupt is enabled
When a bit = 0, the corresponding interrupt is disabled

The following shows the correspondence between the bit and interrupt:
QSPIINTF.OEIE bit: Overrun error interrupt
QSPIINTF.TENDIE bit: End-of-transmission interrupt
QSPIINTF.RBFIE bit: Receive buffer full interrupt
QSPIINTF.TBEIE bit: Transmit buffer empty interrupt

10.7.14 QSPI Transmit Buffer Empty DMA Request Enable

REG[069Ch] QSPITBEDMAEN Register
Default = 0000h R/W

n/a							
15	14	13	12	11	10	9	8
n/a				TBEDMAEN bits 3-0			
7	6	5	4	3	2	1	0

bits 3:0 TBEDMAEN bits [3:0]
These bits enable the QSPI to issue a DMA transfer request to the corresponding DMA channel (Ch.0–Ch.3) when a transmit buffer empty state has occurred.
When a bit is written 1, DMA transfer request is enabled for the corresponding channel

When a bit is written 0, DMA transfer request is disabled for the corresponding channel

Each bit corresponds to a DMA controller channel.

10.7.15 QSPI Receive Buffer Full DMA Request Enable

REG[069Eh] QSPIRBFDMAEN Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a				RBFDMAEN bits 3-0			
7	6	5	4	3	2	1	0

bits 3:0 RBFDMAEN bits [3:0]
These bits enable the QSPI to issue a DMA transfer request to the corresponding DMA channel (Ch.0–Ch.3) when a receive buffer full state has occurred.
When a bit is written 1, DMA transfer request is enabled for the corresponding channel
When a bit is written 0, DMA transfer request is disabled for the corresponding channel

Each bit corresponds to a DMA controller channel.

10.7.16 QSPI FIFO Data Ready DMA Request Enable

REG[06A0h] QSPIFRLDMAEN Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a				FRLDMAEN bits 3-0			
7	6	5	4	3	2	1	0

bits 3:0 FRLDMAEN bits [3:0]
These bits enable the QSPI to issue a DMA transfer request to the corresponding DMA channel (Ch.0–Ch.3) when data is prefetched into the FIFO (FIFO data ready).
When a bit is written 1, DMA transfer request is enabled for the corresponding channel
When a bit is written 0, DMA transfer request is disabled for the corresponding channel

Each bit corresponds to a DMA controller channel.

10.7.17 QSPI Memory Mapped Access Configuration 1

REG[06A2h] QSPIMMACFG1 Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a				TCSH bits 3-0			
7	6	5	4	3	2	1	0

bits 3:0 TCSH bits [3:0]
When non-sequential reading from a serial Flash memory address, which is not continuous to the previous read address occurs in memory mapped access mode, the #QSPISS signal is negated and reasserted. The new address is sent to the serial Flash memory before reading data. The QSPIMMACFG1.TCSH[3:0] bits specify the period to negate the #QSPISS signal in number of clocks as follows:

#QSPISS inactive period between non-sequential readings = (TCHS[3:0] + 1)

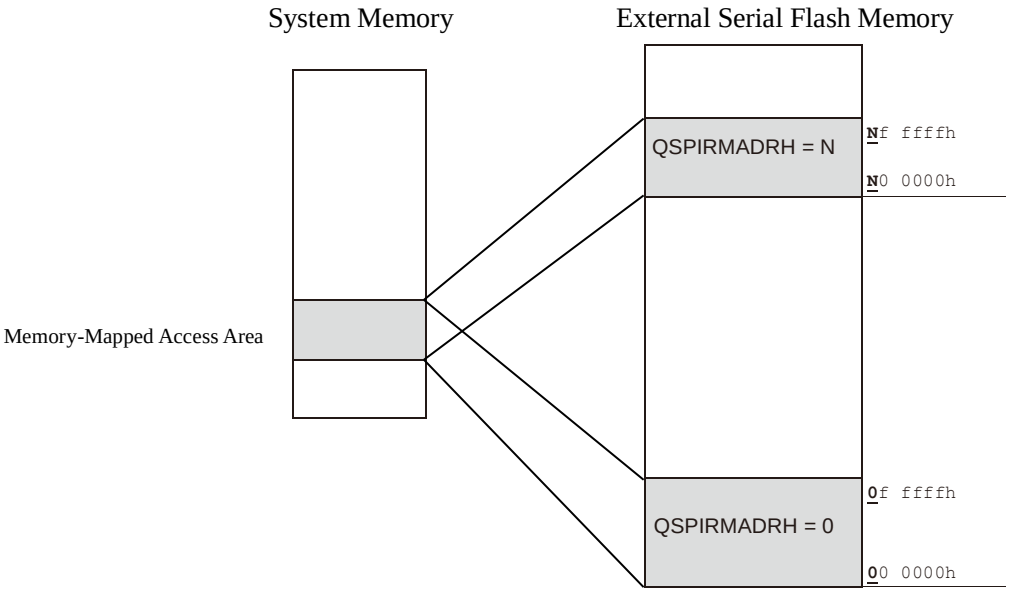
10.7.18 QSPI Remapping Address High Bits

REG[06A4h] QSPIRMADRH Register							
Default = 0000h							R/W
RMADR bits 31-24							
15	14	13	12	11	10	9	8
RMADR bits 23-20				n/a			
7	6	5	4	3	2	1	0

bits 15:4 RMADR bits [31:20]

These bits specify the high-order 12 bits of the external serial Flash memory area start address (assumed as 32 bits) to be remapped to the system memory area allocated for memory mapped access mode. When the external serial Flash memory is read using the memory mapped access function, the value specified here is added, as an offset, to the relative address in the memory mapped access area to generate the external serial Flash memory address to actually be accessed.

NOTE: Make sure the QSPIMMACFG2.MMAEN = 0 when altering the QSPI_MADRH.RMADR[31:20] bits.



10.7.19 QSPI Memory Mapped Access Configuration 2

REG[06A6h] QSPIMMACFG2 Register							
Default = 0000h							R/W
DUMDL bits 3-0				DUMLN bits 3-0			
15	14	13	12	11	10	9	8
DATTMOD bits 1-0		DUMTMOD bits 1-0		ADRTMOD bits 1-0		ADRCYC	MMAEN
7	6	5	4	3	2	1	0

bits 15:12 DUMDL bits [3:0]

These bits set the number of clocks for driving the serial data lines during the dummy cycle output when accessing the external serial Flash memory in the memory mapped access mode. This setting is required to output the XIP confirmation bit to Micron Flash memories or to output the mode byte to Spansion Flash memories. The data line drive length settings are as follows:

Data line drive bit length clock cycles = (DUMDL[3:0] + 1)

These bits must be set to a value smaller than or equal to the QSPIMMACFG2.DUMLN[3:0] bit setting.

bits 11:8 DUMLN bits [3:0]
These bits set the dummy cycle length in a number of clocks when accessing the external serial Flash memory in the memory mapped access mode as follows:
0x0: Setting prohibited
0x1-0xF: number dummy clock cycles = (DUMLN[3:0] + 1)

bits 7:6 DATTMOD bits [1:0]
These bits select the transfer mode for the data cycle when accessing the external serial Flash memory in the memory mapped access mode as follows:
0x0: Single transfer mode. QSPID[1:0] pins are used, QSPID[3:2] pins are unused.
0x1: Dual transfer mode. QSPID[1:0] pins are used, QSPID[3:2] pins are unused.
0x2: Quad transfer mode. QSPID[3:0] pins are used.
0x3: Reserved

bits 5:4 DUMTMOD bits [1:0]
These bits select the transfer mode for the dummy cycle when accessing the external serial Flash memory in the memory mapped access mode as follows:
0x0: Single transfer mode. QSPID[1:0] pins are used, QSPID[3:2] pins are unused.
0x1: Dual transfer mode. QSPID[1:0] pins are used, QSPID[3:2] pins are unused.
0x2: Quad transfer mode. QSPID[3:0] pins are used.
0x3: Reserved

bits 3:2 ADRTMOD bits [1:0]
These bits select the transfer mode for the address cycle when accessing the external serial Flash memory in the memory mapped access mode as follows:
0x0: Single transfer mode. QSPID[1:0] pins are used, QSPID[3:2] pins are unused.
0x1: Dual transfer mode. QSPID[1:0] pins are used, QSPID[3:2] pins are unused.
0x2: Quad transfer mode. QSPID[3:0] pins are used.
0x3: Reserved

bit 1 ADRCYC
This bit selects the address mode from 24 and 32 bits when accessing the external serial Flash memory in the memory mapped access mode.
When this bit = 1, 32-bit address mode (4-byte address cycle) is selected
When this bit = 0, 24-bit address mode (3-byte address cycle) is selected

bit 0 MMAEN
This bit enables memory mapped access mode for accessing the external serial Flash memory.
When this bit = 1, memory mapped access mode is enabled
When this bit = 0, memory mapped access mode is disabled (register access mode)

When this bit is altered from 1 to 0, the QSPI sends extra address and dummy cycles to the external serial Flash memory. The address cycle outputs either a three or four-byte address according to the QSPIMMACFG2.ADRCYC bit setting, with all address bits set to 1. The dummy cycle is output according to the QSPIMMACFG2.DUMLN[3:0] and QSPIMMACFG2.DUMDL[3:0] bit settings, with a mode byte for terminating the XIP session of the external serial Flash memory that has been configured using the QSPIMB.XIPEXT[7:0] bits.

10.7.20 QSPI Mode Byte

REG[06A8h] QSPIMB Register							
Default = 0000h							R/W
XIPACT bits 7-0							
15	14	13	12	11	10	9	8
XIPEXT bits 7-0							
7	6	5	4	3	2	1	0

bits 15:8 XIPACT bits [7:0]
These bits configure the mode byte for activating an XIP session of the external serial Flash memory to be accessed in memory mapped access mode.

bits 7:0 XIPEXT bits [7:0]
These bits configure the mode byte for terminating the XIP session of the external serial Flash memory being accessed in memory mapped access mode.

NOTE: In memory mapped access mode, the mode byte is always output from the LSB first. When using a Flash memory that expects the mode byte to be output from the MSB first, write the mode byte to this register in reverse bit order.

10.8 SND Registers

10.8.1 SND Clock Control

REG[0700h] SNDCLK Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a	CLKDIV bits 2-0			n/a			CLKSRC
7	6	5	4	3	2	1	0

bits 6:4 CLKDIV bits [2:0]
These bits select the division ratio of the clock source for SND (Sound Generator) as follows:

IOSC (CLKSRC = 0)
0x0: 1/16
0x1: 1/32
0x2: 1/64
0x3: 1/128
0x4: 1/256
0x5: 1/512
0x6: Reserved
0x7: Reserved

OSC1 (CLKSRC = 1)
0x0 – 0x7: 1/1

NOTE: The SNDCLK register settings can be altered only when the SNDCTL.MODEN bit = 0.

10.8.2 SND Select

REG[0702h] SNDSEL Register							
Default = 0000h							R/W
n/a				STIM bits 3-0			
15	14	13	12	11	10	9	8
n/a				SINV	MODSEL bits 1-0		
7	6	5	4	3	2	1	0

bits 11:8 STIM bits [3:0]
These bits select a tempo (when melody mode is selected) or a one-shot buzzer output duration (when one-shot buzzer mode is selected).

Table 10.6 Tempo/One-Shot Buzzer Output Duration Selections (CLK_SND = 32,768Hz)

SNDSEL.STIM[3:0]	Tempo (= Quarter note/minute)	One-shot buzzer output duration (ms)
0xF	30	250.0
0xE	32	234.4
0xD	34.3	218.8
0xC	36.9	203.1
0xB	40	187.5
0xA	43.6	171.9
0x9	48	156.3
0x8	53.3	140.6
0x7	60	125.0
0x6	68.6	109.4
0x5	80	93.8
0x4	96	78.1
0x3	120	62.5
0x2	160	46.9
0x1	240	31.3
0x0	480	15.6

NOTE: Be sure to avoid altering these bits when SNDINTF.SBSY bit = 1.

bit 2 SINV
This bit selects an output pin drive mode.
When this bit = 1, normal drive mode is selected
When this bit = 0, direct drive mode is selected

bits 1:0 MODSEL bits [1:0]
These bits select a sound output mode.
0x0: Normal buzzer mode
0x1: One-shot buzzer mode
0x2: Melody mode
0x3: Reserved

10.8.3 SND Control

REG[0704h] SNDCTL Register							
Default = 0000h							R/W
n/a							SSTP
15	14	13	12	11	10	9	8
n/a							MODEN
7	6	5	4	3	2	1	0

bit 8 SSTP

This bit stops sound output.
When this bit is written 1, sound output is stopped
When this bit is written 0, there is no effect
When this bit reads 1, a stop is in process
When this bit reads 0, stop process has completed/idle

The SNDCTL.SSTP bit is used to stop buzzer output in normal buzzer mode. After 1 is written, this bit is cleared to 0 when the sound output has completed. Also in one-shot buzzer mode/melody mode, writing 1 to this bit can forcibly terminate the sound output.

bit 0 MODEN

This bit enables the SND operations.
When this bit = 1, SND operations are enabled (the operating clock is supplied)
When this bit = 0, SND operations are disabled (the operating clock is stopped)

10.8.4 SND Data

REG[0706h] SNDDAT Register							
Default = 00FFh							R/W
MDTI	MDRS	SLEN bits 5-0					
15	14	13	12	11	10	9	8
SFRQ bits 7-0							
7	6	5	4	3	2	1	0

This register functions as a sound buffer. Writing data to the upper byte ([15:8], REG[0707h]) starts sound output. Therefore, the lower byte should be written before the upper byte.

bit 15 MDTI

This bit specifies a tie or slur (continuous play with the previous note) in melody mode.
When this bit = 1, tie/slur is enabled
When this bit = 0, tie/slur is disabled

This bit is ignored in normal buzzer mode/one-shot buzzer mode.

bit 14 MDRS

This bit selects the output type in melody mode from a note or a rest .
When this bit = 1, output type is rest
When this bit = 0, output type is note

When a rest is selected, the BZOUT pin goes low and the #BZOUT pin goes high during the output duration. This bit is ignored in normal buzzer mode/one-shot buzzer mode.

bits 13:8 SLEN bits [5:0]

These bits select a duration (when melody mode is selected) or a buzzer signal duty ratio (when normal buzzer mode/one-shot buzzer mode is selected).

bits 7:0 SFRQ bits [7:0]
These bits select a scale (when melody mode is selected) or a buzzer signal frequency (when normal buzzer mode/one-shot buzzer mode is selected).

NOTES:
In normal buzzer mode/one-shot buzzer mode, only the low-order 6 bits (SNDDAT.SFRQ[5:0] bits) are effective within the SNDDAT.SFRQ[7:0] bits. Always set the SNDDAT.SFRQ[7:6] bits to 0x0.
The SNDDAT register allows 16-bit data writing only. Data writings in 8-bit size will be ignored.

10.8.5 SND Interrupt Flag

REG[0708h] SNDINTF Register
Default = 0002h R/W

n/a							SBSY
15	14	13	12	11	10	9	8
n/a						EMIF	EDIF
7	6	5	4	3	2	1	0

bit 8 SBSY (read-only)
This bit indicates the sound output status.
When this bit reads 1, sound is being output
When this bit reads 0, sound is idle

bit 1 EMIF (read-only)
Sound buffer empty interrupt status.
When this bit reads 1, the interrupt has occurred
When this bit reads 0, no interrupt has occurred

This bit is cleared by writing the SNDDAT register.

bit 0 EDIF
Sound output completion interrupt status.
When this bit reads 1, the interrupt has occurred
When this bit reads 0, no interrupt has occurred
When this bit is written 1, the interrupt flag is cleared
When this bit is written 0, there is no effect

10.8.6 SND Interrupt Enable

REG[070Ah] SNDINTE Register
Default = 0002h R/W

n/a							
15	14	13	12	11	10	9	8
n/a						EMIE	EDIE
7	6	5	4	3	2	1	0

bit 1 EMIE
bit 0 EDIE
These bits enable SND interrupts.
When a bit = 1, he corresponding interrupt is enabled
When a bit = 0, the corresponding interrupt is disabled

The following shows the correspondence between the bit and interrupt:
SNDINTE.EMIE bit: Sound buffer empty interrupt
SNDINTE.EDIE bit: Sound output completion interrupt

10.8.7 SND Buffer Empty DMA Request Enable

REG[070Ch] SNDEMDMAEN Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a				EMDMAEN bits 3-0			
7	6	5	4	3	2	1	0

bits 3:0 EMDMAEN bits [3:0]

These bits enable the SND to issue a DMA transfer request to the corresponding DMA channel (Ch.0–Ch.3) when the sound buffer empty state has occurred.
When a bit is written 1: DMA transfer request is enabled for the corresponding channel
When a bit is written 0: DMA transfer request is disabled for the corresponding channel

Each bit corresponds to a DMA controller channel.

10.9 REMC Registers

10.9.1 REMC Clock Control

REG[0720h] REMCCLK Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
CLKDIV bits 3-0				n/a			CLKSRC
7	6	5	4	3	2	1	0

bits 7:4 CLKDIV bits [3:0]

These bits select the division ratio of the clock source for REMC (IR remote control transmitter) as follows:

IOSC (CLKSRC = 0)
0x0: 1/1
0x1: 1/2
0x2: 1/4
0x3: 1/8
0x4: 1/16
0x5: 1/32
0x6: 1/64
0x7: 1/128
0x8: 1/256
0x9: 1/512
0xA: 1/1,024
0xB: 1/2,048
0xC: 1/4,096
0xD: 1/8,192
0xE: 1/16,384
0xF: 1/32,768

OSC1 (CLKSRC = 1)
0x0: 1/1
0x1: 1/2
0x2: 1/4
0x3: 1/8
0x4: 1/16
0x5: 1/32
0x6: 1/64
0x7: 1/128
0x8: 1/256
0x9-0xF: 1/1

bit 0 CLKSRC
This bit select the clock source of REMC (IR remote control transmitter).
When this bit = 0, IOSC is the clock source
When this bit = 1, OSC1 is the clock source

NOTE: The REMCCLK register settings can be altered only when the REMCDBCTL.MODEN bit = 0.

10.9.2 REMC Data Bit Counter Control

REG[0722h] REMCDBCTL Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						PRESET	PRUN
7	6	5	4	3	2	1	0
n/a			REMOINV	BUFEN	TRMD	REMCRST	MODEN

bit 9 PRESET
This bit resets the internal counters (16-bit counter for data signal generation and 8-bit counter for carrier generation).
When this bit is written 1, internal counters are reset
When this bit is written 0, there is no effect
When this bit reads 1, resetting is in progress
When this bit reads 0, resetting has finished or normal operation

Before the counter can be reset using this bit, the REMCDBCTL.MODEN bit must be set to 1. This bit is cleared to 0 after the counter reset operation has finished or when 1 is written to the REMCDBCTL.REMCRST bit.

bit 8 PRUN
This bit starts/stops counting by the internal counters (16-bit counter for data signal generation and 8-bit counter for carrier generation).
When this bit is written 1, counting is started
When this bit is written 0, counting is stopped
When this bit reads 1, counting is in progress
When this bit reads 0, counters are idle

Before the counter can start counting by this bit, the REMCDBCTL.MODEN bit must be set to 1. While the counter is running, writing 0 to the REMC3BCTL.PRUN bit stops count operations. When the counter stops by occurrence of a compare DB in one-shot mode, this bit is automatically cleared to 0.

bit 4 REMOINV
This bit inverts the REMO output signal.
When this bit = 1, the REMO output signal is inverted
When this bit = 0, the REMO output signal is non-inverted

bit 3 BUFEN
This bit enables or disables the compare buffers.
When this bit = 1, the compare buffers are enabled
When this bit = 0, the compare buffers are disabled

NOTE: The REMCDBCTL.BUFEN bit must be set to 0 when setting the data signal duty and cycle for the first time.

bit 2 TRMD
This bit selects the operation mode of the 16-bit counter for data signal generation.
When this bit = 1, the operation mode is one-shot mode
When this bit = 0, the operation mode is repeat mode

bit 1 REMCRST
This bit issues software reset to the REMC.
When this bit is written 1, a software reset is issued
When this bit is written 0, there is no effect
When this bit reads 1, the software reset is executing.
When this bit reads 0, the software reset has finished (during normal operation)

Setting this bit resets the REMC internal counters and interrupt flags. This bit is automatically cleared after the reset processing has finished.

NOTE: After the data signal is output in one-shot mode, set the REMCDBCTL.REMCRST bit to 1.

bit 0 MODEN
This bit enables the REMC operations.
When this bit = 1, REMC operations are enabled (the operating clock is supplied)
When this bit = 0, REMC operations are disabled (the operating clock is stopped)

NOTE: If the REMCDBCTL.MODEN bit is altered from 1 to 0 while sending data, the data being sent cannot be guaranteed. When setting the REMCDBCTL.MODEN bit to 1 again after that, be sure to write 1 to the REMCDBCTL.REMCRST bit as well.

10.9.3 REMC Data Bit Counter

REG[0724h] REMCDBCNT Register
Default = 0000h RO

DBCNT bits 15-8							
15	14	13	12	11	10	9	8
DBCNT bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 DBCNT bits [15:0] (read-only)
The current value of the 16-bit counter for data signal generation can be read out through these bits.

10.9.4 REMC Data Active Pulse Length

REG[0726h] REMCAPLEN Register
Default = 0000h R/W

APLEN bits 15-8							
15	14	13	12	11	10	9	8
APLEN bits 7-0							
7	6	5	4	3	2	1	0

For this register, the lower byte should be written first followed by the upper byte.

bits 15:0 APLEN bits [15:0]
These bits set the active pulse length of the data signal (high period when the REMCDBCTL.REMOINV bit = 0 or low period when the REMCDBCTL.REMOINV bit = 1). The REMO pin output is set to the active level from the 16-bit counter for data signal generation = 0x0000 and it is inverted to the inactive level when the counter exceeds the REMCAPLEN.APLEN[15:0] bit-setting value. The data signal duty ratio is determined by

this setting and the REMCDBLEN.DBLEN[15:0] bit-setting.
Before this register can be rewritten, the REMCDBCTL.MODEN bit must be set to 1.

10.9.5 REMC Data Bit Length

REG[0728h] REMCDBLEN Register
Default = 0000h R/W

DBLEN bits 15-8							
15	14	13	12	11	10	9	8
DBLEN bits 7-0							
7	6	5	4	3	2	1	0

For this register, the lower byte should be written first followed by the upper byte.

bits 15:0 DBLEN bits [15:0]
These bits set the data length of the data signal (length of one cycle). A data signal cycle begins with the 16-bit counter for data signal generation = 0x0000 and ends when the counter exceeds the REMCDBLEN.DBLEN[15:0] bit-setting value.
Before this register can be rewritten, the REMCDBCTL.MODEN bit must be set to 1.

10.9.6 REMC Status and Interrupt Flag

REG[072Ah] REMCINTF Register
Default = 0000h R/W

n/a					DBCNTRUN	DBLENBSY	APLENBSY
15	14	13	12	11	10	9	8
n/a						DBIF	APIF
7	6	5	4	3	2	1	0

bit 10 DBCNTRUN (read-only)
This bit indicates whether the 16-bit counter for data signal generation is running or not.
When this bit reads 1, the counter is running (counting)
When this bit reads 0, the counter is idle

This bit can be cleared by writing 1 to the REMCDBCTL.REMCRST bit.

bit 9 DBLENBSY (read-only, effective when REMCDBCTL.BUFEN bit = 1)
This bit indicates whether the value written to the REMCDBLEN.DBLEN[15:0] bits is transferred to the REMCDBLEN buffer or not.
When this bit reads 1, the transfer to the REMCDBLEN buffer has not completed.
When this bit reads 0, the transfer to the REMCDBLEN buffer has completed.

While this bit is set to 1, writing to the REMCDBLEN.DBLEN[15:0] bits has no effect.

bit 8 APLENBSY (read-only, effective when REMCDBCTL.BUFEN bit = 1)
This bit indicates whether the value written to the REMCDBLEN.DBLEN[15:0] bits is transferred to the REMCDBLEN buffer or not.
When this bit reads 1 (R): Transfer to the REMCDBLEN buffer has not completed.
When this bit reads 0 (R): Transfer to the REMCDBLEN buffer has completed.

While this bit is set to 1, writing to the REMCDBLEN.DBLEN[15:0] bits has no effect.

bit 1 DBIF
bit 0 APIF
These bits indicate the REMC interrupt cause occurrence status.
When this bit reads 1, the corresponding interrupt has occurred
When this bit reads 0, no interrupt has occurred

When this bit is written 1, the corresponding interrupt flag is cleared
When this bit is written 0, there is no effect

The following shows the correspondence between the bit and interrupt:
REMCINTF.DBIF bit: Compare DB interrupt
REMCINTF.APIF bit: Compare AP interrupt

These interrupt flags are also cleared to 0 when 1 is written to the REMCDBCTL.REMCRST bit.

10.9.7 REMC Interrupt Enable

REG[072Ch] REMCINTE Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
n/a						DBIE	APIE

bit 1 DBIE
bit 0 APIE
These bits enable REMC interrupts.
When this bit = 1, the corresponding interrupt is enabled
When this bit = 0, the corresponding interrupt is disabled

The following shows the correspondence between the bit and interrupt:
REMCINTE.DBIE bit: Compare DB interrupt
REMCINTE.APIE bit: Compare AP interrupt

10.9.8 REMC Carrier Waveform

REG[0730h] REMCCARR Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
CRDTY bits 7-0							
7	6	5	4	3	2	1	0
CRPER bits 7-0							

bits 15:8 CRDTY bits [7:0]
These bits set the high level period of the carrier signal.
The carrier signal is set to high level from the 8-bit counter for carrier generation = 0x00 and it is inverted to low level when the counter exceeds the REMCCARR.CRDTY[7:0] bit-setting value. The carrier signal duty ratio is determined by this setting and the REMCCARR.CRPER[7:0] bit-setting.

bits 7:0 CRPER bits [7:0]
These bits set the carrier signal cycle.
A carrier signal cycle begins with the 8-bit counter for carrier generation = 0x00 and ends when the counter exceeds the REMCCARR.CRPER[7:0] bit-setting value.

10.9.9 REMC Carrier Modulation Control

REG[0732h] REMCCCTL Register							
Default = 0000h							R/W
n/a							OUTINVEN
15	14	13	12	11	10	9	8
n/a							CARREN
7	6	5	4	3	2	1	0

bit 8 OUTINVEN
This bit inverts the REMO output polarity.
When this bit = 1, REMO output polarity is inverted
When this bit = 0, REMO output polarity is non-inverted

bit 0 CARREN
This bit enables carrier modulation.
When this bit = 1, carrier modulation is enabled
When this bit = 0, carrier modulation is disabled (output data signal only)

NOTE: When carrier modulation is disabled, the REMCDBCTL.REMOINV bit should be set to 0.

10.10 DMA Controller (DMAC) Registers

10.10.1 DMAC Status

REG[1000h] DMACSTAT Register							
Default = 00030000h							RO
n/a							
31	30	29	28	27	26	25	24
n/a			CHNLS bits 4-0				
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
STATE bits 3-0				n/a			MSTENSTAT
7	6	5	4	3	2	1	0

bits 20:16 CHNLS bits [4:0] (read-only) = 0_0011b (3h)
These bits show the number of DMAC channels implemented in this IC.
Number of channels implemented = CHNLS + 1 = 4.

bits 7:4 STATE bits [3:0] (read-only)
These bits indicates the DMA transfer status.
0x0: Idle
0x1: Control data is being read
0x2: Transfer source end pointer is being read
0x3: Transfer destination end pointer is being read
0x4: Transfer data is being read
0x5: Transfer data is being written
0x6: Standby for transfer request to be cleared
0x7: Control data is being written
0x8: Transfer has been suspended
0x9: Transfer has completed
0xA: Peripheral scatter-gather transfer is in progress
0xB-0xF: Reserved

bit 0 MSTENSTAT (read-only)
This bit indicates the DMA controller status.
When this bit reads 1, the DMA controller is operating.
When this bit reads 0, the DMA controller is idle.

10.10.2 DMAC Configuration

REG[1004h] DMACCFG Register							
Default = 00000000h							WO
n/a							
31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a							MSTEN
7	6	5	4	3	2	1	0

bit 0 MSTEN (write-only)
This bit enables the DMA controller.
When this bit is written 1, the DMA controller is enabled
When this bit is written 0, the DMA controller is disabled

10.10.3 DMAC Control Data Base Pointer

REG[1008h] DMACCPTR Register							
Default = 00000000h							R/W
CPTR bits 31-24							
31	30	29	28	27	26	25	24
CPTR bits 23-16							
23	22	21	20	19	18	17	16
CPTR bits 15-8							
15	14	13	12	11	10	9	8
CPTR bit 7	CPTR bits 6-0 (read only)						
7	6	5	4	3	2	1	0

bits 31:0 CPTR bits [31:0]
These bits set the leading address of the data structure.
CPTR[31:7] are writable. CPTR[6:0] are read-only.

10.10.4 DMAC Alternate Control Data Base Pointer

REG[100Ch] DMACACPTR Register							
Default = 00000000h							RO
ACPTR bits 31-24							
31	30	29	28	27	26	25	24
ACPTR bits 23-16							
23	22	21	20	19	18	17	16
ACPTR bits 15-8							
15	14	13	12	11	10	9	8
ACPTR bits 7-0							
7	6	5	4	3	2	1	0

bits 31:0 ACPTR bits [31:0] (read-only)
These bits show the alternate data structure base address.

10.10.5 DMAC Software Request

REG[1014h] DMACSWREQ Register							
Default = 00000000h							WO
n/a							
31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a				SWREQ bits 3-0			
7	6	5	4	3	2	1	0

bits 3:0 SWREQ bits [3:0] (write-only)
These bits issue a software DMA transfer request to the corresponding DMA channel (Ch.0–Ch.3). Each bit corresponds to a DMA controller channel.
When a bit is written 1, a software DMA transfer request is issued for the corresponding channel
When a bit is written 0, there is no effect

10.10.6 DMAC Request Mask Set

REG[1020h] DMACRMSET Register							
Default = 00000000h							R/W
n/a							
31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a				RMSET bits 3-0			
7	6	5	4	3	2	1	0

bits 3:0 RMSET bits [3:0]
These bits mask DMA transfer requests from peripheral circuits to the corresponding DMA channel (Ch.0–Ch.3). Each bit corresponds to a DMA controller channel.
When a bit is written 1, the corresponding DMA transfer requests are masked from peripheral circuits
When a bit is written 0, there is no effect
When a bit reads 1, the corresponding DMA transfer requests from peripheral circuits have been disabled.
When a bit reads 0, the corresponding DMA transfer requests from peripheral circuits have been enabled.

10.10.7 DMAC Request Mask Clear

REG[1024h] DMACRMCLR Register
Default = 00000000h — WO

31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
n/a				RMCLR bits 3-0			

bits 3-0 RMCLR[3:0] (write-only)
These bits cancel the mask state of DMA transfer requests from peripheral circuits to the corresponding DMA channel (Ch.0–Ch.3). Each bit corresponds to a DMA controller channel.
When a bit is written 1, cancel mask state of DMA transfer requests from peripheral circuits (the DMACRMSET register is cleared to 0)
When a bit is written 0, there is no effect

10.10.8 DMAC Enable Set

REG[1028h] DMACENSET Register
Default = 00000000h — R/W

31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
n/a				ENSET bits 3-0			

bits 3-0 ENSET[3:0]
These bits enable each DMAC channel (Ch.0–Ch.3). Each bit corresponds to a DMA controller channel.
When a bit is written 1, the corresponding DMAC channel is enabled
When a bit is written 0, there is no effect
When a bit reads 1, the corresponding DMAC channel is enabled
When a bit reads 0, the corresponding DMAC channel is disabled

These bits are cleared after the DMA transfer has completed.

10.10.9 DMAC Enable Clear

REG[102Ch] DMACENCLR Register							
Default = 00000000h							WO
n/a							
31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a				ENCLR bits 3-0			
7	6	5	4	3	2	1	0

bits 3-0 ENCLR[3:0] (write-only)
These bits disable each DMAC channel (Ch.0–Ch.3). Each bit corresponds to a DMA controller channel.
When a bit is written 1, the corresponding DMAC channel is disabled (the DMACENSET register is cleared to 0.)
When a bit is written 0, there is no effect

10.10.10 DMAC Primary-Alternate Set

REG[1030h] DMACPASET Register							
Default = 00000000h							R/W
n/a							
31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a				PASET bits 3-0			
7	6	5	4	3	2	1	0

bits 3-0 PASET[3:0]
These bits enable the alternate data structures for the corresponding DMA channel (Ch.0–Ch.3). Each bit corresponds to a DMA controller channel.
When a bit is written 1, alternate data structure is enabled for the corresponding DMA channel
When a bit is written 0, there is no effect
When a bit reads 1, The alternate data structure has been enabled for the corresponding DMA channel
When a bit reads 0, The primary data structure has been enabled for the corresponding DMA channel

10.10.11 DMAC Primary-Alternate Clear

REG[1034h] DMACPACLR Register
Default = 00000000h — WO

31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
				PACLR bits 3-0			

bits 3-0 PACLR[3:0] (write-only)
These bits disable the alternate data structures for the corresponding DMA channel (Ch.0–Ch.3). Each bit corresponds to a DMA controller channel.
When a bit is written 1, the alternate data structure is disabled for the corresponding DMA channel (The DMACPASET register is cleared to 0.)
When a bit is written 0, there is no effect

10.10.12 DMAC Priority Set

REG[1038h] DMACPRSET Register
Default = 00000000h — R/W

31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
				PRSET bits 3-0			

bits 3-0 PRSET[3:0]
These bits increase the priority of the corresponding DMA channel (Ch.0–Ch.3). Each bit corresponds to a DMA controller channel.
When a bit is written 1, the priority of the corresponding DMA channel is increased
When a bit is written 0, there is no effect
When a bit reads 1, the priority of the corresponding DMA channel = High
When a bit reads 0, the priority of the corresponding DMA channel = Normal

10.10.13 DMAC Priority Clear

REG[103Ch] DMACPRCLR Register
Default = 00000000h — WO

31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
				PRCLR bits 3-0			

bits 3-0 PRCLR[3:0] (write-only)
These bits decrease the priority of the corresponding DMA channel (Ch.0–Ch.3). Each bit corresponds to a DMA controller channel.

When a bit is written 1, the priority of the corresponding DMA channel is decreased (the DMACPRSET register is cleared to 0.)
When a bit is written 0, there is no effect

10.10.14 DMAC Error Interrupt Flag

REG[104Ch] DMACERRIF Register
Default = 00000000h R/W

31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
							ERRIF

bit 0 ERRIF
This bit indicates the DMAC error interrupt cause occurrence status.
When this bit reads 1, the interrupt has occurred
When this bit reads 0, no interrupt has occurred
When this bit is written 1, the interrupt flag is cleared
When this bit is written 0, there is no effect

10.10.15 DMAC Transfer Completion Interrupt Flag

REG[2000h] DMACENDIF Register
Default = 00000000h R/W

31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a				ENDIF bits 3-0			
7	6	5	4	3	2	1	0

bits 3-0 ENDIF[3:0]
These bits indicate the DMA transfer completion interrupt cause occurrence status of the corresponding DMA channel (Ch.0–Ch.3). Each bit corresponds to a DMA controller channel.
When a bit reads 1, the interrupt has occurred for the corresponding DMA channel
When a bit reads 0, no interrupt has occurred
When a bit is written 1, the interrupt flag is cleared the corresponding DMA channel
When a bit is written 0, there is no effect

10.10.16 DMAC Transfer Completion Interrupt Enable Set

REG[2008h] DMACENDIESET Register
Default = 00000000h R/W

n/a							
31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a				ENDIESET bits 3-0			
7	6	5	4	3	2	1	0

bits 3-0 ENDIESET[3:0]
These bits enable DMA transfer completion interrupts to be generated from the corresponding DMA channel (Ch.0–Ch.3). Each bit corresponds to a DMA controller channel.
When a bit is written 1, Enable interrupt
When a bit is written 0, Ineffective
When a bit reads 1, Interrupt has been enabled.
When a bit reads 0, Interrupt has been disabled.

10.10.17 DMAC Transfer Completion Interrupt Enable Clear

REG[200Ch] DMACENDIECLR Register
Default = 00000000h WO

n/a							
31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a				ENDIECLR bits 3-0			
7	6	5	4	3	2	1	0

bits 3-0 ENDIECLR[3:0] (write-only)
These bits disable DMA transfer completion interrupts to be generated from the corresponding DMA channel (Ch.0–Ch.3). Each bit corresponds to a DMA controller channel.
When a bit is written 1, the corresponding DMA transfer completion interrupt is disabled (the DMACENDIESET register is cleared to 0.)
When a bit is written 0, there is no effect

10.10.18 DMAC Error Interrupt Enable Set

REG[2010h] DMACERRIESET Register
Default = 00000000h R/W

n/a							
31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a							ERRIESET
7	6	5	4	3	2	1	0

bit 0 ERRIESET
This bit enables DMA error interrupts.
When this bit is written 1, DMA error interrupts are enabled

When this bit is written 0, there is no effect
When this bit reads 1, DMA error interrupts have been enabled.
When this bit reads 0, DMA error interrupts have been disabled.

10.10.19 DMAC Error Interrupt Enable Clear

REG[2014h] DMACERRIECLR Register
Default = 00000000h — WO

n/a							
31	30	29	28	27	26	25	24
n/a							
23	22	21	20	19	18	17	16
n/a							
15	14	13	12	11	10	9	8
n/a							ERRIECLR
7	6	5	4	3	2	1	0

bit 0 ERRIECLR (write-only)
This bit disables DMA error interrupts.
When a bit is written 1, DMA error interrupts are disabled (the DMACERRIESET register is cleared to 0.)
When a bit is written 0, there is no effect

10.11 Memory Display Controller (MDC) Registers

10.11.1 MDC Display Control

REG[3000h] MDCDISPCTL Register							
Default = 0000h							R/W
DCRDEN	CSPULSE	GSIF[1:0]		DISPGS	RGBORD	ADDRLSB	AUTOCOM
15	14	13	12	11	10	9	8
DISPINVERT	UPDFUNC	SPITYPE	DISPSPI	ROTSEL bits 1-0		VCOMEN	DISPEPD
7	6	5	4	3	2	1	0

bit 15 DCRDEN

For EPD panel 3-wire interface, this bit selects whether or not the D/C bit is driven when reading data bytes from the panel.

0 = D/C bit is not driven when reading data bytes from panel
1 = D/C bit is driven to 1 when reading data bytes from panel

bit 14 CSPULSE

For EPD panel interfaces, this bit selects whether or not the chip-select pin is deasserted and reasserted between data bytes.

0 = chip-select is not deasserted and reasserted between data bytes
1 = chip-select is deasserted and reasserted between data bytes

bits 13-12 GSIF[1:0]

These bits select the panel interface type for EPD (DISPEPD=1) and grayscale (DISPEPD=0, DISPGS=1) displays.

For EPD panel:
00b = Reserved
01b = 3-wire serial
10b = 4-wire serial
11b = Reserved

For grayscale panel:
0xb = 8-Bit Parallel interface
10b = 3-Wire Serial interface
11b = 4-Wire Serial interface

bit 11 DISPGS

When DISPEPD=0, this bit selects grayscale panel or other panel types.
0 = panel is other types. See DISPSPI bit for selection of other types.
1 = panel is grayscale. GSIF[1:0] bits select grayscale panel interface type.

bit 10 RGBORD

This bit configures the bit order of the pixel data sent for SPI 3-bit color panels, or the order of pixel data within a byte (for 1/2/4bpp grayscale format in memory) sent for EPD panels.

For SPI 3-bit color panel:
0 = pixel data is sent RGB (red is sent first)
1 = pixel data is sent BGR (blue is sent first)

For EPD panel:
0 = least significant pixel (LSP) in a byte of 1/2/4bpp pixel format in memory sent first
1 = most significant pixel (MSP) in a byte of 1/2/4bpp pixel format in memory sent first

bit 9 ADDRLSB

This bit configures the order of the Address bits sent (LSB or MSB first) for SPI panels, or the bit order of the pixel data sent for EPD panels.

For SPI panels:
0 = the address bits are sent MSB first
1 = the address bits are sent LSB first

For EPD panels:
0 = least significant bit (LSB) in a pixel in memory is sent first
1 = most significant bit (MSB) in a pixel in memory is sent first

bit 8 AUTOCOM

This bit enables the automatic generation of update commands to send to SPI panel with toggling of internal COM bit. It is used if the SPI panel is configured with EXTMODE=0 and the COM bit needs to be toggled periodically. When this bit is 1, a dummy command is sent to the SPI panel triggered by rising edges of the VCOM signal.

0 = automatic COM bit toggling is disabled
1 = automatic COM bit toggling is enabled

bit 7 DISPINVERT

This bit enables or disables inversion of pixels sent to the display.

0 = pixel bits sent to the display are normal (no inversion)
1 = pixel bits sent to the display are inverted

bit 6 UPDFUNC

For SPI panel (DISPEPD=0, DISPGS=0, DISPSPI = 1), this bit selects the function for SPI display when display update is triggered:

0 = normal function is selected, frame update
1 = command write is selected

When the MDCDISPCTL.UPDFUNC bit = 1 (send Mode + Dummy only)
MDCDISPPRM43.TIM3[7:0] bits = Mode bits
MDCDISPPRM43.TIM4[7:0] bits = Number of Mode bits to send (LSB first)
MDCDISPPRM21.TIM2[7:0] bits = Number of Dummy bits (0) to send

When the MDCDISPCTL.UPDFUNC bit = 0 (send Mode + Address + Pixel data + Dummy)
MDCDISPPRM43.TIM3[7:0] bits = Mode bits
MDCDISPPRM43.TIM4[7:0] bits = Number of Mode bits to send (LSB first)
MDCDISPPRM65.TIM5[7:0] bits = Number of Address bits to send
(The MDCDISPCTL.ADDRLSB bit determines the order of the bits.)
MDCDISPPRM21.TIM2[7:0] bits = Number of Dummy bits (0) to send.

For EPD panel, this bit selects write command sequence or read command sequence:

0 = read command sequence, SDA pin is input function during data bytes
1 = write command sequence, SDA pin is output function during data bytes

bit 5 SPITYPE

This bit selects the type of SPI panel.

0 = the SPI panel is 1-bit black and white

1 = the SPI panel is 3-bit color

bit 4 DISPSPI
When DISPEPD=0 and DISPGS =0, this bit selects panel type between 6-bit color and SPI.

0 = 6-bit color panel is selected
1 = SPI 1-bit black-and-white or 3-bit color is selected (see MDCDISPCTL.SPITYPE bit)

bits 3:2 ROTSEL bits [1:0]
These bits select the rotation of the image from RAM to the SPI panel, 6-bit color panel, or EPD panel during a frame update.

0x0: No rotation (0 degrees)
0x1: 90-degree counterclockwise rotation
0x2: 180-degree counterclockwise rotation
0x3: 270-degree counterclockwise rotation

bit 1 VCOMEN
This bit enables or disables the VCOM and XFRP outputs.

0 = VCOM/XFRP outputs disabled (low)
1 = VCOM/XFRP outputs enabled

When enabled, the frequency of the VCOM/XFRP is determined by the MDCDISPVCOMDIV.DISPVCOMDIV[15:0] bits.

bit 0 DISPEPD
This bit selects the EPD panel or other panel types.

0 = panel is other types. See DISPGS and DISPSPI bits for selection of other types.
1 = panel is EPD. GSIF[1:0] bits select EPD panel interface type.

10.11.2 MDC Display Width

REG[3002h] MDCDISPWIDTH Register
Default = 00B4h R/W

15	14	13	12	11	10	9	8
n/a						DISPWIDTH bits 9-8	
DISPWIDTH bits 7-0							
7	6	5	4	3	2	1	0

bits 9:0 DISPWIDTH bits [9:0]
These bits set the panel display width in pixels.

10.11.3 MDC Display Height

REG[3004h] MDCDISPHEIGHT Register
Default = 00B4h R/W

15	14	13	12	11	10	9	8
n/a						DISPHEIGHT bits 9-8	
DISPHEIGHT bits 7-0							
7	6	5	4	3	2	1	0

bits 9:0 DISPHEIGHT bits [9:0]
These bits set the panel display height in pixels.

10.11.4 MDC VCOM Clock Divider

REG[3006h] MDCDISPVCOMDIV Register							
Default = 0222h							R/W
DISPVCOMDIV bits 15-8							
15	14	13	12	11	10	9	8
DISPVCOMDIV bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 DISPVCOMDIV bits [15:0]
These bits set the VCOM clock division ratio to determine the VCOM/XFRP frequency.

VCOM/XFRP frequency = (OSC1 frequency) / [4 x (DISPVCOMDIV[15:0] + 1)]

10.11.5 MDC Display Clock Divider

REG[3008h] MDCDISPCLKDIV Register							
Default = 0404h							R/W
TIM0 bits 7-0							
15	14	13	12	11	10	9	8
CLKDIV bits 7-0							
7	6	5	4	3	2	1	0

This register controls the frequency of the output clocking rate during a panel frame update.

bits 15:8 TIM0 bits [7:0]
These bits set timing parameter 0.

For 6-bit color panel:
t0 = VCK to HST rise and VCK to VST fall = (TIM0[7:0] + 1) panel timing units (T)

For SPI panel:
SCS rise to first SI data = (TIM0[7:0] + 1) SCLK periods (T)

For grayscale and EPD panels:
Command byte value

bits 7:0 CLKDIV bits [7:0]
These bits set the MDC system clock divider value.

For 6-bit color panel:
(Setting these bits determines the panel timing unit.)
Panel timing unit T = (CLKDIV[7:0] + 1) / (System clock frequency [Hz])

For SPI panel:
(Setting these bits determines the SPI clock frequency.)
SPI clock frequency = (System clock frequency [Hz]) / (CLKDIV[7:0] + 1)

NOTE: The CLKDIV[7:0] bits should not be programmed to 0.

10.11.6 MDC Display Parameters 1 and 2

REG[300Ah] MDCDISPPRM21 Register							
Default = 0004h							R/W
TIM2 bits 7-0							
15	14	13	12	11	10	9	8
TIM1 bits 7-0							
7	6	5	4	3	2	1	0

bits 15:8 TIM2 bits [7:0]
These bits set timing parameter 2.

For 6-bit color panel:
t2 = HST rise to HCK rise = (TIM2[7:0] + 1) panel timing units (T)

For SPI panel:
Number of dummy clocks for "data transfer period" = (TIM2[7:0] + 1) SCLK periods (T)

For grayscale panels:
Parameter 2 value for command sequence.

bits 7:0 TIM1 bits [7:0]
These bits set timing parameter 1.

For 6-bit color panel:
t1 = VST rise to VCK rise = (TIM1[7:0] + 1) panel timing units (T)

For SPI panel:
Last SI data to SCS fall = (TIM1[7:0] + 1) SCLK periods (T)

For grayscale panels:
Dummy byte value for frame update, Parameter 1 value for command sequence.

10.11.7 MDC Display Parameters 3 and 4

REG[300Ch] MDCDISPPRM43 Register							
Default = 0404h							R/W
TIM4 bits 7-0							
15	14	13	12	11	10	9	8
TIM3 bits 7-0							
7	6	5	4	3	2	1	0

bits 15:8 TIM4 bits [7:0]
These bits set timing parameter 4.

For 6-bit color panel:
t4 = VCK to ENB rise = [(TIM4[7:0] * (t3 + t7)) + t0 + t2] =
[(TIM4[7:0] * (TIM3[7:0] + TIM7]7:0] + 2)) + TIM0[7:0] + TIM2[7:0] + 2] panel timing units (T)

For SPI panel:
Number of Mode bits to send minus 1.

For grayscale panels:
Parameter 4 value for command sequence.

bits 7:0 TIM3 bits [7:0]
These bits set timing parameter 3.

For 6-bit color panel:
t3 = data to HCK rise/fall = (TIM3[7:0] + 1) panel timing units (T)

For SPI panel:
Mode bits values. TIM3[7:0] = M[7:0]. TIM3[1] has no effect on the M[1] bit, as M[1] is always 0.

For grayscale panels:
Parameter 3 value for command sequence.

10.11.8 MDC Display Parameters 5 and 6

REG[300Eh] MDCDISPPRM65 Register
Default = 0404h R/W

TIM6 bits 7-0							
15	14	13	12	11	10	9	8
TIM5 bits 7-0							
7	6	5	4	3	2	1	0

bits 15:8 TIM6 bits [7:0]
These bits set timing parameter 6.

For 6-bit color panel:
t6 = Trigger to XRST rise, XRST rise to VST rise, data transfer end to XRST fall = (TIM6[7:0] + 1) panel timing units (T)

For SPI panel:
SCS fall to SCS rise minimum time = (TIM6[7:0] + 2) SCLK periods (T)

For grayscale panels:
Parameter 6 value for command sequence.

bits 7:0 TIM5 bits [7:0]
These bits set timing parameter 5.

For 6-bit color panel:
t5 = ENB width = [TIM5[7:0] * (t3 + t7)] =
[TIM5[7:0] * (TIM3[7:0] + TIM7[7:0] + 2)] panel timing units (T)

For SPI panel:
Number of Address bits to send minus 1.

For grayscale panels:
Parameter 5 value for command sequence.

10.11.9 MDC Display Parameters 7 and 8

REG[3010h] MDCDISPPRM87 Register
Default = 0400h R/W

15	14	13	12	11	10	9	8
TIM8 bits 7-0							
7	6	5	4	3	2	1	0
TIM7 bits 7-0							

bits 15:8 TIM8 bits [7:0]
These bits set timing parameter 8.

For 6-bit color panel:
t8 = horizontal end to next VCK edge = (TIM8[7:0] + 1) panel timing units (T)

For grayscale panels:
Parameter 8 value for command sequence.

bits 7:0 TIM7 bits [7:0]
These bits set timing parameter 7.

For 6-bit color panel:
t7 = HCK rise/fall to data and HST fall = (TIM7[7:0] + 1) panel timing units (T)

For grayscale panels:
Parameter 7 value for command sequence.

10.11.10 MDC Display Update Start Line

REG[3012h] MDCDISPSTARTY Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						STARTY bits 9-8	
7	6	5	4	3	2	1	0
STARTY bits 7-0							

bits 9:0 STARTY bits [9:0]
These bits set the starting line of image for display update. These bits and the MDCDISPENDY.ENDY[9:0] bits specify which block of lines to update on the panel. The first image line is line 0.

10.11.11 MDC Display Update End Line

REG[3014h] MDCDISPENDY Register
Default = 00B3h R/W

15	14	13	12	11	10	9	8
n/a						ENDY bits 9-8	
7	6	5	4	3	2	1	0
ENDY bits 7-0							

bits 9:0 ENDY bits [9:0]
These bits set the ending line of image for display update. These bits and the MDCDISPSTARTY.STARTY[9:0] bits specify which block of lines to update on the panel. The first image line is line 0.

10.11.12 MDC Display Frame Buffer Stride

REG[3016h] MDCDISPSTRIDE Register
Default = 00B4h R/W

15	14	13	12	11	10	9	8
n/a						DISPSTRIDE bits 9-8	
7	6	5	4	3	2	1	0
DISPSTRIDE bits 7-0							

bits 9:0 DISPSTRIDE bits [9:0]
These bits set the display frame buffer stride (number of pixels per line). The stride can be greater than the value set to the MDCDISPWIDTH.DISPWIDTH[9:0] bits to allow for a wider image in RAM and panning.

10.11.13 MDC Display Frame Buffer Base Address 0

REG[3018h] MDCDISPFRMBUFF0 Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
FRMBUFFADDR bits 15-8							
7	6	5	4	3	2	1	0
FRMBUFFADDR bits 7-0							

bits 15:0 FRMBUFFADDR bits [15:0]
These bits set the lower 16 bits of the display frame buffer base address.

10.11.14 MDC Display Frame Buffer Base Address 1

REG[301Ah] MDCDISPFRMBUFF1 Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
FRMBUFFADDR bits 31-24							
7	6	5	4	3	2	1	0
FRMBUFFADDR bits 23-16							

bits 15:0 FRMBUFFADDR bits [31:16]
These bits set the upper 16 bits of the display frame buffer base address.

10.11.15 MDC Trigger Control

REG[301Ch] MDCTRIGCTL Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						GSBPP[1:0]	
7	6	5	4	3	2	1	0
NPARAM[3:0]				n/a		UPDTRIG	GFXTRIG

bits 9-8 GSBPP[1:0]
These bits specify the number of bits per pixel (BPP) for grayscale panel pixel format.
00b = 1BPP
01b = 2BPP
10b = 4BPP
11b = 8BPP

bits 7-4 NPARAM[3:0]
These bits specify the number of parameter bytes to send for grayscale panel Command Write function. The parameter byte values should be written into the MDCDISPPRMxx.TIMn[7:0] registers.

bit 1 UPDTRIG
This bit triggers the display update function.
When this bit is written 1, the display update is triggered
When this bit is written 0, there is no effect
When this bit reads 1, the display update function is running
When this bit reads 0, the display update function is idle

This bit retains 1 until the graphics function is finished. At the end of the display update function execution, this bit reverts to 0 and the MDCINTCTL.UPDIF bit is set to 1.

bit 0 GFXTRIG
This bit triggers the graphics acceleration function specified in the MDCGFXCTL.GFXFUNC[2:0] bits.
When this bit is written 1, the graphics function is triggered
When this bit is written 0, there is no effect
When this bit reads 1, the graphics function is running
When this bit reads 0, the graphics function is idle

This bit retains 1 until the graphics function is finished. At the end of the graphics function execution, this bit reverts to 0 and the MDCINTCTL.GFXIF bit is set to 1.

10.11.16 MDC Interrupt Control

REG[301Eh] MDCINTCTL Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a					VCNTIE	UPDIE	GFXIE
7	6	5	4	3	2	1	0
n/a					VCNTIF	UPDIF	GFXIF

bit 10 VCNTIE
This bit enables the VCNT match interrupt for 6-bit color panel.
When this bit = 1: Enable interrupt
When this bit = 0: Disable interrupt

bit 9 UPDIE
This bit enables the display update interrupt.
When this bit = 1: Enable interrupt
When this bit = 0: Disable interrupt

bit 8 GFXIE
This bit enables the graphics interrupt.
When this bit = 1: Enable interrupt
When this bit = 0: Disable interrupt

bit 2 VCNTIF
bit 1 UPDIF
bit 0 GFXIF
These bits indicates the MDC interrupt cause occurrence status.
When a bit reads 1, the corresponding interrupt has occurred
When a bit reads 0, no interrupt has occurred
When a bit is written 1, the corresponding interrupt flag is cleared
When a bit is written 0, there is no effect

10.11.17 MDC Graphics Control

REG[3020h] MDCGFXCTL Register							
Default = 0000h							R/W
CPYNEGY	GPYNEGX	SHEARNEGY	SHEARNEGX	FILLEN	BITMAPFMT	BITMAPEN	n/a
15	14	13	12	11	10	9	8
ALPHAVAL bits 1-0		ALPHAOVRRD	n/a		GFXFUNC bits 2-0		
7	6	5	4	3	2	1	0

bit 15 CPYNEGY

For the copy functions, this bit selects whether or not the Y coordinate value of the source pixel (with respect to the center of transformation of the source window) should be negated (make negative) to create the effect of "flip around the X axis."
When this bit = 1, the Y coordinate of the source pixel is negated
When this bit = 0, the Y coordinate of the source pixel is normal

bit 14 CPYNEGX

For the copy functions, this bit selects whether or not the X coordinate value of the source pixel (with respect to the center of transformation of the source window) should be negated (make negative) to create the effect of "flip around the Y axis."
When this bit = 1, the X coordinate of the source pixel is negated
When this bit = 0, the X coordinate of the source pixel is normal

bit 13 SHEARNEGY

For the copy with shearing function, this bit selects whether or not the vertical shear value is negated (made negative).
When this bit = 1, the vertical shear value is negated
When this bit = 0, the vertical shear value is normal (positive)

bit 12 SHEARNEGX

For the copy with shearing function, this bit selects whether or not the horizontal shear value is negated (made negative).
When this bit = 1, the horizontal shear value is negated
When this bit = 0, the horizontal shear value is normal (positive)

bit 11 FILLEN

This bit enables/disables fill option for drawing and copying functions.
When this bit = 1, fill is enabled
When this bit = 0, fill is disabled

Drawing functions

For rectangle and ellipse drawing functions, this bit selects whether the function is filled or unfilled.

Copying functions

For the copying functions, when this bit is enabled, the source image pixels are ignored and the destination image is written with the color specified in the MDCGFXCOLOR.COLOR[5:0] bits.

bit 10 BITMAPFMT

This bit selects 1-bit or 2-bit bitmap format for copying functions. The 2-bit bitmap format is only applicable to 6-bit color panel.
When this bit = 1, 2-bit bitmap format is selected
When this bit = 0, 1-bit bitmap format is selected

bit 9 BITMAPEN

This bit specifies whether the source for copying functions is an image or a bitmap.
When this bit = 1, the source is a bitmap
When this bit = 0, the source is an image

bit 7:6 ALPHAVAL bits [1:0]

These bits specify the override alpha value to use when alpha value override is enabled (MDCGFXCTL.ALPHAOVRRD bit = 1).

bit 5 ALPHAOVRRD

This bit enables/disables uniform alpha value override for copying functions for 6-bit color and 2/4/8BPP grayscale with alpha enabled pixel formats.
When this bit = 1: Alpha value override enabled
When this bit = 0: Alpha value override disabled

Normally, the alpha-blending value for copying functions is provide in the source pixel. When alpha value override is enabled, the alpha value set in MDCGFXCTL.ALPHAVAL[1:0] is used for all pixels.

The following are the uniform alpha blending values for all pixels when ALPHAOVRRD=1 for the applicable pixel formats:

6-bit Color:	2-bit alpha value = ALPHAVAL[1:0]
2BPP Grayscale:	2-bit alpha value = ALPHAVAL[1:0]
4BPP Grayscale:	4-bit alpha value = {ALPHAVAL[1:0], ALPHAVAL[1:0]}
8BPP Grayscale:	8-bit alpha value = {ALPHAVAL[1:0], ALPHAVAL[1:0], ALPHAVAL[1:0], ALPHAVAL[1:0]}

bits 2:0 GFXFUNC bits [2:0]

These bits specify the graphics accelerator function to execute when the MDCTRIGCTL.GFXTRIG bit is set to 1.

0x0: COPYROTSCALE – image/bitmap copy with rotation and scaling
0x1: COPYHVSHEAR – image/bitmap copy with horizontal and vertical shear
0x2: RECTDRAW – rectangle (filled or unfilled) draw
0x3: LINEDRAW – line draw with thickness
0x4: ELLIPDRAW – Ellipse (filled or unfilled) draw
0x5: Reserved
0x6: Reserved
0x7: Reserved

10.11.18 MDC Input X Coordinate

REG[3022h] MDCGFXIXCENTER Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						IXCENTER bits 9-8	
7	6	5	4	3	2	1	0
IXCENTER bits 7-0							

bits 9:0 IXCENTER bits [9:0]
These bits specify an X coordinate depending on the graphics accelerator function selected. The following shows the usage of these bits:

MDCGFXCTL.GFXFUNC[2:0]	IXCENTER[9:0] Function
0x0 (COPYROTSCALE)	X coordinate of center of rotation/scaling of source Window relative to top-left corner of source window
0x1 (COPYHVSHEAR)	X coordinate of center of horizontal/vertical shearing in source window relative to top-left corner of source window
0x2 (RECTDRAW)	X coordinate of top-left corner of rectangle
0x3 (LINEDRAW)	X coordinate of starting point of line
0x4 (ELLIPDRAW)	X coordinate of center of ellipse

10.11.19 MDC Input Y Coordinate

REG[3024h] MDCGFXIYCENTER Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						IYCENTER bits 9-8	
7	6	5	4	3	2	1	0
IYCENTER bits 7-0							

bits 9:0 IYCENTER bits [9:0]
These bits specify a Y coordinate depending on the graphics accelerator function selected. The following shows the usage of these bits:

MDCGFXCTL.GFXFUNC[2:0]	IYCENTER[9:0] Function
0x0 (COPYROTSCALE)	Y coordinate of center of rotation/scaling of source Window relative to top-left corner of source window
0x1 (COPYHVSHEAR)	Y coordinate of center of horizontal/vertical shearing in source window relative to top-left corner of source window
0x2 (RECTDRAW)	Y coordinate of top-left corner of rectangle
0x3 (LINEDRAW)	Y coordinate of starting point of line
0x4 (ELLIPDRAW)	Y coordinate of center of ellipse

10.11.20 MDC Input Width

REG[3026h] MDCGFXIWIDTH Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						IWIDTH bits 9-8	
7	6	5	4	3	2	1	0
IWIDTH bits 7-0							

bits 9:0 IWIDTH bits [9:0]
These bits specify a width or thickness depending on the graphics accelerator function selected.

The following shows the usage of these bits:

MDCGFXCTL.GFXFUNC[2:0]	IWIDTH[9:0] Function
0x0 (COPYROTSCALE)	Width of source window
0x1 (COPYHVSHEAR)	Width of source window
0x2 (RECTDRAW)	Vertical line thickness for unfilled rectangle drawing
0x3 (LINEDRAW)	Line thickness
0x4 (ELLIPDRAW)	Thickness at X-axis crossings for unfilled ellipse drawing

10.11.21 MDC Input Height

REG[3028h] MDCGFXIHEIGHT Register

Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						IHEIGHT bits 9-8	
7	6	5	4	3	2	1	0
IHEIGHT bits 7-0							

bits 9:0 IHEIGHT bits [9:0]

These bits specify a height or thickness depending on the graphics accelerator function selected. The following shows the usage of these bits:

MDCGFXCTL.GFXFUNC[2:0]	IHEIGHT[9:0] Function
0x0 (COPYROTSCALE)	Height of source window
0x1 (COPYHVSHEAR)	Height of source window
0x2 (RECTDRAW)	Horizontal line thickness for unfilled rectangle drawing
0x3 (LINEDRAW)	-
0x4 (ELLIPDRAW)	Thickness at Y-axis crossings for unfilled ellipse drawing

10.11.22 MDC Output X Coordinate

REG[302Ah] MDCGFXOXCENTER Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						OXCENTER bits 9-8	
7	6	5	4	3	2	1	0
OXCENTER bits 7-0							

bits 9:0 OXCENTER bits [9:0]

These bits specify an X coordinate or radius depending on the graphics accelerator function selected. The following shows the usage of these bits:

MDCGFXCTL.GFXFUNC[2:0]	OXCENTER[9:0] Function
0x0 (COPYROTSCALE)	X coordinate of center of rotation/scaling location in destination window relative top-left corner of destination window
0x1 (COPYHVSHEAR)	X coordinate of center of horizontal/vertical shearing location in destination window relative to the top-left corner of destination window
0x2 (RECTDRAW)	X coordinate of bottom-right corner of rectangle
0x3 (LINEDRAW)	X coordinate of ending point in line
0x4 (ELLIPDRAW)	X radius of ellipse

10.11.23 MDC Output Y Coordinate

REG[302Ch] MDCGFXOYCENTER Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						OYCENTER bits 9-8	
7	6	5	4	3	2	1	0
OYCENTER bits 7-0							

bits 9:0 OYCENTER bits [9:0]

These bits specify an Y coordinate or radius depending on the graphics accelerator function selected. The following shows the usage of these bits:

MDCGFXCTL.GFXFUNC[2:0]	OYCENTER[9:0] Function
0x0 (COPYROTSCALE)	Y coordinate of center of rotation/scaling location in destination window relative top-left corner of destination window
0x1 (COPYHVSHEAR)	Y coordinate of center of horizontal/vertical shearing location in destination window relative to the top-left corner of destination window
0x2 (RECTDRAW)	Y coordinate of bottom-right corner of rectangle
0x3 (LINEDRAW)	Y coordinate of ending point in line
0x4 (ELLIPDRAW)	Y radius of ellipse

10.11.24 MDC Output Width

REG[302Eh] MDCGFXOWIDTH Register
Default = 0000h R/W

n/a						OWIDTH bits 9-8	
15	14	13	12	11	10	9	8
OWIDTH bits 7-0							
7	6	5	4	3	2	1	0

bits 9:0 OWIDTH bits [9:0]

These bits specify a width depending on the graphics accelerator function selected. The following shows the usage of these bits:

MDCGFXCTL.GFXFUNC[2:0]	**OWIDTH[9:0] Function**
0x0 (COPYROTSCALE)	Width of destination window
0x1 (COPYHVSHEAR)	Width of destination window

10.11.25 MDC Output Height

REG[3030h] MDCGFXOHEIGHT Register
Default = 0000h R/W

n/a						OHEIGHT bits 9-8	
15	14	13	12	11	10	9	8
OHEIGHT bits 7-0							
7	6	5	4	3	2	1	0

bits 9:0 OHEIGHT bits [9:0]

These bits specify a height depending on the graphics accelerator function selected. The following shows the usage of these bits:

MDCGFXCTL.GFXFUNC[2:0]	**OHEIGHT[9:0] Function**
0x0 (COPYROTSCALE)	Height of destination window
0x1 (COPYHVSHEAR)	Height of destination window

10.11.26 MDC X Left Scale

REG[3032h] MDCGFXXLSCALE Register
Default = 0000h R/W

n/a		XLSCALE bits 13-8					
15	14	13	12	11	10	9	8
XLSCALE bits 7-0							
7	6	5	4	3	2	1	0

bits 13:0 XLSCALE bits [13:0]

These bits are used by the COPYROTSCALE function (image/bitmap copy with rotation and scaling) to specify the scaling factor to be applied to the left half of the source window pixels relative to the center of transformation when copying to the destination window. The scaling ratio is XLSCALE[13:0]/256.

10.11.27 MDC X Right Scale

REG[3034h] MDCGFXXRSCALE Register							
Default = 0000h							R/W
n/a		XRSCALE bits 13-8					
15	14	13	12	11	10	9	8
XRSCALE bits 7-0							
7	6	5	4	3	2	1	0

bits 13:0 XRSCALE bits [13:0]

These bits are used by the COPYROTSCALE function (image/bitmap copy with rotation and scaling) to specify the scaling factor to be applied to the right half of the source window pixels relative to the center of transformation when copying to the destination window. The scaling ratio is XRSCALE[13:0]/256.

10.11.28 MDC Y Top Scale

REG[3036h] MDCGFXYTSCALE Register							
Default = 0000h							R/W
n/a		YTSCALE bits 13-8					
15	14	13	12	11	10	9	8
YTSCALE bits 7-0							
7	6	5	4	3	2	1	0

bits 13:0 YTSCALE bits [13:0]

These bits are used by the COPYROTSCALE function (image/bitmap copy with rotation and scaling) to specify the scaling factor to be applied to the top half of the source window pixels relative to the center of transformation when copying to the destination window. The scaling ratio is YTSCALE[13:0]/256.

10.11.29 MDC Y Bottom Scale

REG[3038h] MDCGFXYBSCALE Register							
Default = 0000h							R/W
n/a		YBSCALE bits 13-8					
15	14	13	12	11	10	9	8
YBSCALE bits 7-0							
7	6	5	4	3	2	1	0

bits 13:0 YBSCALE bits [13:0]

These bits are used by the COPYROTSCALE function (image/bitmap copy with rotation and scaling) to specify the scaling factor to be applied to the bottom half of the source window pixels relative to the center of transformation when copying to the destination window. The scaling ratio is YBSCALE[13:0]/256.

10.11.30 MDC X/Y Shear

REG[303Ah] MDCGFXSHEAR Register							
Default = 0000h							R/W
n/a	YSHEAR bits 6-0						
15	14	13	12	11	10	9	8
n/a	XSHEAR bits 6-0						
7	6	5	4	3	2	1	0

bits 14:8 YSHEAR bits [6:0]
These bits are used by the COPYHVSHEAR function (image/bitmap copy with horizontal/vertical shear) to specify the vertical shearing factor to be applied to the source window pixels relative to the center of transformation when copying to the destination window. The shearing ratio is YSHEAR[6:0]/32. A shearing ratio of 0 means no shearing.

bits 6:0 XSHEAR bits [6:0]
These bits are used by the COPYHVSHEAR function (image/bitmap copy with horizontal/vertical shear) to specify the horizontal shearing factor to be applied to the source window pixels relative to the center of transformation when copying to the destination window. The shearing ratio is XSHEAR[6:0]/32. A shearing ratio of 0 means no shearing.

10.11.31 MDC Rotation

REG[303Ch] MDCGFXROTVAL Register							
Default = 0000h							R/W
n/a							ROTVAL bit 8
15	14	13	12	11	10	9	8
ROTVAL bits 7-0							
7	6	5	4	3	2	1	0

bits 8:0 ROTVAL bits [8:0]
These bits are used by the COPYROTSCALE function (image/bitmap copy with rotation and scaling) to specify the angle of rotation of the source window pixels about the center of transformation when copying to the destination window. The angle in degrees is (ROTVAL[8:0] × 360) / 512 in the counterclockwise direction.

10.11.32 MDC Color

REG[303Eh] MDCGFXCOLOR Register							
Default = 0003h							R/W
n/a							
15	14	13	12	11	10	9	8
COLOR bits 7-0							
7	6	5	4	3	2	1	0

bits 7:0 COLOR bits [7:0]
These bits specify fill color for copying functions with fill enabled (MDCGFXCTL.FILLEN bit = 1) or pen color for drawing functions. The following shows the usage of these bits depending on the panel type selected:

Panel Type	Color Specification
SPI 1-bit BW	COLOR[0] is the black-and-white pixel color.
SPI 3-bit color	COLOR[2:0] is the RGB pixel color.
6-bit color	COLOR[5:0] is the RRGGBB pixel color. COLOR[7:6] is the alpha value for drawing functions to specify the alpha-blending between COLOR[5:0] and the background/destination pixel color.

10.11.33 MDC Source Window Base Address 0

REG[3040h] MDCGFXIBADDR0 Register							
Default = 0000h							R/W
IBASEADDR bits 15-8							
15	14	13	12	11	10	9	8
IBASEADDR bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 IBASEADDR bits [15:0]
These bits set the lower 16 bits of the source window base address for copying functions.

10.11.34 MDC Source Window Base Address 1

REG[3042h] MDCGFXIBADDR1 Register							
Default = 0000h							R/W
IBASEADDR bits 31-24							
15	14	13	12	11	10	9	8
IBASEADDR bits 23-16							
7	6	5	4	3	2	1	0

bits 15:0 IBASEADDR bits [31:16]
These bits set the upper 16 bits of the source window base address for copying functions.

10.11.35 MDC Destination Window Base Address 0

REG[3044h] MDCGFXOBADDR0 Register							
Default = 0000h							R/W
OBASEADDR bits 15-8							
15	14	13	12	11	10	9	8
OBASEADDR bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 OBASEADDR bits [15:0]
These bits set the lower 16 bits of the destination window base address for copying and drawing functions.

10.11.36 MDC Destination Window Base Address 1

REG[3046h] MDCGFXOBADDR1 Register							
Default = 0000h							R/W
OBASEADDR bits 31-24							
15	14	13	12	11	10	9	8
OBASEADDR bits 23-16							
7	6	5	4	3	2	1	0

bits 15:0 OBASEADDR bits [31:16]
These bits set the upper 16 bits of the destination window base address for copying and drawing functions.

10.11.37 MDC Source Image Stride

REG[3048h] MDCGFXISTRIDE Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						ISTRIDE bits 9-8	
7	6	5	4	3	2	1	0
ISTRIDE bits 7-0							

bits 9:0 ISTRIDE bits [9:0]
These bits are used by the COPYROTSCALE (image/bitmap copy with rotation and scaling) and COPYHVSHEAR (image/bitmap copy with horizontal/vertical shear) functions to specify the source image stride (number of pixels per line). The value set to these bits can be greater than or equal to the MDCGFXIWIDTH.IWIDTH[9:0] bits to support source images which are wider/larger than the source window to support panning.

10.11.38 MDC Destination Image Stride

REG[304Ah] MDCGFXOSTRIDE Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a						OSTRIDE bits 9-8	
7	6	5	4	3	2	1	0
OSTRIDE bits 7-0							

bits 9:0 OSTRIDE bits [9:0]
These bits are used by the drawing and copying functions to specify the destination image stride (number of pixels per line). The value set to these bits can be greater than or equal to the MDCGFXOWIDTH. OWIDTH[9:0] bits to support destination images which are wider/larger than the destination window to support panning.

10.11.39 MDC Output Window Left Edge

REG[304Ch] MDCGFXOWLEFT Register
Default = 0000h RO

15	14	13	12	11	10	9	8
n/a						OWLEFT bits 9-8	
7	6	5	4	3	2	1	0
OWLEFT bits 7-0							

bits 9:0 OWLEFT bits [9:0] (read-only)
These read-only bits provide the X coordinate, relative to the center of transformation of the destination window, of the left edge of the calculated output window from the last copy function.

10.11.40 MDC Output Window Right Edge

REG[304Eh] MDCGFXOWRIGHT Register
Default = 0000h RO

15	14	13	12	11	10	9	8
n/a						OWRIGHT bits 9-8	
7	6	5	4	3	2	1	0
OWRIGHT bits 7-0							

bits 9:0 OWRIGHT bits [9:0] (read-only)
These read-only bits provide the X coordinate, relative to the center of transformation of the

destination window, of the right edge of the calculated output window from the last copy function.

10.11.41 MDC Output Window Top Edge

REG[3050h] MDCGFXOWTOP Register
Default = 0000h RO

15	14	13	12	11	10	9	8
n/a						OWTOP bits 9-8]	
7	6	5	4	3	2	1	0
OWTOP bits 7-0							

bits 9:0 OWTOP bits [9:0] (read-only)
These read-only bits provide the Y coordinate, relative to the center of transformation of the destination window, of the top edge of the calculated output window from the last copy function.

10.11.42 MDC Output Window Bottom Edge

REG[3052h] MDCGFXOWBOT Register
Default = 0000h RO

15	14	13	12	11	10	9	8
n/a						OWBOT bits 9-8	
7	6	5	4	3	2	1	0
OWBOT bits 7-0							

bits 9:0 OWBOT bits [9:0] (read-only)
These read-only bits provide the Y coordinate, relative to the center of transformation of the destination window, of the bottom edge of the calculated output window from the last copy function.

10.11.43 MDC Display Parameters 9 and 10

REG[3054h] MDCDISPPRM109 Register
Default = 0202h R/W

15	14	13	12	11	10	9	8
TIM10 bits 7-0							
7	6	5	4	3	2	1	0
TIM9 bits 7-0							

bits 15:8 TIM10 bits [7:0]
These bits set timing parameter 10.

For 6-bit color panel:
t9 = Number of additional HCK counts after end of line pixel data = TIM10[7:0]

For grayscale panels:
Parameter 10 value for command sequence.

bits 7:0 TIM9 bits [7:0]
These bits set timing parameter 9.

For 6-bit color panel:
t10 = HCK count for start of pixel data = TIM9[7:0]

For grayscale panels:
Parameter 9 value for command sequence.

10.11.44 MDC Display Parameters 11 and 12

REG[3056h] MDCDISPPRM1211 Register
Default = 0202h R/W

TIM12 bits 7-0							
15	14	13	12	11	10	9	8
TIM11 bits 7-0							
7	6	5	4	3	2	1	0

bits 15:8 TIM12 bits [7:0]
These bits set timing parameter 12.

For 6-bit color panel:
t12 = Number of additional VCK counts after end of pixel data = TIM12[7:0]

For grayscale panels:
Parameter 12 value for command sequence.

bits 7:0 TIM11 bits [7:0]
These bits set timing parameter 11.

For 6-bit color panel:
t11 = VCK count for start of pixel data (MSB) = TIM11[7:0]

For grayscale panels:
Parameter 11 value for command sequence.

10.11.45 MDC Display Parameters 13 and 14

REG[3058h] MDCDISPPRM1413 Register
Default = 0202h R/W

n/a							
15	14	13	12	11	10	9	8
TIM13 bits 7-0							
7	6	5	4	3	2	1	0

bits 7:0 TIM13 bits [7:0]
These bits set timing parameter 13.

For 6-bit color panel:
t13 = VCK high/low width for fast VCK mode = (TIM13[7:0]+1) panel timing units (T)

For grayscale panels:
Parameter 13 value for command sequence.

10.11.46 MDC Display Control 2

REG[305Ah] MDCDISPCTL2 Register							
Default = 0202h							R/W
MDC_REV bits 7-0							
15	14	13	12	11	10	9	8
n/a		GSALPHA	CSPOL	VSTFALL	HSTFALL	ENBPHASE	FASTVCK
7	6	5	4	3	2	1	0

bits 15:8 MDC_REV bits [7:0] (read-only)
These bits indicate the version of the MDC hardware.
The value is 0x02.

bit 5 GSALPHA
Alpha channel enable/disable for black-and-white and grayscale pixel formats.
When this bit = 0, black-and-white and grayscale pixel formats do not have alpha channel.
When this bit = 1, black-and-white and grayscale pixel formats have alpha channel.

bit 4 CSPOL
Polarity of chip-select output for panels which have a chip-select signal.
When this bit = 0, chip select output for panels is normal polarity
When this bit = 1, chip select output for panels is inverted polarity

bit 3 VSTFALL
This bit sets the VST falling edge phase.
When this bit = 0, VST falling edge is after VCK fall
When this bit = 1, VST falling edge is after VCK rise

bit 2 HSTFALL
This bit sets the HST falling edge phase.
When this bit = 0, HST falling edge is after HCK fall
When this bit = 1, HST falling edge is after HCK rise

bit 1 ENBPHASE
This bit sets the ENB start phase.
When this bit = 0, ENB pulses start on first LSB pixel data
When this bit = 1, ENB pulses start on first MSB pixel data

bit 0 FASTVCK
This bit sets the mode of the VCK pulses for partial updates.
When this bit = 0, VCK pulses are normal
When this bit = 1, VCK pulse are fast for the lines that are not updated (ENB=0)

10.11.47 MDC VCK Count Compare

REG[305Ch] MDCVCNTCOMP Register							
Default = 0FFFh							R/W
n/a				VCNTCOMP bits 11-8			
15	14	13	12	11	10	9	8
VCNTCOMP bits 7-0							
7	6	5	4	3	2	1	0

bits 11:0 VCNTCOMP bits [11:0]
VCK count compare value for generating the VCNTIF flag. VCNTIF flag is set when the internal VCK count matches VCNTCOMP[11:0].

10.11.48 MDC VCK Count

REG[305Eh] MDCVCNT Register							
Default = 0FFFh							RO
n/a				VCNT bits 11-8			
15	14	13	12	11	10	9	8
VCNT bits 7-0							
7	6	5	4	3	2	1	0

bits 11:0 VCNT bits [11:0] (read-only)
This registers provides the current value of the internal VCK counter.

10.11.49 MDC Scratchpad A 0

REG[3060h] MDCSCRATCHA0 Register							
Default = 0000h							R/W
SCRATCHA bits 15-8							
15	14	13	12	11	10	9	8
SCRATCHA bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 SCRATCHA bits [15:0]
These bits are the lower 16 bits of the 32-bit scratchpad A register.

10.11.50 MDC Scratchpad A 1

REG[3062h] MDCSCRATCHA1 Register							
Default = 0000h							R/W
SCRATCHA bits 31-24							
15	14	13	12	11	10	9	8
SCRATCHA bits 23-16							
7	6	5	4	3	2	1	0

bits 15:0 SCRATCHA bits [31:16]
These bits are the upper 16 bits of the 32-bit scratchpad A register.

10.11.51 MDC Event Processor Base Address 0

REG[3064h] MDCEPBASEADDR0 Register							
Default = 0000h							R/W
EPBASEADDR bits 15-8							
15	14	13	12	11	10	9	8
EPBASEADDR bits 7-0							
7	6	5	4	3	2	1	0

bits 15:0 EPBASEADDR bits [15:0]
These bits form the lower 16 bits of the base address where the Event Processor code is located.

10.11.52 MDC Event Processor Base Address 1

REG[3066h] MDCEPBASEADDR1 Register							
Default = 0000h							R/W
EPBASEADDR bits 31-24							
15	14	13	12	11	10	9	8
EPBASEADDR bits 23-16							
7	6	5	4	3	2	1	0

bits 15:0 EPBASEADDR bits [31:16]
These bits form the upper 16 bits of the base address where the Event Processor code is located.

10.11.53 MDC VCOM Clock Control Register

REG[3068h] MDCVCOMCLKCTL Register							
Default = 0000h							R/W
SCRATCHB1 bits 15-10						VCOMCNTRST	VCOMCLKDIS
15	14	13	12	11	10	9	8
SCRATCHB0 bits 7-0							
7	6	5	4	3	2	1	0

bits 15:10 SCRATCHB1 bits [15:10]
These bits form the 6-bit scratchpad B1 register.

bit 9 VCOMCNTRST
This bit resets the internal 16-bit counter which generates the VCOM output clock to 0x0000.
When this bit = 0, internal 16-bit counter for VCOM is not reset.
When this bit = 1, internal 16-bit counter for VCOM is set to 0x0000.
NOTE: The internal 16-bit counter for VCOM is held in 0x0000 state while this bit is 1. This bit should be 0 for the counter to run.

bit 8 VCOMCLKDIS
This bit gates the clock for the VCOM clock generation circuit. The clock source for the VCOM clock generation circuit is OSC1.
When this bit = 0, the clock for the VCOM clock generation circuit is enabled.
When this bit = 1, the clock for the VCOM clock generation circuit is disabled (gated off).

bits 7:0 SCRATCHB0 bits [7:0]
These bits form the 8-bit scratchpad B0 register.

10.11.54 MDC Event Processor Control

REG[306Ah] MDCEPCTRL Register							
Default = 0000h							R/W
n/a							
15	14	13	12	11	10	9	8
n/a					TRIGSEL bits 1-0		EPEN
7	6	5	4	3	2	1	0

bits 2:1 TRIGSEL bits [1:0]
These bits select the trigger signal source to wake up the Event Processor to start executing.

0x0: RTC interrupt
0x1: MDC interrupt

0x2: any interrupt (logical OR of all interrupts)
0x3: GPIO port interrupt

bits 0 EPEN
Event Processor enable.
When this bit = 1, the event processor is enabled
When this bit = 0, the event processor is disabled

10.11.55 Product Code

REG[306Ch] PRODCODE Register							
Default = 005Fh							RO
PRODCODE[15:8]							
15	14	13	12	11	10	9	8
PRODCODE[7:0]							
7	6	5	4	3	2	1	0

bits 15-0 PRODCODE[15:0] (read-only)
These read-only bits provide the Product Code. Its value is 0x005F.

10.11.56 Revision Code

REG[306Eh] REVCODE Register							
Default = 0000h							RO
n/a							
15	14	13	12	11	10	9	8
PRODCODE[7:0]							
7	6	5	4	3	2	1	0

bits 7-0 REVCODE[7:0] (read-only)
These read-only bits provide the Revision Code. Its value is **0x0000**.

10.11.57 MDC Voltage Booster/Regulator Clock Control

REG[3080h] MDCBSTCLK Register							
Default = 0100h							R/W
n/a							Scratchpad Bit 1
15	14	13	12	11	10	9	8
Scratchpad Bit 0	CLKDIV bits 2-0			n/a			CLKSRC
7	6	5	4	3	2	1	0

bit 8 Scratchpad Bit 1
This bit will read back whatever is written to it and may be used as a scratchpad bit.
This bit always reads 1 on reset.

bit 7 Scratchpad Bit 0
This bit will read back whatever is written to it and may be used as a scratchpad bit.
This bit always reads 0 on reset.

bits 6:4 CLKDIV bits [2:0]
These bits select the division ratio of the MDC voltage booster operating clock source.

IOSC (CLKSRC=0)	**OSC1 (CLKSRC=1)**
0x0: 1/16	1/1
0x1: 1/32	1/2

0x2:	1/64	1/4
0x3:	1/128	1/8
0x4:	1/256	1/16
0x5:	1/512	1/32
0x6:	Reserved	1/64
0x7:	Reserved	1/128

bit 0 CLKSRC
This bit select the clock source of MDC voltage booster.
When this bit = 1, OSC1 is the clock source
When this bit = 0, IOSC is the clock source

10.11.58 MDC Power Output Control

REG[3084h] MDCBSTPWR Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a							
7	6	5	4	3	2	1	0
n/a				VMDBUP	BSTON	REGECO	REGON

bit 3 VMDBUP
This bit select the VMD output response speed.
When this bit = 1, the VMD output response speed is fast
When this bit = 0, the VMD output response speed is normal

bit 2 BSTON
This bit turns the voltage booster on and off.
When this bit = 1, the voltage booster is turned on
When this bit = 0, the voltage booster is turned off (Hi-Z)

bit 1 REGECO
This bit puts the voltage regulator into economy mode.
When this bit = 1, the voltage regulator is in economy mode
When this bit = 0, the voltage regulator is in normal mode

bit 0 REGON
This bit turns the voltage regulator on and off.
When this bit = 1, voltage regulator is on
When this bit = 0, voltage regulator is off

10.11.59 MDC Voltage Booster/Regulator VMD Output Control

REG[3088h] MDCBSTVMD Register
Default = 0000h R/W

15	14	13	12	11	10	9	8
n/a	VMDHVOL bits 2-0			n/a			VMDHON
7	6	5	4	3	2	1	0
n/a	VMDLVOL bits 2-0			n/a			VMDLON

bits 14:12 VMDHVOL bits [2:0]
These bits set the VMDH output voltage level.
0x0: 4.30V
0x1: 4.40V
0x2: 4.50V
0x3: 4.60V
0x4: 4.70V

0x5: 4.80V
0x6: 4.90V
0x7: 5.00V

bit 8 VMDHON
This bit controls the VMDH output.
When this bit = 1, VMDH output is turned on
When this bit = 0, VMDH output is turned off

bits 6:4 VMDLVOL bits [2:0]
These bits set the VMDL output voltage level.
0x0: 2.30V
0x1: 2.70V
0x2: 3.00V
0x3: 3.10V
0x4: 3.20V
0x5: 3.30V
0x6: 3.40V
0x7: 3.60V

bit 0 VMDLON
This bit controls the VMDL output.
When this bit = 1, VMDL output is turned on
When this bit = 0, VMDL output is turned off

11. Host Interface

The type of interface can be selected from Indirect 8-bit parallel and SPI/QSPI (single, dual, and quad SPI) by setting the HIFCNF pin. See Section 4.3 for details on HIFCNF.

The command interface between the host MCU and the S1D13C00 is similar to the command interface between an MCU and a serial flash chip. The same read and program (write) command codes are used.

11.1 Indirect 8-bit Parallel Interface

The Indirect 8-bit parallel interface is selected by pulling the HIFCNF pin down to low. The external host MCU uses the 8-bit data bus of this interface to send commands and addresses, and to read/write data to the internal memory bus of the S1D13C00.

Write access from external host MCU

Figure 11.1 shows an example of write access by the external host MCU.

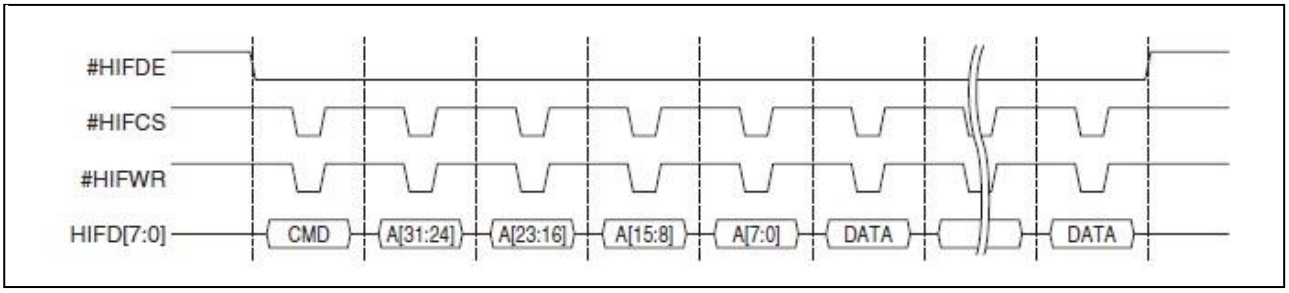


Figure 11.1 Indirect 8-bit Write Access Example

The external host MCU asserts the #HIFDE (device enable) signal to start accessing this MCU. The write access sequence consists of a 1-byte command write cycle, a 4-byte address write cycle (the write start address bytes are sent sequentially from the most significant byte (A[31:24]) first), and a data write cycle for one or more data bytes. The write address is automatically incremented every time a data byte is written. When the data write cycle has completed, the external host MCU negates the #HIFDE signal to terminate the write access sequence. The write access sequence can also be terminated by setting the #HIFDE signal to high even if the data write cycle has not completed.
In a write cycle, both the #HIFCS and #HIFWR signals are set to low.

Write commands

In an Indirect 8-bit parallel interface, the external host MCU can send any one of the commands (CMD byte) shown below at the beginning of a write access.

- PAGEPROG = 0x02
- EXTDUALINFASTPROG = 0xD2
- EXTQUADINFASTPROG = 0x12
- DUALINFASTPROG = 0xA2
- QUADINFASTPROG = 0x32

The S1D13C00 host interface accepts any one of these commands as a “write” command to write data to the internal memory.

Read access from external host MCU

Figure 11.2 shows an example of read access by the external host MCU.

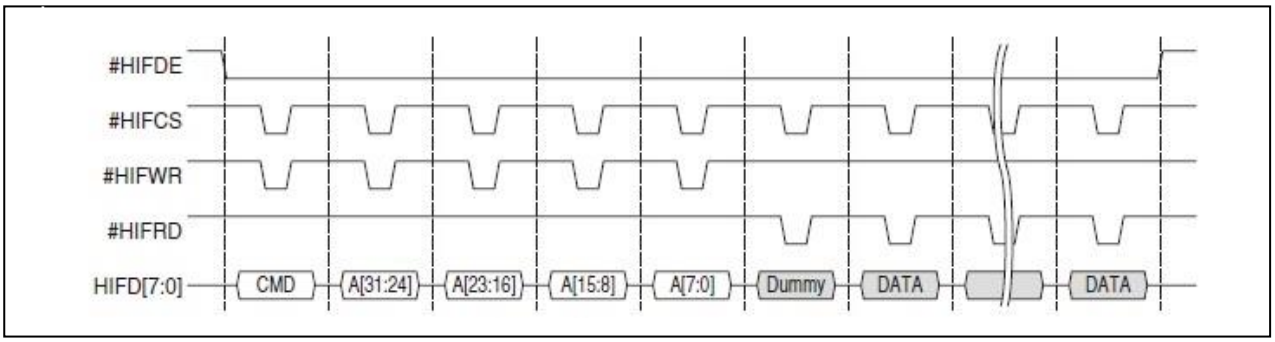


Figure 11.2 Indirect 8-bit Read Access Example

The external host MCU asserts the #HIFDE (device enable) signal to start accessing the S1D13C00. The read access sequence consists of a 1-byte command write cycle, a 4-byte address write cycle (the read start address bytes are sent sequentially from the most significant byte (A[31:24]) first), a 1-byte dummy read cycle, and a data read cycle for one or more data bytes. The read address is automatically incremented every time a data byte is read. When the data read cycle has completed, the external host MCU negates the #HIFDE signal to terminate the read access sequence. The read access sequence can also be terminated by setting the #HIFDE signal to high even if the data read cycle has not completed.
In a read cycle, both the #HIFCS and #HIFRD signals are set to low.

Read commands

In an Indirect 8-bit parallel interface, the external host MCU can send any one of the commands (CMD byte) shown below at the beginning of a read access.

- READ = 0x03
- FASTREAD = 0x0B
- DUALOUTFASTREAD = 0x3B
- DUALIOFASTREAD = 0xBB
- QUADOUTFASTREAD = 0x6B
- QUADIOFASTREAD = 0xEB

The S1D13C00 host interface accepts any one of these commands as a “read” command to read data from the internal memory.

11.2 SPI/QSPI Serial Interface

The SPI/QSPI serial interface is selected by pulling the HIFCNF pin up to high. Furthermore, setting the HSPISEL[1] (HIFD[5]) and HSPISEL[0] (HIFD[4]) pins select the protocol as Single/Extended SPI (HSPISEL[1:0] = 0b1*), Dual SPI (HSPISEL[1:0] = 0b10), or Quad SPI (HSPISEL[1:0] = 0b11).

In SPI/QSPI mode, the #HIFDE pin is the active-low #HSPISS (chip-select) input and the #HIFWR pin is the HSPICLK input serial clock. There are 5 types of protocols for SPI/QSPI mode:

- Single protocol: HIFD[0] is HSPIDI input and HIFD[1] is HSPIDO output
- Extended Dual protocol: Command byte is Single protocol, Address bytes are Single or Dual depending on the command, Data bytes are Dual protocol with HIFD[1:0] acting as bidirectional data
- Extended Quad protocol: Command byte is Single protocol, Address bytes are Single or Quad depending on the command, Data bytes are Quad protocol with HIFD[3:0] acting as bidirectional data
- Dual protocol: HIFD[1:0] are bidirectional for Command, Address, and Data bytes
- Quad protocol: HIFD[3:0] are bidirectional for Command, Address, and Data bytes

The following table summarizes the types of SPI/QSPI interfaces supported, associated command bytes, and data lines used:

Table 11.1 SPI/QSPI Host Interfaces

Command	HIFSEL [1:0]	SPI/QSPI Protocol	Data Lines Used and Number of Clock Cycles					
			CMD[7:0]	ADDR[31:24]	ADDR[23:16]	ADDR[15:8]	ADDR[7:0]	DATA[7:0]
Read (0x03) or FASTREAD (0x0B)	0b0*	Single	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[1] (8 clocks)
PAGEPROG (0x02)		Single	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)
DUALOUTFASTREAD (0x3B) DUALINFASTPROG (0xA2)		Extended Dual	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[1:0] (4 clocks)
DUALIOFASTREAD (0xBB) EXTDUALINFASTPROG (0xD2)			HIFD[0] (8 clocks)	HIFD[1:0] (4 clocks)	HIFD[1:0] (4 clocks)	HIFD[1:0] (4 clocks)	HIFD[1:0] (4 clocks)	HIFD[1:0] (4 clocks)
QUADOUTFASTREAD (0x6B) QUADINFASTPROG (0x32)		Extended Quad	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[0] (8 clocks)	HIFD[3:0] (2 clocks)
QUADIOFASTREAD (0xEB) EXTQUADINFASTPROG (0x12)			HIFD[0] (8 clocks)	HIFD[3:0] (2 clocks)	HIFD[3:0] (2 clocks)	HIFD[3:0] (2 clocks)	HIFD[3:0] (2 clocks)	HIFD[3:0] (2 clocks)
All read commands All write commands	0b10	Dual	HIFD[1:0] (4 clocks)	HIFD[1:0] (4 clocks)	HIFD[1:0] (4 clocks)	HIFD[1:0] (4 clocks)	HIFD[1:0] (4 clocks)	HIFD[1:0] (4 clocks)
All read commands All write commands	0b11	Quad	HIFD[3:0] (2 clocks)	HIFD[3:0] (2 clocks)	HIFD[3:0] (2 clocks)	HIFD[3:0] (2 clocks)	HIFD[3:0] (2 clocks)	HIFD[3:0] (2 clocks)

Write access from external host MCU

Figure 11.3 shows an example of write access by the external host MCU.

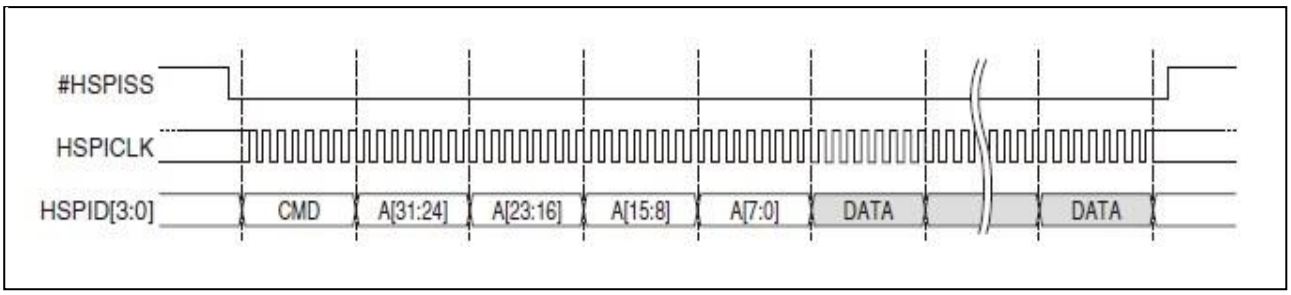


Figure 11.3 SPI/QSPI Write Access Example

The external host MCU asserts the #HSPISS (slave select) signal to start accessing the S1D13C00. The write access sequence consists of a 1-byte command write, a 4-byte address write (the write start address bytes are sent sequentially from the most significant byte (A[31:24]) first), and a burst write for one or more data bytes. The write address is automatically incremented every time a data byte is written. When the data write has completed, the external host MCU negates the #HSPISS signal to terminate the write access sequence. The

write access sequence can also be terminated by setting the #HSPISS signal to high even if the data write cycle has not completed. For each byte, the MSB (most significant bit) is sent first.

Read access from external host MCU

Figure 11.4 shows an example of read access by the external host MCU.

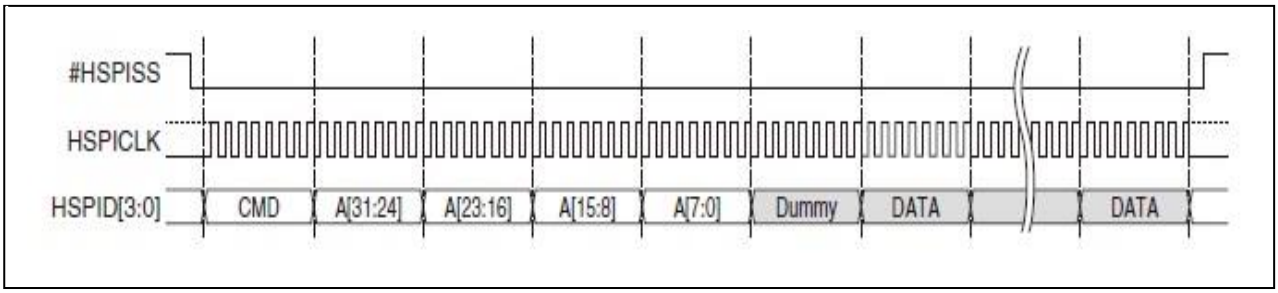


Figure 11.4 SPI/QSPI Read Access Example

The external host MCU asserts the #HSPISS (slave select) signal to start accessing the S1D13C00. The read access sequence consists of a 1-byte command write, a 4-byte address write (the read start address bytes are sent sequentially from the most significant byte (A[31:24]) first), a 1-byte dummy read, and a burst read for one or more data bytes. The read address is automatically incremented every time a data byte is read. When the data read has completed, the external host MCU negates the #HSPISS signal to terminate the read access sequence. The read access sequence can also be terminated by setting the #HSPISS signal to high even if the data read has not completed.

SPI/QSPI protocols

a. Single/Extended SPI protocol (HSPISEL1 = 0, HSPISEL0 = don't care)

READ/FASTREAD/PAGEPROG commands
- Command byte: single-bit I/O (8 clocks per byte)
- Address bytes: single-bit I/O (8 clocks per byte)
- Data bytes: single-bit I/O (8 clocks per byte)

DUALOUTFASTREAD/DUALINFASTPROG commands
- Command byte: single-bit I/O (8 clocks per byte)
- Address bytes: single-bit I/O (8 clocks per byte)
- Data bytes: dual-bit I/O (4 clocks per byte)

DUALIOFASTREAD/EXTDUALINFASTPROG commands
- Command byte: single-bit I/O (8 clocks per byte)
- Address bytes: dual-bit I/O (4 clocks per byte)
- Data bytes: dual-bit I/O (4 clocks per byte)

QUADOUTFASTREAD/QUADINFASTPROG commands:
- Command byte: single-bit I/O (8 clocks per byte)
- Address bytes: single-bit I/O (8 clocks per byte)
- Data bytes: quad-bit I/O (2 clocks per byte)

QUADIOFASTREAD/EXTQUADINFASTPROG commands:
- Command byte: single-bit I/O (8 clocks per byte)
- Address bytes: quad-bit I/O (2 clocks per byte)
- Data bytes: quad-bit I/O (2 clocks per byte)

b. Dual SPI protocol (HSPISEL1 = 1, HSPISEL0 = 0)

All commands
- Command byte: dual-bit I/O (4 clocks per byte)
- Address bytes: dual-bit I/O (4 clocks per byte)
- Data bytes: dual-bit I/O (4 clocks per byte)

c. Quad SPI protocol (HSPISEL1 = 1, HSPISEL0 = 1)

All commands

Command byte: quad-bit I/O (2 clocks per byte)
Address bytes: quad-bit I/O (2 clocks per byte)
Data bytes: quad-bit I/O (2 clocks per byte)

Data format

The host interface supports the following SPI/QSPI data format:

Data length: 8 bits
Input/output permutation: MSB first
Clock phase: Mode 0/Mode 3 (data is latched at the rising edge and is shifted at the falling edge)
Clock polarity: Mode 0 (low level at idle)
Mode 3 (high level at idle)

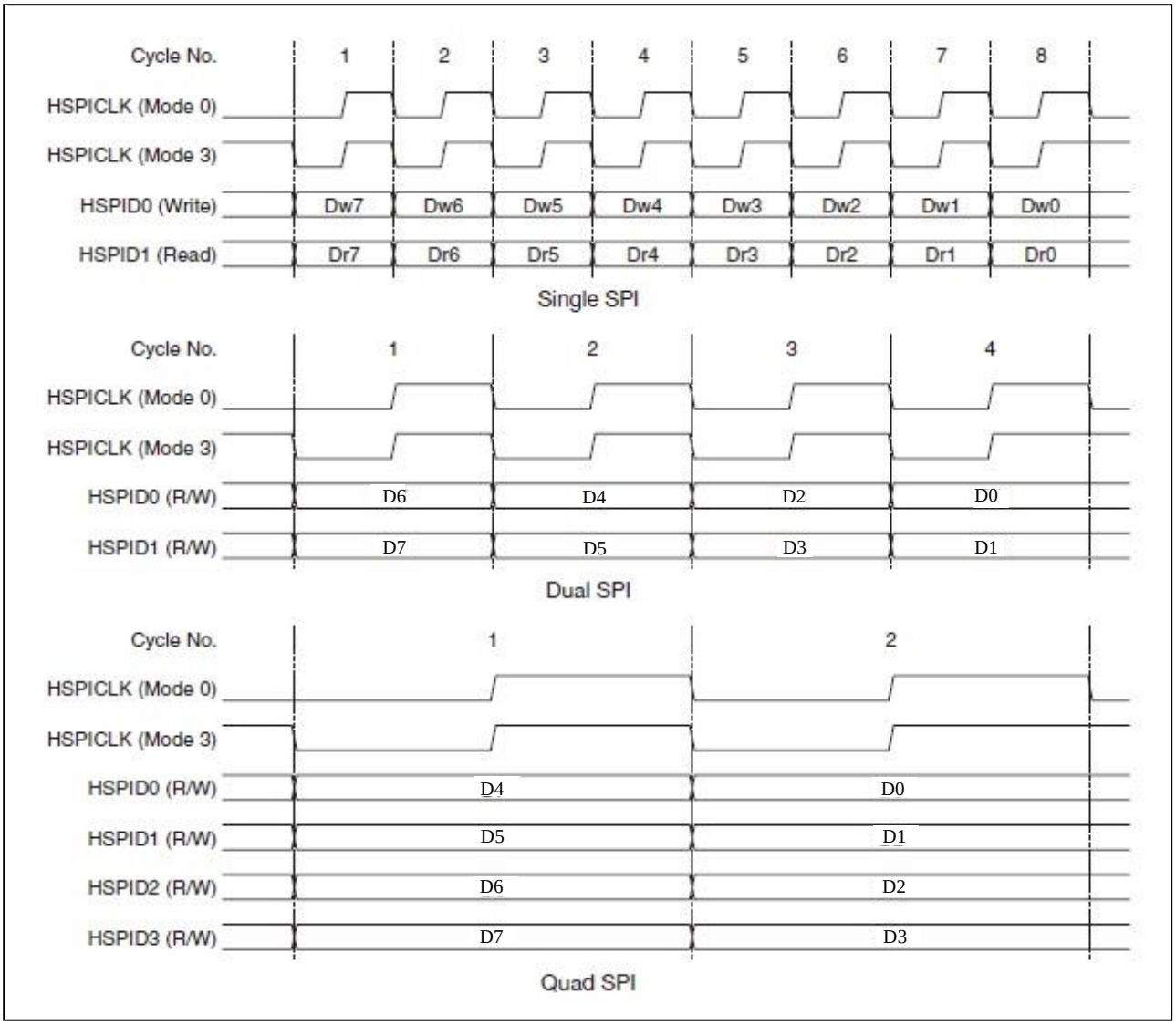


Figure 11.5 SPI/QSPI Data Format

Read/write Commands

The SPI/QSPI serial interface uses the same commands as the Indirect 8-bit parallel interface.

Write commands

- PAGEPROG = 0x02
- EXTDUALINFASTPROG = 0xD2 *
- EXTQUADINFASTPROG = 0x12 *
- DUALINFASTPROG = 0xA2 *
- QUADINFASTPROG = 0x32 *

Read commands

- READ = 0x03
- FASTREAD = 0x0B
- DUALOUTFASTREAD = 0x3B *
- DUALIOFASTREAD = 0xBB *
- QUADOUTFASTREAD = 0x6B *
- QUADIOFASTREAD = 0xEB *

*** Extended SPI mode**

If a command indicated with * in the list above is received in single SPI mode, the host interface enters extended SPI mode. The address and data bytes used HIFD[1:0] or HIFD[3:0] depending on the command. Please refer Table 11.1.

11.3 Host Access Timings

The SYSCTRL and SYSINTS registers are Asynchronous. They can be accessed (read and write) from the Host MCU with fixed signal timings described in Section 9.2, and the IOSC oscillator in the S1D13C00 does not need to be turned on.

The rest of the registers, internal RAM, and memory-mapped serial flash area are Synchronous. They require the IOSC oscillator to be turned on. Because the Host Interface signals are asynchronous relative to the internal system clock which runs the Synchronous memory/registers, the timing requirements for the Host Interface signals are variable depending on the activity of the S1D13C00's internal busses. The following diagram shows the internal busses of the S1D13C00:

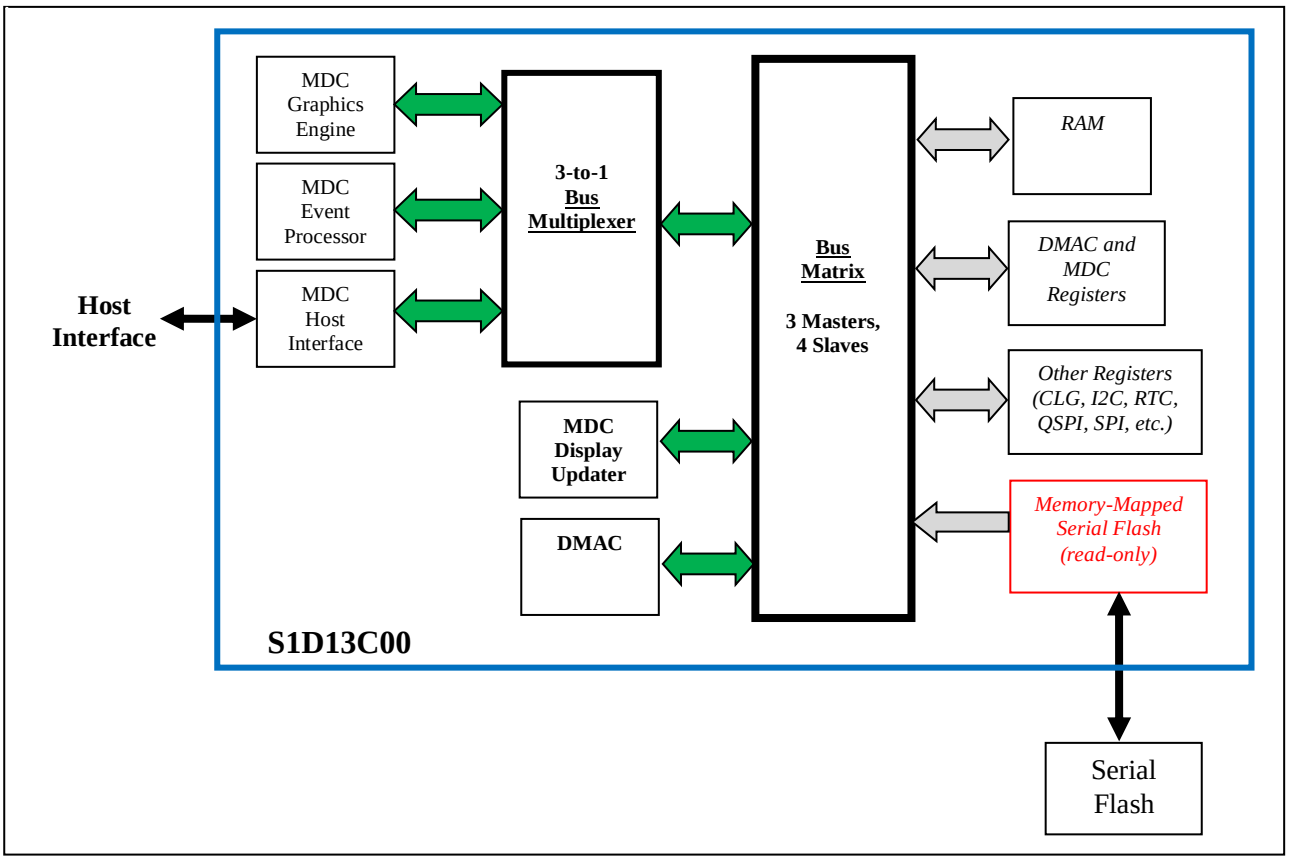


Figure 11.6 S1D13C00 Internal Busses

11.3.1 Internal Bus Matrix (3 Masters, 4 Slaves)

The Bus Matrix connects 3 Masters (3-to-1 Bus Multiplexer, MDC Display Updater, DMAC) to 4 Slaves (RAM, DMAC and MDC Registers, Other Registers, and Memory-Mapped Serial Flash). It allows concurrent active master-slave accesses as long as two masters are not accessing the same slave. If two or more masters are simultaneously trying to access the same slave, the arbitration hardware will grant access to one master and hold off the other masters to wait their turn in a round-robin fashion. Or, if a master is in the middle of an access to a slave and another master wants to access the same slave, it will be held off and needs to wait until the current master is finished.

11.3.2 Internal 3-to-1 Bus Multiplexer (3 Masters)

The 3-to-1 Bus Multiplexer, which is one of the masters of the Bus Matrix, is shared between 3 Masters: MDC Graphics Engine, MDC Event Processor, and MDC Host Interface. The arbitration policy is round-robin. If one master is currently accessing a location in the memory map and another master wants to access any location in the memory map, it will be held off until the current master is finished.

11.3.3 Bus Matrix Slave Access Times

Read or write access to RAM and registers takes 2 system clock cycles.

Read access from Memory-Mapped Serial Flash can take many system clock cycles depending on factors such as QSPI clock speed and transfer size (8-, 16-, or 32-bit). Host reads from Memory-Mapped Serial Flash is not recommended because the Host Interface timing needs to be slowed down by a large amount and the access time is variable by the factors just mentioned. For Host reading of Memory-Mapped Serial Flash data, it is recommend to use the DMA controller (DMAC) to transfer blocks of data to RAM and then reading the data from RAM over the Host Interface.

11.3.4 Bus Activity and Host Access

When the Host MCU accesses RAM or register and there are no other bus masters accessing the same RAM or register, it only takes 2 system clock cycles. However, if one or more other bus master is active and accessing the same Bus Matrix slave (RAM or register), the worst case delay before the Host Interface is granted access to the bus is [2 x nACTIVE] system clock cycles, where nACTIVE is the number of other bus masters actively accessing the same Bus Matrix slave.

Host Access Restriction
If the DMAC and/or MDC Display Updater is actively reading data from Memory-Mapped Serial Flash, the Host MCU can still access RAM or register without any problems because the Bus Matrix has a separate arbitration hardware for each slave, and the accesses do not clash.

However, if the MDC Graphics Engine and/or MDC Event Processor is actively reading from the Memory-Mapped Serial Flash and the Host MCU tries to access RAM or register area, the access will be unreliable because the 3-to-1 Bus Multiplexer allows only one master to be active at a time. If the MDC Graphics Engine or MDC Event Processor is in the middle of a long multi-cycle read from Memory-Mapped Serial Flash, the bus request from the Host will be missed. Therefore, it is recommended that the Host MCU should not access the S1D13C00 while either one of MDC Graphics Engine or MDC Event Processor is actively reading from Memory-Mapped Serial Flash.

12. Real-Time Clock (RTC)

12.1 Overview

The RTC is a real-time clock with a perpetual calendar function. The main features of RTC are outlined below.

- Includes a BCD real-time clock counter to implement a time-of-day clock (second, minute, and hour) and calendar (day, day of the week, month, and year with leap year supported).
- Provides a hold function for reading correct counter values by suspending the real-time clock counter operation.
- 24-hour or 12-hour mode is selectable.
- Capable of controlling the starting and stopping of the time-of-day clock.
- Provides a 30-second correction function to adjust time using a time signal.
- Includes a 1 Hz counter to count 128 to 1 Hz.
- Includes a BCD stopwatch counter with 1/100-second counting supported.
- Provides a theoretical regulation function to correct clock error due to frequency tolerance with no external parts required.

Figure 12.1 shows a block diagram of the RTC.

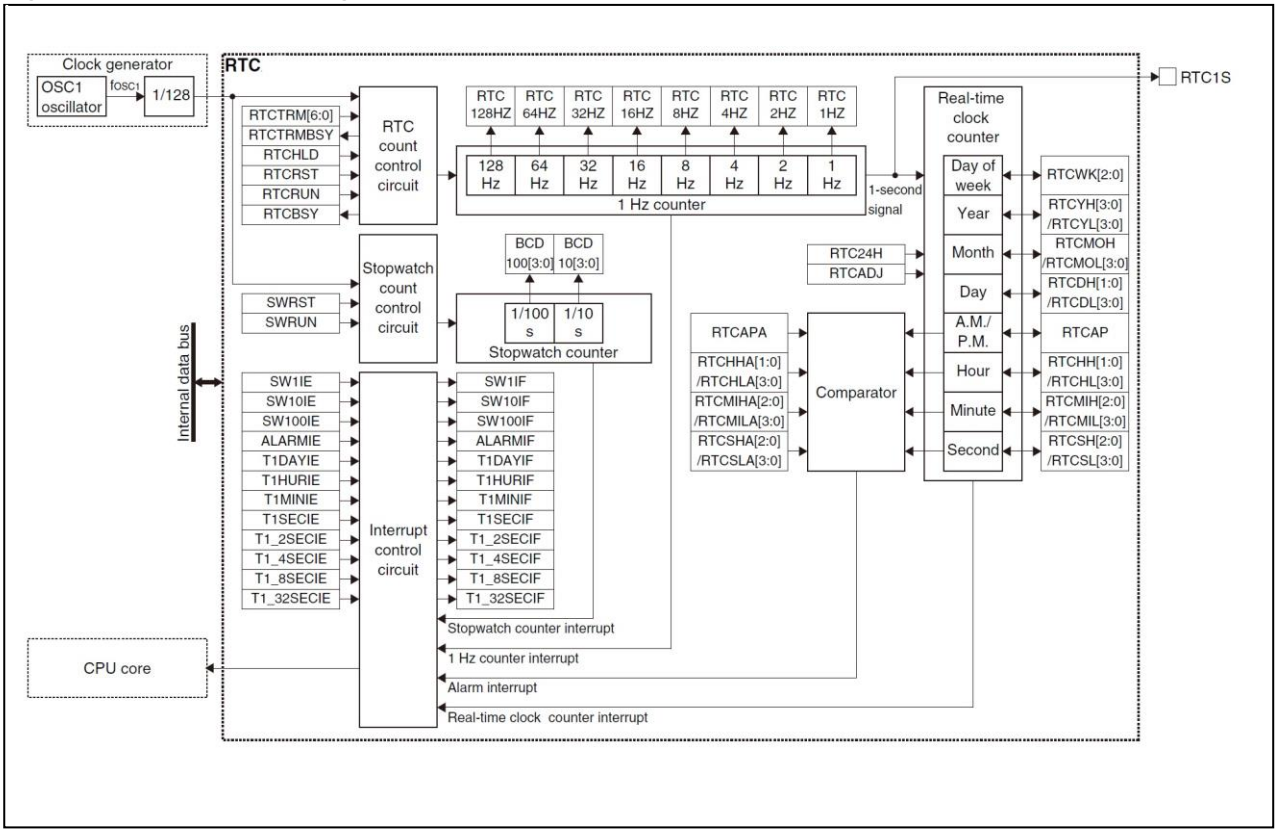


Figure 12.1 RTC Block Diagram

12.2 Output Pin

An output signal called RTC1S on the shared GPIO port P10 provides the option to monitor the internal 1-second pulse of the RTC for measuring the accuracy of the OSC1 oscillator.

12.3 RTC Operating Clock

RTC uses CLK_RTC, which is generated by the clock generator from OSC1 as the clock source, as its operating clock. RTC is operable when OSC1 is enabled.

12.3.1 Theoretical Regulation Function

The RTC is clocked by a 256 Hz internal clock. This internal 256Hz clock is generated by the OSC1 clock (32.768kHz nominal) divided down by 128 (32768/128 = 256), and it is used to clock an 8-bit counter which counts from 0 to 255. When this counter reaches its maximum value of 255 it rolls over to 0 and generates the RTC1S pulse. If OSC1 has no frequency error, the RTC1S pulse will be generated at a frequency of exactly 1 Hz. However, because of manufacturer variations of the oscillation circuit components and temperature drift, the frequency of the OSC1 oscillator will deviate from its nominal frequency of 32.768 kHz and the RTC1S pulse does not occur at exactly 1-second intervals.

In order to compensate for frequency error, the RTC has a Theoretical Regulation hardware which, when triggered, modifies the 8-bit counter's value in order to maintain the 1 Hz signal of the RTC1S. For example, if the OSC1 oscillator was running at a frequency lower than the nominal frequency of 32.768 kHz, the 8-bit counter would reach the value of 255 at a slower frequency than 1 Hz. To correct this, the Theoretical Regulation hardware, when triggered, will increase the value of the 8-bit counter by a software-programmable value (RTCCTL.RTCTRM[6:0] register) in order to reach the value of 255 faster. The Theoretical Regulation is triggered by writing the desired correction value to the RTCCTL.RTCTRM[6:0] register. The RTCCTL.RTCTRMBSY bit indicates that the Theoretical Regulation is running and the RTCINTF.RTCTRMIF interrupt flag is set to 1 when Theoretical Regulation is finished.

The RTCCTL.RTCTRM[6:0] register value is the 2's complement representation of the Theoretical Regulation correction value, **CV**, and its range is -64 to +63. The Theoretical Regulation circuit can correct the accumulated time error by (**CV** / 256) seconds. Each time Theoretical Regulation is triggered, the 8-bit counter which generates the 256Hz internal clock is "bumped" by the value of CV, and this can be used to compensate for the frequency error of OSC1. The Theoretical Regulation hardware of the RTC can be used to implement a software-based compensation for the OSC1 crystal's frequency error drift over temperature.

12.4 Operations

12.4.1 RTC Control

Follow the sequences shown below to set time to RTC, to read the current time and to set alarm.

Time setting

1. Set RTC to 12H or 24H mode using the RTCCTLL.RTC24H bit.
2. Write 1 to the RTCCTLL.RTCRUN bit to enable for the real-time clock counter to start counting up.
3. Check to see if the RTCCTLL.RTCBSY bit = 0 that indicates the counter is ready to rewrite. If the RTCCTLL.RTCBSY bit = 1, wait until it is set to 0.
4. Write the current date and time in BCD code to the control bits listed below.
   RTCSEC.RTCSH[2:0]/RTCSL[3:0] bits (second)
   RTCHUR.RTCMIH[2:0]/RTCMIL[3:0] bits (minute)
   RTCHUR.RTCHH[1:0]/RTCHL[3:0] bits (hour)
   RTCHUR.RTCAP bit (AM/PM) (effective when RTCCTLL.RTC24H bit = 0)
   RTCMON.RTCDH[1:0]/RTCDL[3:0] bits (day)
   RTCMON.RTCMOH/RTCMOL[3:0] bits (month)
   RTCYAR.RTCYH[3:0]/RTCYL[3:0] bits (year)
   RTCYAR.RTCWK[2:0] bits (day of the week)
5. Write 1 to the RTCCTLL.RTCADJ bit (execute 30-second correction) using a time signal to adjust the time. (For more information on the 30-second correction, refer to "Real-Time Clock Counter Operations.")
6. Write 1 to the real-time clock counter interrupt flags in the RTCINTF register to clear them.

7. Write 1 to the interrupt enable bits in the RTCINTE register to enable real-time clock counter interrupts.

Time read

1. Check to see if the RTCCTLL.RTCBSY bit = 0. If the RTCCTLL.RTCBSY bit = 1, wait until it is set to 0.
2. Write 1 to the RTCCTLL.RTCHLD bit to suspend count-up operation of the real-time clock counter.
3. Read the date and time from the control bits listed in "Time setting, Step 4" above.
4. Write 0 to the RTCCTLL.RTCHLD bit to resume count-up operation of the real-time clock counter. If a second count-up timing has occurred in the count hold state, the hardware corrects the second counter for +1 second (for more information on the +1 second correction, refer to "Real-Time Clock Counter Operations").

Alarm setting

1. Write 0 to the RTCINTE.ALARMIE bit to 0 to disable alarm interrupts.
2. Write the alarm time in BCD code to the control bits listed below (a time within 24 hours from the current time can be specified).
   RTCALM1.RTCSHA[2:0]/RTCSLA[3:0] bits (second)
   RTCALM2.RTCMIHA[2:0]/RTCMILA[3:0] bits (minute)
   RTCALM2.RTCHHA[1:0]/RTCHLA[3:0] bits (hour)
   RTCALM2.RTCAPA bit (AM/PM) (effective when RTCCTLL.RTC24H bit = 0)
3. Write 1 to the RTCINTF.ALARMIF bit to clear the alarm interrupt flag.
4. Write 1 to the RTCINTE.ALARMIE bit to enable alarm interrupts. When the real-time clock counter reaches the alarm time set in Step 2, an alarm interrupt occurs.

12.4.2 Real-Time Clock Counter Operations

The real-time clock counter consists of second, minute, hour, AM/PM, day, month, year, and day of the week counters and it performs counting up using the RTC1S signal. It has the following functions as well.

Recognizing leap years

The leap year recognizing algorithm used in RTC is effective only for Christian Era years. Years within 0 to 99 that can be divided by four without a remainder are recognized as leap years. If the year counter = 0x00, RTC assumes it as a common year. If a leap year is recognized, the count range of the day counter changes when the month counter is set to February.

Corrective operation when a value out of the effective range is set

When a value out of the effective range is set to the year, day of the week, or hour (in 24H mode) counter, the counter will be cleared to 0 at the next count-up timing. When a such value is set to the month, day, or hour (in 12H mode) counter, the counter will be set to 1 at the next count-up timing.

30-second correction

This function is provided to set the time-of-day clock by the time signal. Writing 1 to the RTCCTLL.RTCADJ bit adds 1 to the minute counter if the second counter represents 30 to 59 seconds, or clears the second counter with the minute counter left unchanged if the second counter represents 0 to 29 seconds.

+1 second correction

If a second count-up timing occurred while the RTCCTLL.RTCHLD bit = 1 (count hold state), the real-time clock counter counts up by +1 second (performs +1 second correction) after the counting has resumed by writing 0 to the RTCCTLL.RTCHLD bit.

NOTE: If two or more second count-up timings occurred while the RTCCTLL.RTCHLD bit = 1, the counter is always corrected for +1 second only.

12.4.3 Stopwatch Control

Follow the sequences shown below to start counting of the stopwatch and to read the counter.

Count start

1. Write 1 to the RTCSWCTL.SWRST bit to reset the stopwatch counter.
2. Write 1 to the stopwatch interrupt flags in the RTCINTF register to clear them.
3. Write 1 to the interrupt enable bits in the RTCINTE register to enable stopwatch interrupts.
4. Write 1 to the RTCSWCTL.SWRUN bit to start stopwatch count up operation.

Counter read

1. Read the count value from the RTCSWCTL.BCD10[3:0] and BCD100[3:0] bits.
2. Read again.
   i. If the two read values are the same, assume that the count values are read correctly.
   ii. If different values are read, perform reading once more and compare the read value with the previous one.

12.4.4 Stopwatch Count-up Pattern

The stopwatch consists of 1/100-second and 1/10-second counters and these counters perform counting up in increments of approximate 1/100 and 1/10 seconds with the count-up patterns shown in Figure 12.2.

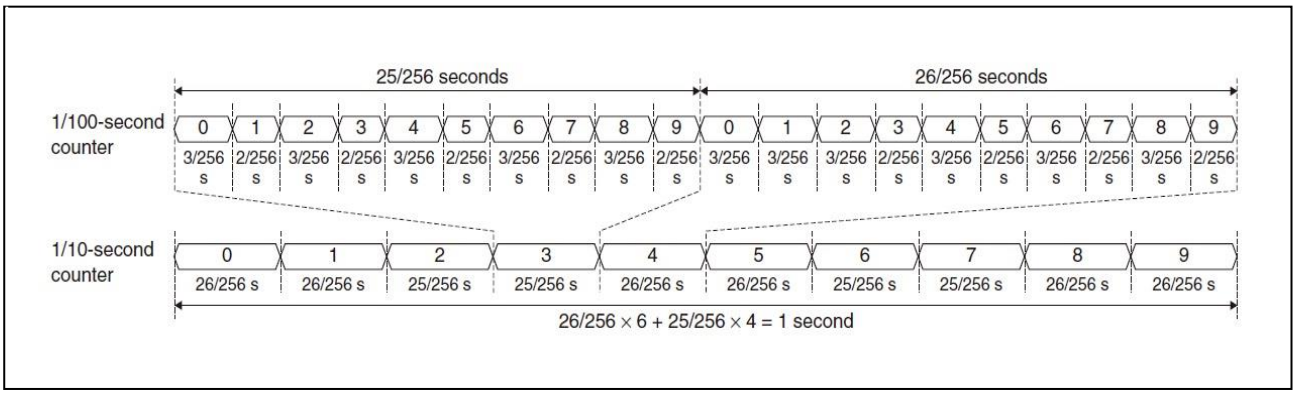


Figure 12.2 Stopwatch Count-Up patterns

12.5 Interrupts

RTC has a function to generate the interrupts shown in Table 12.1.

Table 12.1 RTC Interrupt Function

Interrupt	Interrupt flag	Set Condition	Clear condition
Alarm	RTCAINTF.ALARMIF	Matching between the RTCAALM1–2 register contents and the real-time clock counter contents	Writing 1
1-day	RTCAINTF.T1DAYIF	Day counter count up	Writing 1
1-hour	RTCAINTF.T1HURIF	Hour counter count up	Writing 1
1-minute	RTCAINTF.T1MINIF	Minute counter count up	Writing 1
1-second	RTCAINTF.T1SECIF	Second counter count up	Writing 1
1/2-second	RTCAINTF.T1_2SECIF	See Figure 12.3	Writing 1
1/4-second	RTCAINTF.T1_4SECIF	See Figure 12.3	Writing 1
1/8-second	RTCAINTF.T1_8SECIF	See Figure 12.3	Writing 1
1/32-second	RTCAINTF.T1_32SECIF	See Figure 12.3	Writing 1
Stopwatch 1 Hz	RTCAINTF.SW1IF	1/10-second counter overflow	Writing 1
Stopwatch 10 Hz	RTCAINTF.SW10IF	1/10-second counter count up	Writing 1
Stopwatch 100 Hz	RTCAINTF.SW100IF	1/100-second counter count up	Writing 1
Theoretical regulation completion	RTCAINTF.RTCTRMIF	At the end of theoretical regulation operation	Writing 1

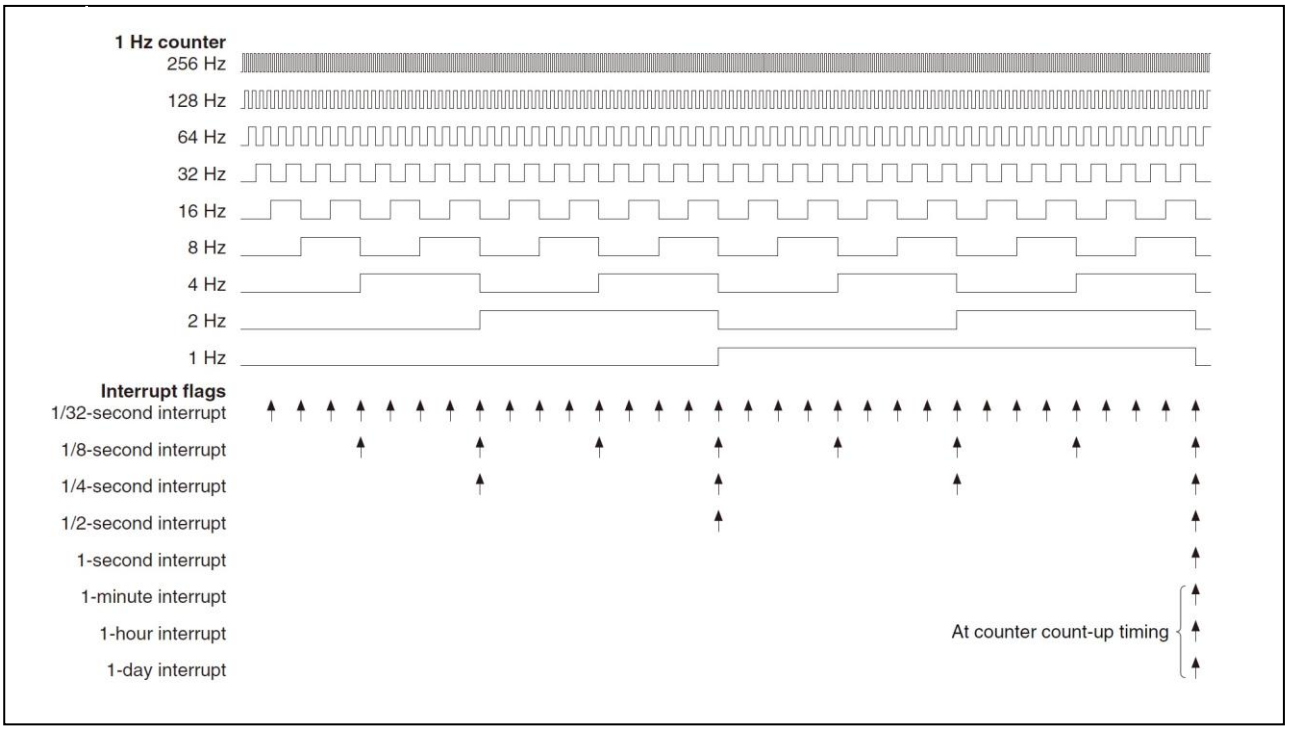


Figure 12.3 RTC Interrupt Timings

NOTES:
- 1-second to 1/32-second interrupts occur after a lapse of 1/256 second from change of the 1 Hz counter value.
- An alarm interrupt occurs after a lapse of 1/256 second from matching between the AM/PM (in 12H mode), hour, minute, and second counter value and the alarm setting value.

The RTCINTE register provides interrupt enable bits corresponding to each interrupt flag.
Figure 12.4 shows a diagram of the RTCINT interrupt signal. The RTCINT signal is logical ORed with interrupt signals from other peripherals to generate the HIFIRQ interrupt signal to the host MCU.
See Section 22 ("Interrupts") for more details on the HIFIRQ output.

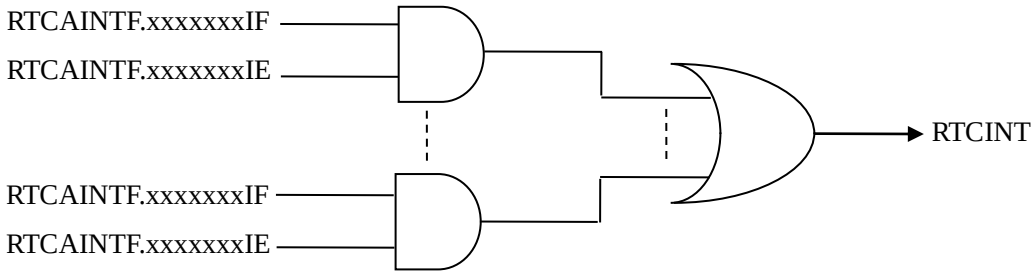


Figure 12.4 RTCINT Interrupt Circuit

12.6 Control Registers

See Section 10.3 for descriptions of the control registers for RTC.

13. GPIO Ports

13.1 Overview

GPIO controls the I/O ports. The main features are outlined below.

- Allows port-by-port function configurations:
  - Each port can be configured with or without a pull-up or pull-down resistor.
  - Each port can be configured with or without a chattering filter.
  - Allows selection of the function (general-purpose I/O port (GPIO) function, up to four peripheral I/O functions) to be assigned to each port.
- P0x pins are initially placed into Hi-Z state and P1x pins are initially configured as outputs driving low.

NOTE: 'x', which is used in the port names Pxy, register names, and bit names, refers to a port group (x = 0 or 1) and 'y' refers to a port number (y = 0, 1, 2, ⋯ , 7).

Figure 13.1 shows the configuration of the GPIO module.

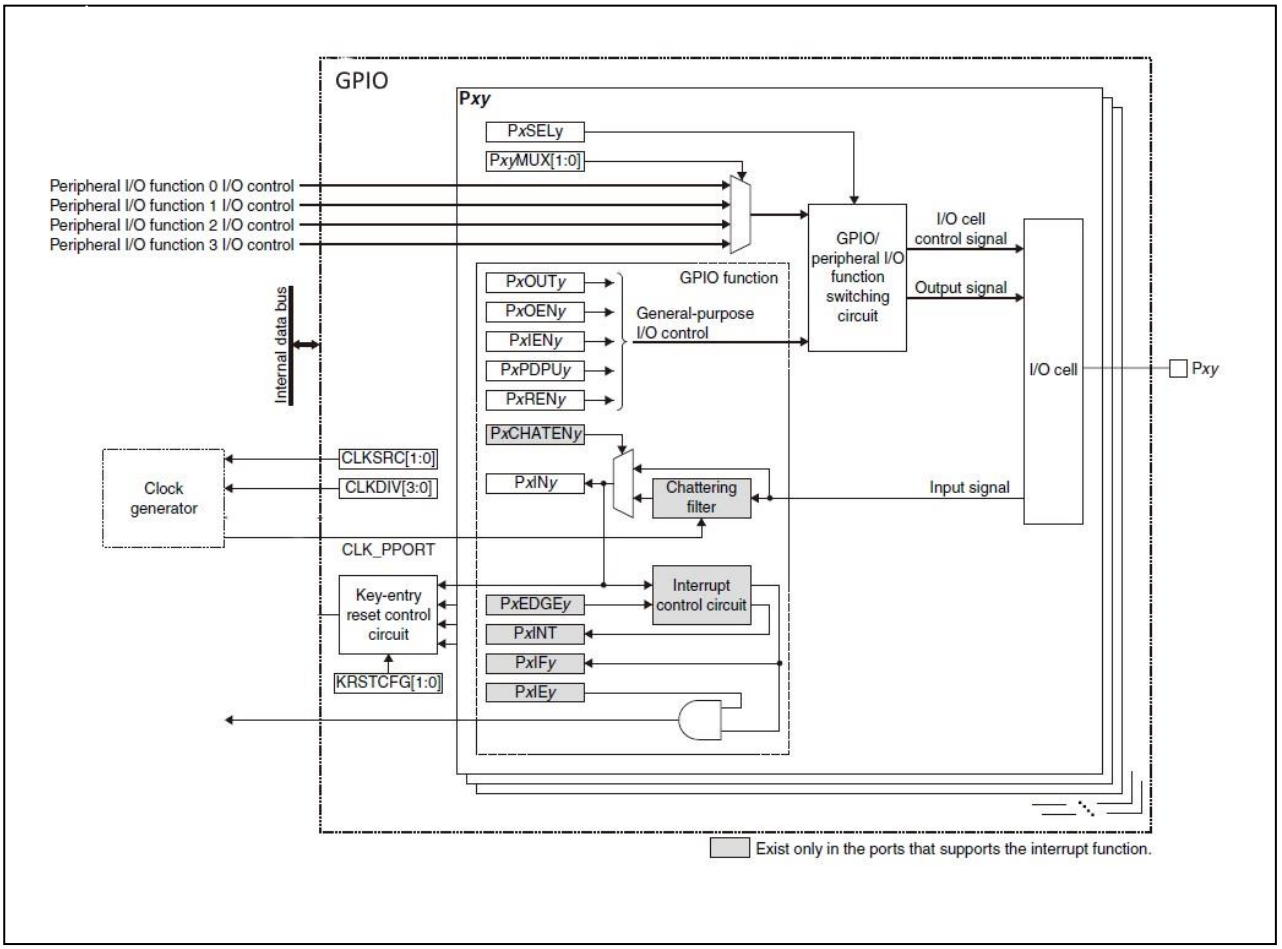


Figure 13.1 GPIO Configuration

13.2 I/O Cell Structure and Functions

Figure 13.2 shows the I/O cell configuration.

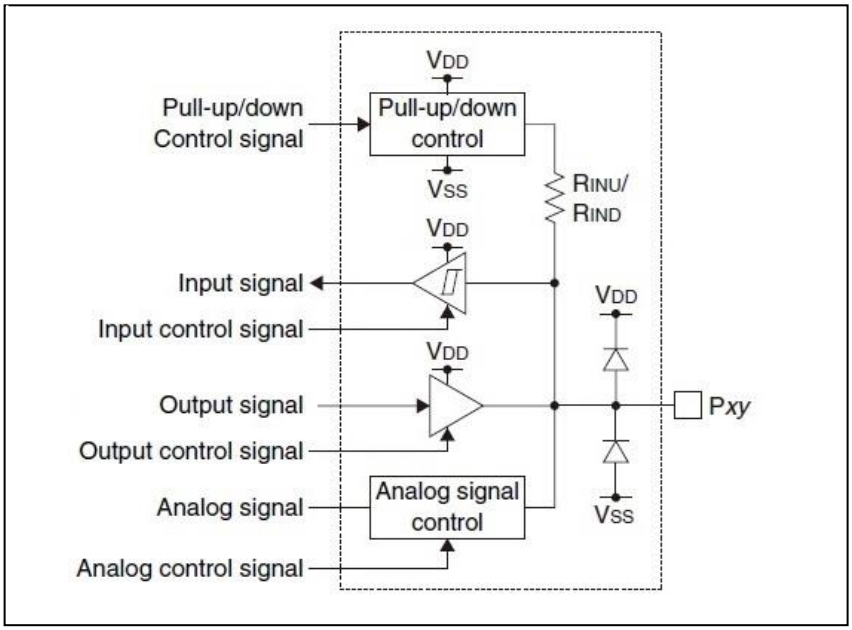


Figure 13.2 I/O Cell Configuration

13.2.1 Schmitt Input

The input functions are all configured with the Schmitt interface level. When a port is set to input disable status (PORTPxIOEN.PxIENy bit = 0), unnecessary current is not consumed if the Pxy pin is placed into floating status.

13.2.2 Pull-Up/Pull-Down

The GPIO port has a pull-up/pull-down function. Either pull-up or pull-down may be selected for each port individually. This function may also be disabled for the port that does not require pulling up/down. When the port level is switched from low to high through the pull-up resistor included in the I/O cell or from high to low through the pull-down resistor, a delay will occur in the waveform rising/falling edge depending on the time constant by the pull-up/pull-down resistance and the pin load capacitance. The rising/falling time is commonly determined by the following equation:

$$tPR = -RINU \times (CIN + CBOARD) \times \ln(1 - VT+/VDD)$$
$$tPF = -RIND \times (CIN + CBOARD) \times \ln(1 - VT-/VDD)$$

Where
- tPR: Rising time (port level = low → high) [second]
- tPF: Falling time (port level = high → low) [second]
- VT+: High level Schmitt input threshold voltage [V]
- VT-: Low level Schmitt input threshold voltage [V]
- RINU/RIND: Pull-up/pull-down resistance [W]
- CIN: Pin capacitance [F]
- CBOARD: Parasitic capacitance on the board [F]

13.2.3 CMOS Output and High Impedance State

The I/O cells except for analog output can output signals in the VDD and VSS levels. Also the GPIO ports may be put into high-impedance (Hi-Z) state.

13.3 Clock Settings

13.3.1 GPIO Operating Clock

When using the chattering filter for external GPIO inputs, the GPIO operating clock CLK_GPIO must be supplied to the GPIO module from the clock generator.
The CLK_GPIO clock should be controlled as in the procedure shown below.

1. Enable the clock source in the clock generator if it is stopped.
2. Write 0x0096 to the SYSPROT.PROT[15:0] bits. (Remove system protection)
3. Set the following PORTCLK register bits:
   - PORTCLK.CLKSRC bit (Clock source selection)
   - PORTCLK.CLKDIV[3:0] bits (Clock division ratio selection = Clock frequency setting)
4. Write a value other than 0x0096 to the SYSPROT.PROT[15:0] bits. (Set system protection)

Settings in Step 3 determine the input sampling time of the chattering filter.

13.4 Operations

13.4.1 Initialization

After a reset, the ports are configured as shown below.

- Port input: P0x and P1x dsabled
- Port output: P0x disabled, P1x enabled driving LOW
- Pull-up: Off
- Pull-down: Off
- Port pins: P0x high impedance state, P1x pins output enabled (low impedance)
- Port function: Configured to GPIO

This status continues until the ports are configured via software.

Initial settings when using a port for a peripheral I/O function

When using the Pxy port for a peripheral I/O function, perform the following software initial settings:

1. Set the following PORTPxIOEN register bits:
   - Set the PORTPxIOEN.PxIENy bit to 0. (Disable input)
   - Set the PORTPxIOEN.PxOENy bit to 0. (Disable output)
2. Set the PORTPxMODSEL.PxSELy bit to 0. (Disable peripheral I/O function)
3. Initialize the peripheral circuit that uses the pin.
4. Set the PORTPxFNCSEL.PxyMUX[1:0] bits. (Select peripheral I/O function)
5. Set the PORTPxMODSEL.PxSELy bit to 1. (Enable peripheral I/O function)

For the list of the peripheral I/O functions that can be assigned to each port of this IC, refer to the "GPIO Port Registers" section of this document. For the specific information on the peripheral I/O functions, refer to the respective peripheral circuit chapter.

Initial settings when using a port as a general-purpose output port

When using the Pxy port pin as a general-purpose output pin, perform the following software initial settings:

1. Set the PORTPxIOEN.PxOENy bit to 1. (Enable output)
2. Set the PORTPxMODSEL.PxSELy bit to 0. (Enable GPIO function)

Initial settings when using a port as a general-purpose input port

When using the Pxy port pin as a general-purpose input pin, perform the following software initial settings:

1. Write 0 to the PORTPxINTCTL.PxIEy bit. * (Disable interrupt)
2. When using the chattering filter, configure the GPIO operating clock (see "GPIO Operating Clock") and set the PORTPxCHATEN.PxCHATENy bit to 1. *
   When the chattering filter is not used, set the PORTPxCHATEN.PxCHATENy bit to 0 (supply of the GPIO operating clock is not required).
3. Configure the following PORTPxRCTL register bits when pulling up/down the port using the internal pull-up or down resistor:
   - PORTPxRCTL.PxPDPUy bit (Select pull-up or pull-down resistor)
   - Set the PORTPxRCTL.PxRENy bit to 1. (Enable pull-up/down)
   Set the PORTPxRCTL.PxRENy bit to 0 if the internal pull-up/down resistors are not used.
4. Set the PORTPxMODSEL.PxSELy bit to 0. (Enable GPIO function)
5. Configure the following bits when using the port input interrupt: *
   - Write 1 to the PORTPxINTF.PxIFy bit. (Clear interrupt flag)
   - PORTPxINTCTL.PxEDGEy bit (Select interrupt edge (input rising edge/falling edge))
   - Set the PORTPxINTCTL.PxIEy bit to 1. (Enable interrupt)
6. Set the following PORTPxIOEN register bits:
   - Set the PORTPxIOEN.PxOENy bit to 0. (Disable output)
   - Set the PORTPxIOEN.PxIENy bit to 1. (Enable input)

* Steps 1 and 5 are required for the ports with an interrupt function. Step 2 is required for the ports with a chattering filter function.

Table 13.1 lists the port status according to the combination of data input/output control and pull-up/down control.

Table 13.1 GPIO Port Control List

PORTPxIOEN. PxIENy bit	PORTPxIOEN. PxOENy bit	PORTPxRCTL. PxRENy bit	PORTPxRCTL. PxPDPUy bit	Input	Output	Pull-up/pull-down condition
0	0	0	X	Disabled		Off (Hi-Z) *1
0	0	1	0	Disabled		Pulled down
0	0	1	1	Disabled		Pulled up
1	0	0	x	Enabled	Disabled	Off (Hi-Z) *2
1	0	1	0	Enabled	Disabled	Pulled down
1	0	1	1	Enabled	Disabled	Pulled up
0	1	0	x	Disabled	Enabled	Off
0	1	1	0	Disabled	Enabled	Off
0	1	1	1	Disabled	Enabled	Off
1	1	1	0	Enabled	Enabled	Off
1	1	1	1	Enabled	Enabled	Off

*1 Initial status. Current does not flow if the pin is placed into floating status.
*2 Use of the pull-up or pull-down function is recommended, as undesired current will flow if the port input is set to floating status.

13.4.2 Port Input/Output Control

Peripheral I/O function control

The port for which a peripheral I/O function is selected is controlled by the peripheral circuit. For more information, refer to the respective peripheral circuit chapter.

Setting output data to a GPIO port

Write data (1 = high output, 0 = low output) to be output from the Pxy pin to the PORTPxDAT.PxOUTy bit.

Reading input data from a GPIO port

The data (1 = high input, 0 = low input) input from the Pxy pin can be read out from the PORTPxDAT.PxINy bit.

Chattering filter function

All ports have a chattering filter function and it can be controlled in each port. This function is enabled by setting the PORTPxCHATEN.PxCHATENy bit to 1. The input sampling time to remove chattering is determined by the CLK_GPIO frequency configured using the PORTCLK register in common to all ports. The chattering filter removes pulses with a shorter width than the input sampling time.

Input sampling time = 2 to 3 (1 / CLK_GPIO frequency [Hz]) [second]

Make sure the Pxy port interrupt is disabled before altering the PORTCLK register and PORTPxCHATEN. PxCHATENy bit settings. A Pxy port interrupt may erroneously occur if these settings are altered in an interrupt enabled status. Furthermore, enable the interrupt after a lapse of four or more CLK_GPIO cycles from enabling the chattering filter function.

13.5 Interrupts

When the GPIO function is selected for the port with an interrupt function, the port input interrupt function can be used.

Table 13.2 Port Input Interrupt Function

Interrupt	Interrupt flag	Set condition	Clear condition
Port input	PORTPxINTF.PxIFy	Rising or falling edge of the input signal	Writing 1
Interrupt	PORTINTFGRP.PxINT	Setting an interrupt flag in the port group	Clearing PORTPxINTF.PxIFy

Interrupt edge selection

Port input interrupts will occur at the falling edge of the input signal when setting the PORTPxINTCTL. PxEDGEy bit to 1, or the rising edge when setting to 0.

Interrupt enable

GPIO provides interrupt enable bits (PORTPxINTCTL.PxIEy bit) corresponding to each interrupt flag. An interrupt request is enabled only when the interrupt flag, of which interrupt has been enabled by the interrupt enable bit, is set.

Interrupt check in port group unit

When interrupts are enabled in two or more port groups, check the PORTINTFGRP.PxINT bit in the interrupt handler first. It helps minimize the handler codes for finding the port that has generated an interrupt. If this bit is set to 1, an interrupt has occurred in the port group. Next, check the PORTPxINTF.PxIFy bit set to 1 in the port group to determine the port that has generated an interrupt. Clearing the PORTPxINTF.PxIFy bit also clears the PORTINTFGRP.PxINT bit. If the port is set to interrupt disabled status by the PORTPxINTCTL. PxIEy bit, the PORTINTFGRP.PxINT bit will not be set even if the PORTPxINTF.PxIFy bit is set to 1.

Figure 13.3 shows a diagram of the PORTINT interrupt signal. The PORTINT signal is logical ORed with interrupt signals from other peripherals to generate the HIFIRQ interrupt signal to the host MCU.
See Section 22 (“Interrupts”) for more details on the HIFIRQ output.

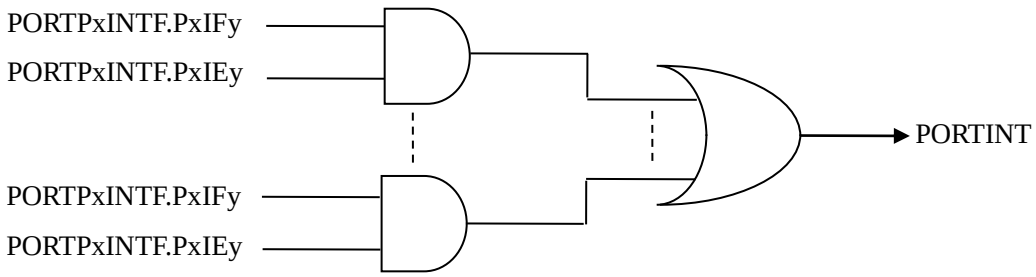


Figure 13.3 PORTINT Interrupt Circuit

13.6 Control Registers

See Section 10.4 for descriptions of the control registers for GPIO.

14. 16-bit Timers (T16)

14.1 Overview

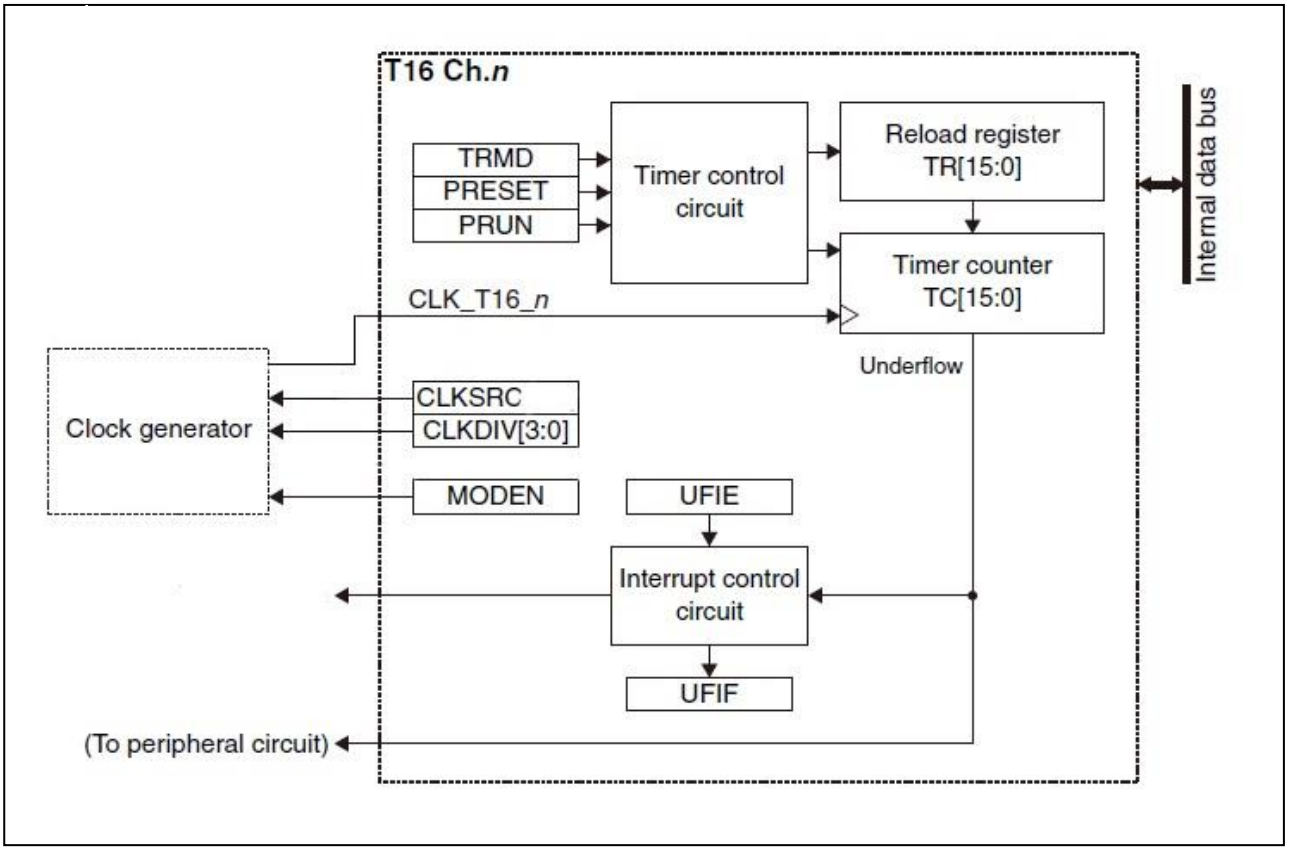
T16 is a 16-bit timer. The features of T16 are listed below.

- 16-bit presettable down counter
- Provides a reload data register for setting the preset value.
- A clock source and clock division ratio for generating the count clock are selectable.
- Repeat mode or one-shot mode is selectable.
- Can generate counter underflow interrupts.

There are two T16 channels (T16 CH1 and T16 CH2) in the S1D13C00. T16 CH1 is used as the clock source for the SPI and T16 CH2 is used as the clock source for the QSPI. However, both channels can be also be used as general-purpose timers.

Figure 14.1 shows the configuration of a T16 channel.

Figure 14.1 Configuration of a T16 Channel



14.2 Clock Settings

14.2.1 T16 Operating Clock

When using T16 Ch.n, the T16 Ch.n operating clock CLK_T16_n must be supplied to T16 Ch.n from the clock generator. The CLK_T16_n supply should be controlled as in the procedure shown below.

1. Enable the clock source in the clock generator if it is stopped.
2. Set the following T16_nCLK register bits:
   - T16_nCLK.CLKSRC bit (Clock source selection)
   - T16_nCLK.CLKDIV[3:0] bits (Clock division ratio selection = Clock frequency setting)

14.3 Operations

14.3.1 Initialization

T16 Ch.n should be initialized and started counting with the procedure shown below.

1. Configure the T16 Ch.n operating clock (see "T16 Operating Clock").
2. Set the T16_nCTL.MODEN bit to 1. (Enable count operation clock)
3. Set the T16_nMOD.TRMD bit. (Select operation mode (Repeat mode or One-shot mode)).
4. Set the T16_nTR register. (Set reload data (counter preset data))
5. Set the following bits when using the interrupt:
   - Write 1 to the T16_nINTF.UFIF bit. (Clear interrupt flag)
   - Set the T16_nINTE.UFIE bit to 1. (Enable underflow interrupt)
6. Set the following T16_nCTL register bits:
   - Set the T16_nCTL.PRESET bit to 1. (Preset reload data to counter)
   - Set the T16_nCTL.PRUN bit to 1. (Start counting)

14.3.2 Counter Underflow

Normally, the T16 counter starts counting down from the reload data value preset and generates an underflow signal when an underflow occurs. This signal is used to generate an interrupt and may be output to a specific peripheral circuit as a clock (T16 Ch.n must be set to repeat mode to generate a clock). The underflow cycle is determined by the T16 Ch.n operating clock setting and reload data (counter initial value) set in the T16_nTR register. The following shows the equations to calculate the underflow cycle and frequency:

$$T = (TR + 1) / f_{CLK\_T16\_n} \qquad f_T = f_{CLK\_T16\_n} / (TR + 1)$$

Where

T:	Underflow cycle [s]
f_T:	Underflow frequency [Hz]
TR:	T16_*n*TR register setting
$f_{CLK\_T16\_n}$:	T16 CH*n* operating clock frequency [Hz]

14.3.3 Operations in Repeat Mode

T16 Ch.n enters repeat mode by setting the T16_nMOD.TRMD bit to 0.
In repeat mode, the count operation starts by writing 1 to the T16_nCTL.PRUN bit and continues until 0 is written. A counter underflow presets the T16_nTR register value to the counter, so underflow occurs periodically. Select this mode to generate periodic underflow interrupts or when using the timer to output a trigger/clock to the peripheral circuit.

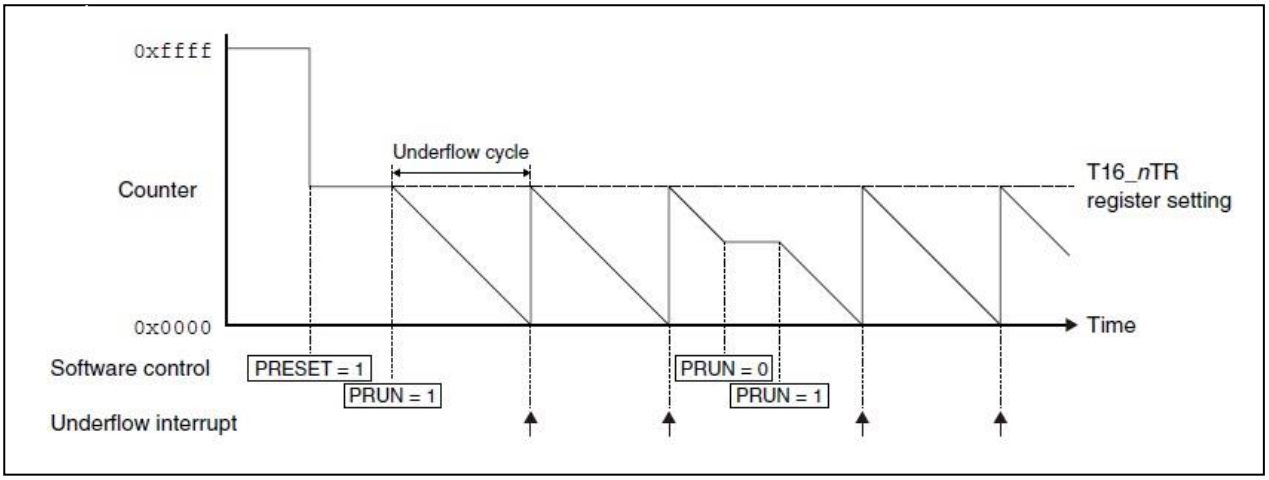


Figure 14.2 Count Operations in Repeat Mode

14.3.4 Operations in One-shot Mode

T16 Ch.n enters one-shot mode by setting the T16_nMOD.TRMD bit to 1.
In one-shot mode, the count operation starts by writing 1 to the T16_nCTL.PRUN bit and stops after the T16_nTR register value is preset to the counter when an underflow has occurred. At the same time the counter stops, the T16_nCTL.PRUN bit is cleared automatically. Select this mode to stop the counter after an interrupt has occurred once, such as for checking a specific lapse of time.

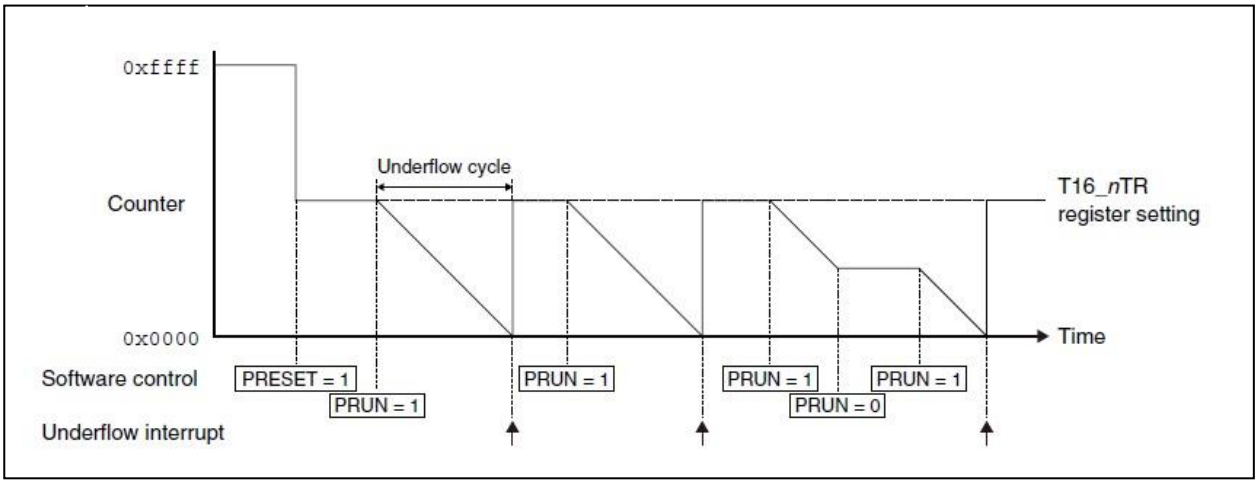


Figure 14.3 Count Operations in One-shot Mode

14.3.5 Counter Value Read

The counter value can be read out from the T16_nTC.TC[15:0] bits. However, since T16 operates on CLK_T16_n, one of the operations shown below is required to read correctly by the host MCU.

- Read the counter value twice or more and check to see if the same value is read.
- Stop the timer and then read the counter value.

14.4 Interrupt

Each T16 channel has a function to generate the interrupt shown in Table 14.1.

Table 14.1 T16 Interrupt Function

Interrupt	Interrupt flag	Set condition	Clear condition
Underflow	T16_nINTF.UFIF	When the counter underflows	Writing 1

T16 provides interrupt enable bits corresponding to each interrupt flag. An interrupt request is sent to the host MCU only when the interrupt flag, of which interrupt has been enabled by the interrupt enable bit, is set.

Figure 14.4 shows a diagram of the T16_nINT interrupt signal. The T16_nINT signal is logical ORed with interrupt signals from other peripherals to generate the HIFIRQ interrupt signal to the host MCU.
See Section 22 (“Interrupts”) for more details on the HIFIRQ output.

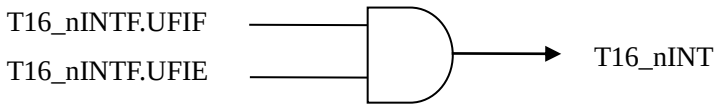


Figure 14.4 T16_nINT Interrupt Circuit

14.5 Control Registers

See Section 10.5 and 10.7 for descriptions of the control registers for T16 CH1 and CH2, respectively.

15. SPI Interface

15.1 Overview

SPI is a synchronous serial interface. The features of SPI are listed below.

- Supports both master and slave modes.
- Data length: 2 to 16 bits programmable
- Either MSB first or LSB first can be selected for the data format.
- Clock phase and polarity are configurable.
- Supports full-duplex communications.
- Includes separated transmit data buffer and receive data buffer registers.
- Can generate receive buffer full, transmit buffer empty, end of transmission, and overrun interrupts.
- Can issue a DMA transfer request when a receive buffer full or a transmit buffer empty occurs.
- Master mode allows use of a 16-bit timer to set baud rate.
- Slave mode is capable of being operated with the external input clock SPICLK only.
- Input pins can be pulled up/down with an internal resistor.

Figure 15.1 shows the configuration of the SPI module. It uses either the clock source or the CLK_T16 output of the T16 CH1 timer as its clock source.

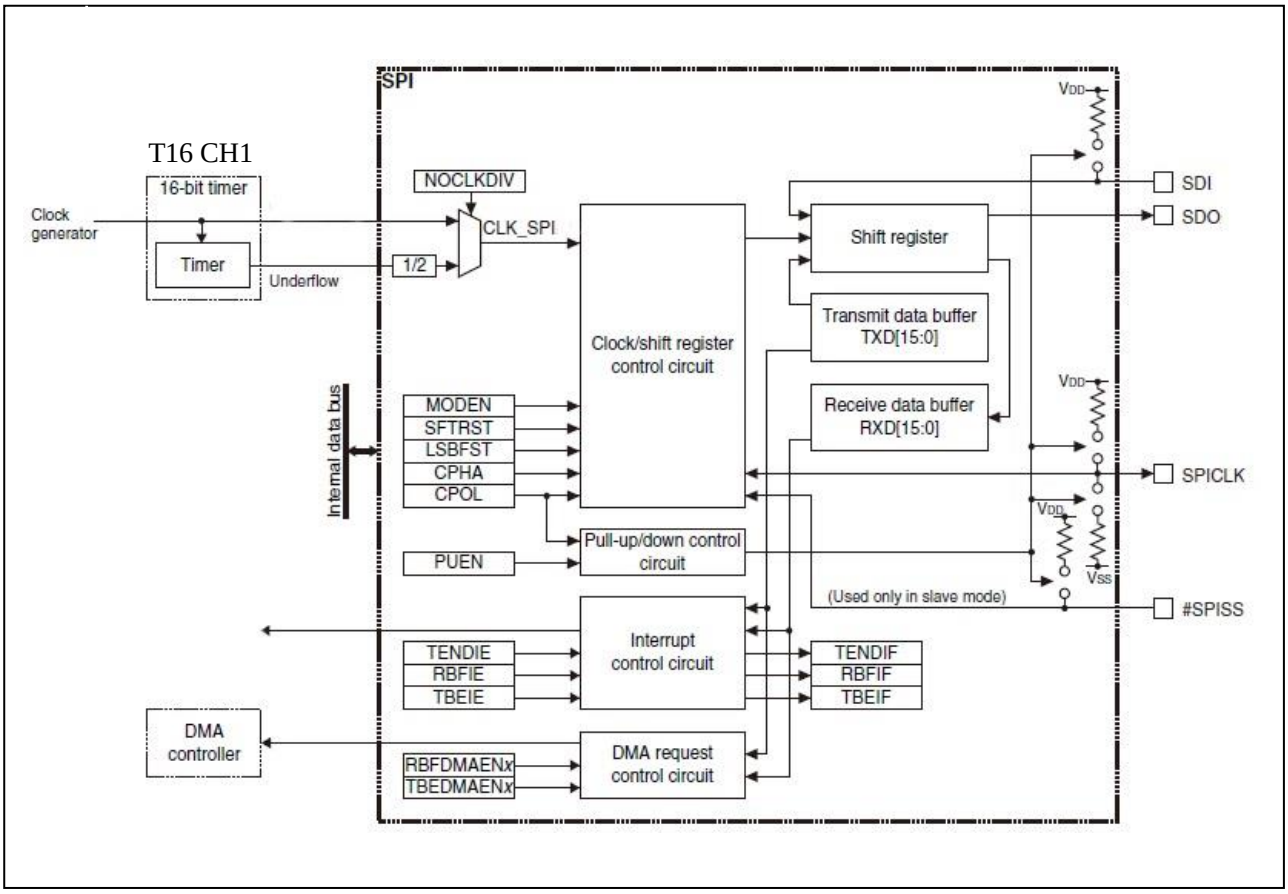


Figure 15.1 SPI Configuration

15.2 Input/Output Pins and External Connections

15.2.1 List of Input/Output Pins

Table 15.1 lists the SPI pins.

Table 15.1 List of SPI Pins

Pin name	I/O*	Initial status*	Function
SDI	I	I (Hi-Z)	SPI data input pin
SDO	O or Hi-Z	Hi-Z	SPI data output pin
SPICLK	I or O	I (Hi-Z)	SPI external clock input/output pin
#SPISS	I	I (Hi-Z)	SPI slave select signal input pin

* Indicates the status when the pin is configured for the SPI.

If the port is shared with the SPI pin and other functions, the SPI input/output function must be assigned to the port before activating SPI. For more information, refer to the "GPIO Ports" chapter.

15.2.2 External Connections

SPI operates in master mode or slave mode. Figure 15.2 and Figure 15.3 show connection diagrams between SPI in each mode and external SPI devices.

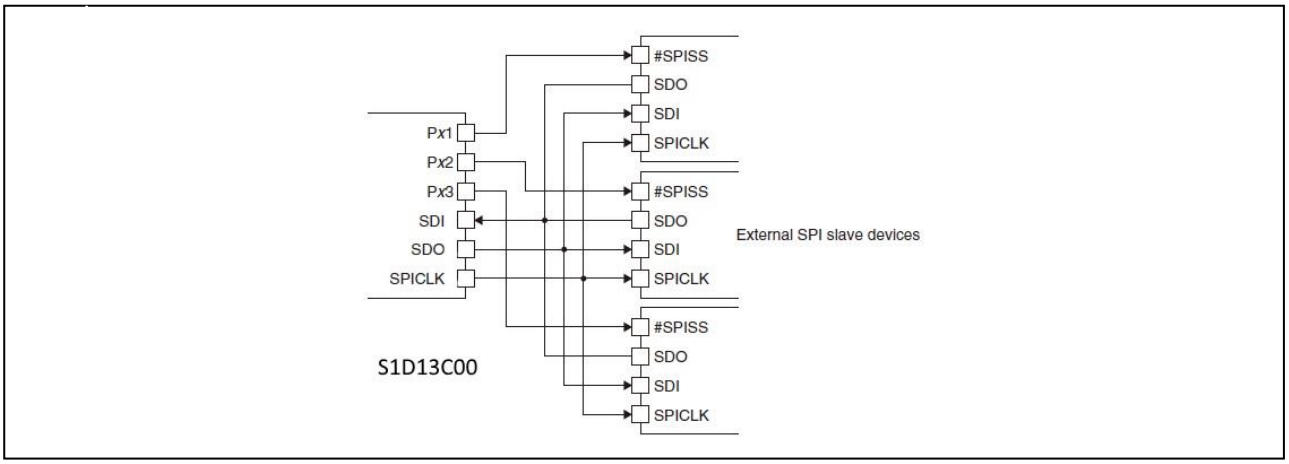


Figure 15.2 Connections between SPI in Master Mode and External SPI Slave Devices

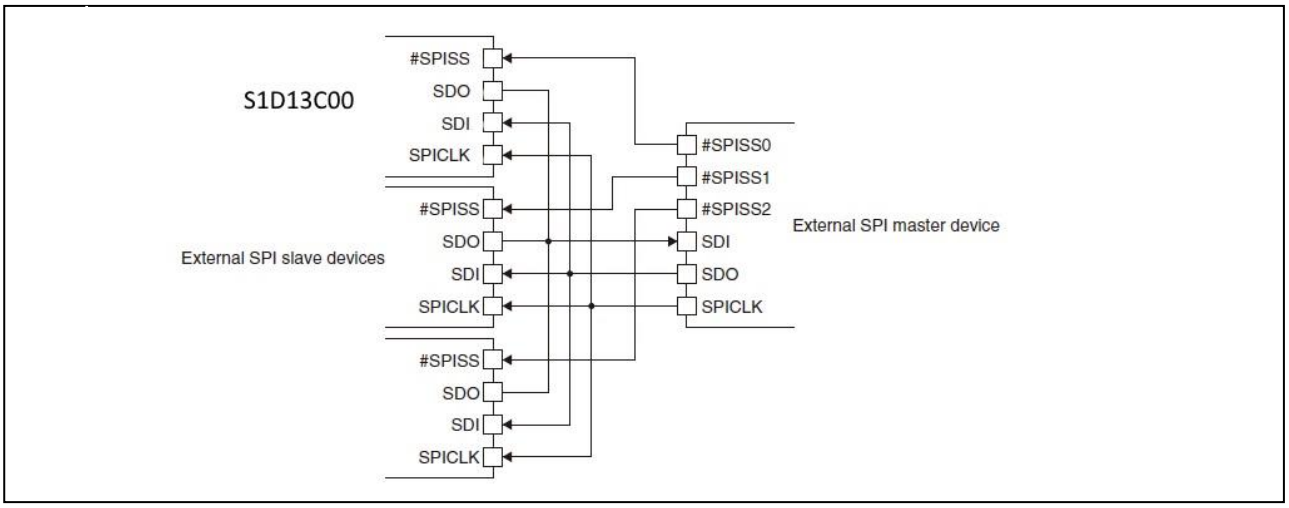


Figure 15.3 Connections between SPI in Slave Mode and External SPI Master Device

15.2.3 Pin Functions in Master Mode and Slave Mode

The pin functions are changed according to the master or slave mode selection. The differences in pin functions between the modes are shown in Table 15.2.

Table 15.2 Pin Function Differences between Modes

Pin	Function in master mode	Function in slave mode
SDI	Always placed into input state.	
SDO	Always placed into output state.	This pin is placed into output state while a low level is applied to the #SPISS pin or placed into Hi-Z state while a high level is applied to the #SPISS pin.
SPICLK	Outputs the SPI clock to external devices. Output clock polarity and phase can be configured if necessary.	Inputs an external SPI clock. Clock polarity and phase can be designated according to the input clock.
#SPISS	Not used. This input function is not required to be assigned to the port. To output the slave select signal in master mode, use a general-purpose I/O port function.	Applying a low level to the #SPISS pin enables SPI to transmit/receive data. While a high level is applied to this pin, SPI is not selected as a slave device. Data input to the SDI pin and the clock input to the SPICLK pin are ignored. When a high level is applied, the transmit/receive bit count is cleared to 0 and the already received bits are discarded.

15.2.4 Input Pin Pull-Up/Pull-Down Function

The SPI input pins (SDI in master mode or SDI, SPICLK, and #SPISS pins in slave mode) have a pull-up or pull-down function as shown in Table 15.3. This function is enabled by setting the SPIMOD.PUEN bit to 1.

Table 15.3 Pull-Up or Pull-Down of Input Pins

Pin	Master mode	Slave mode
SDI	Pull-up	Pull-up
SPICLK	-	SPIMOD.CPOL bit = 1: Pull-up SPIMOD.CPOL bit = 0: Pull-down
#SPISS	-	Pull-up

15.3 Clock Settings

15.3.1 SPI Operating Clock

Operating clock in master mode

In master mode, the SPI operating clock is supplied from the 16-bit timer (T16 CH1). The following two options are provided for the clock configuration.

Use the 16-bit timer operating clock without dividing

By setting the SPIMOD.NOCLKDIV bit to 1, the operating clock CLK_T16_1, which is configured by selecting a clock source and a division ratio, for the 16-bit timer channel corresponding to the SPI is input to SPI as CLK_SPI. Since this clock is also used as the SPI clock SPICLK without changing, the CLK_SPI frequency becomes the baud rate. To supply CLK_SPI to SPI, the 16-bit timer clock source must be enabled in the clock generator. It does not matter how the T16_1CTL.MODEN and T16_1CTL.PRUN bits of the corresponding 16-bit timer channel are set (1 or 0).
When setting this mode, the timer function of the corresponding 16-bit timer channel may be used for another purpose.

Use the 16-bit timer as a baud rate generator

By setting the SPIMOD.NOCLKDIV bit to 0, SPI inputs the underflow signal generated by the corresponding 16-bit timer channel and converts it to the SPICLK. The 16-bit timer must be run with an appropriate reload data set. The SPICLK frequency (baud rate) and the 16-bit timer reload data are calculated by the equations shown below.

$$f_{SPICLK} = \frac{f_{CLK\_SPI}}{2 \times (RLD + 1)} \qquad RLD = \frac{f_{CLK\_SPI}}{f_{SPICLK} \times 2} - 1$$

Where

f_{SPICLK}: SPICLK frequency [Hz] (= baud rate [bps])
f_{CLK_SPI} : SPI operating clock frequency [Hz]
RLD: 16-bit timer reload data value

Operating clock in slave mode

SPI set in slave mode operates with the clock supplied from the external SPI master to the SPICLK pin. The 16-bit timer channel (including the clock source selector and the divider) corresponding to the SPI channel is not used. Furthermore, the SPIMOD.NOCLKDIV bit setting becomes ineffective. SPI keeps operating using the clock supplied from the external SPI master even if all the internal clocks are halted, so SPI can receive data and can generate receive buffer full interrupts.

15.3.2 SPI Clock (SPICLK) Phase and Polarity

The SPICLK phase and polarity can be configured separately using the SPIMOD.CPHA bit and the SPIMOD.CPOL bit, respectively. Figure 15.4 shows the clock waveform and data input/output timing in each setting.

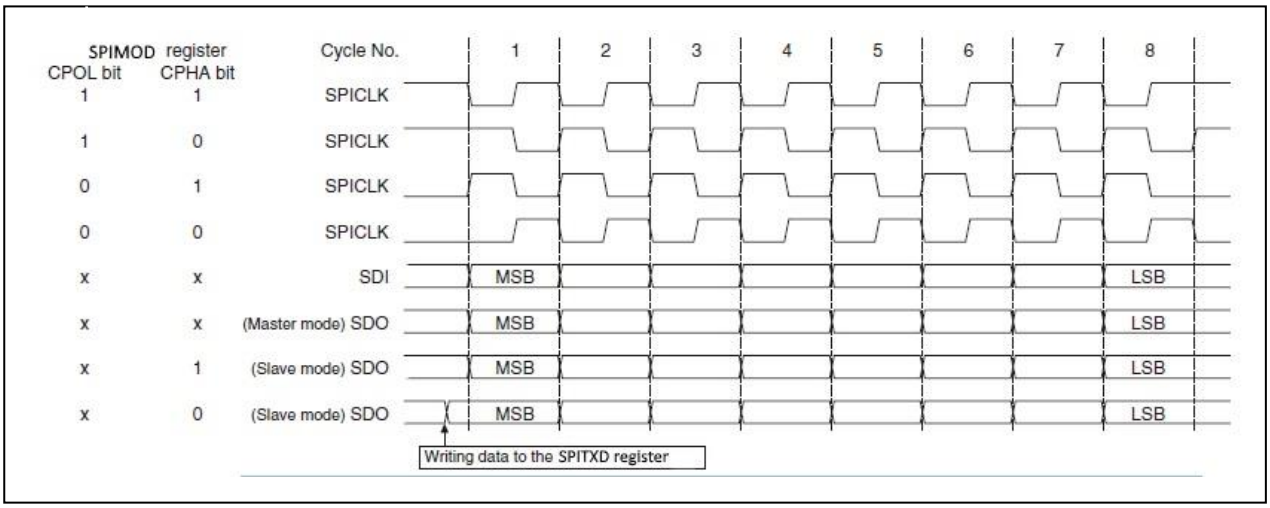


Figure 15.4 SPI Clock Phase and Polarity (SPIMOD.LSBFST bit = 0, SPIMOD.CHLN[3:0] bits = 0x7)

15.4 Data Format

The SPI data length can be selected from 2 bits to 16 bits by setting the SPIMOD.CHLN[3:0] bits. The input/ output permutation is configurable to MSB first or LSB first using the SPIMOD.LSBFST bit. Figure 15.5 shows a data format example when the SPIMOD.CHLN[3:0] bits = 0x7, the SPIMOD.CPOL bit = 0 and the SPIMOD.CPHA bit = 0.

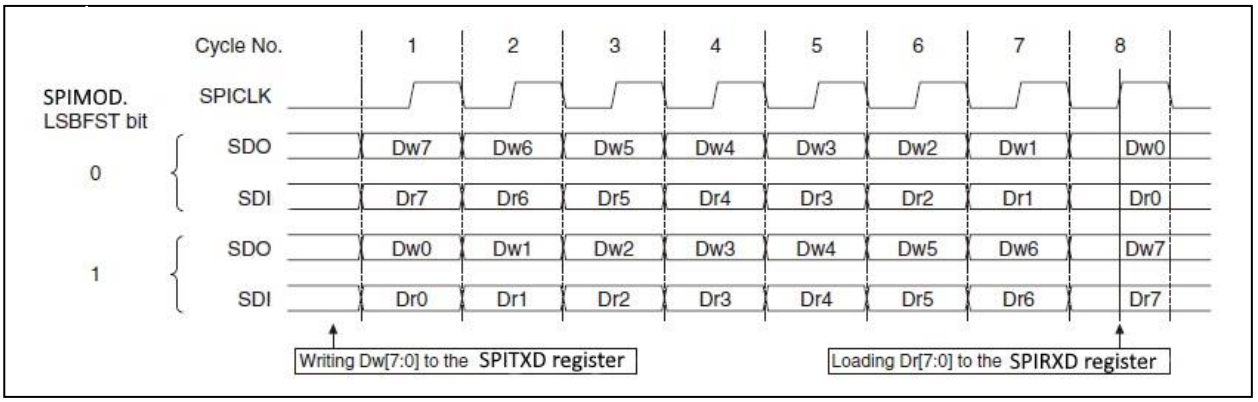


Figure 15.5 Data Format Selection Using the SPIMOD.LSBFST Bit
(SPIMOD.CHLN[3:0] bits = 0x7, SPIMOD.CPOL bit = 0, SPIMOD.CPHA bit = 0)

15.5 Operations

15.5.1 Initialization

SPI should be initialized with the procedure shown below.

7. <Master mode only> Generate a clock by controlling the 16-bit timer and supply it to SPI.
8. Configure the following SPIMOD register bits:
   - SPIMOD.PUEN bit (Enable input pin pull-up/down)
   - SPIMOD.NOCLKDIV bit (Select master mode operating clock)
   - SPIMOD.LSBFST bit (Select MSB first/LSB first)
   - SPIMOD.CPHA bit (Select clock phase)
   - SPIMOD.CPOL bit (Select clock polarity)
   - SPIMOD.MST bit (Select master/slave mode)
9. Assign the SPI input/output function to the ports. (Refer to the "GPIO Ports" chapter.)
10. Set the following SPICTL register bits:
    - Set the SPICTL.SFTRST bit to 1. (Execute software reset)
    - Set the SPICTL.MODEN bit to 1. (Enable SPI operations)
11. Set the following bits when using the interrupt:
    - Write 1 to the interrupt flags in the SPIINTF register. (Clear interrupt flags)
    - Set the interrupt enable bits in the SPIINTE register to 1. * (Enable interrupts)

    * The initial value of the SPIINTF.TBEIF bit is 1, therefore, an interrupt will occur immediately after the SPIINTE.TBEIE bit is set to 1.
12. Configure the DMA controller and set the following SPI control bits when using DMA transfer:
    - Write 1 to the DMA transfer request enable bits in the SPITBEDMAEN and SPIRBFDMAEN registers. (Enable DMA transfer requests)

15.5.2 Data Transmission in Master Mode

A data sending procedure and operations in master mode are shown below. Figure 15.6 and Figure 15.7 show a timing chart and a flowchart, respectively.

Data sending procedure

1. Assert the slave select signal by controlling the general-purpose output port (if necessary).
2. Check to see if the SPIINTF.TBEIF bit is set to 1 (transmit buffer empty).
3. Write transmit data to the SPITXD register.
4. Wait for an SPI interrupt when using the interrupt.
5. Repeat Steps 2 to 4 (or 2 and 3) until the end of transmit data.
6. Negate the slave select signal by controlling the general-purpose output port (if necessary).

Data sending operations

SPI starts data sending operations when transmit data is written to the SPITXD register. The transmit data in the SPITXD register is automatically transferred to the shift register and the SPIINTF.TBEIF bit is set to 1. If the SPIINTE.TBEIE bit = 1 (transmit buffer empty interrupt enabled), a transmit buffer empty interrupt occurs at the same time.

The SPICLK pin outputs clocks of the number of the bits specified by the SPIMOD.CHLN[3:0] bits and the transmit data bits are output in sequence from the SDO pin in sync with these clocks.

Even if the clock is being output from the SPICLK pin, the next transmit data can be written to the SPITXD register after making sure the SPIINTF.TBEIF bit is set to 1.

If transmit data has not been written to the SPITXD register after the last clock is output from the SPICLK pin, the clock output halts and the SPIINTF.TENDIF bit is set to 1. At the same time SPI issues an end-of-transmission interrupt request if the SPIINTE.TENDIE bit = 1.

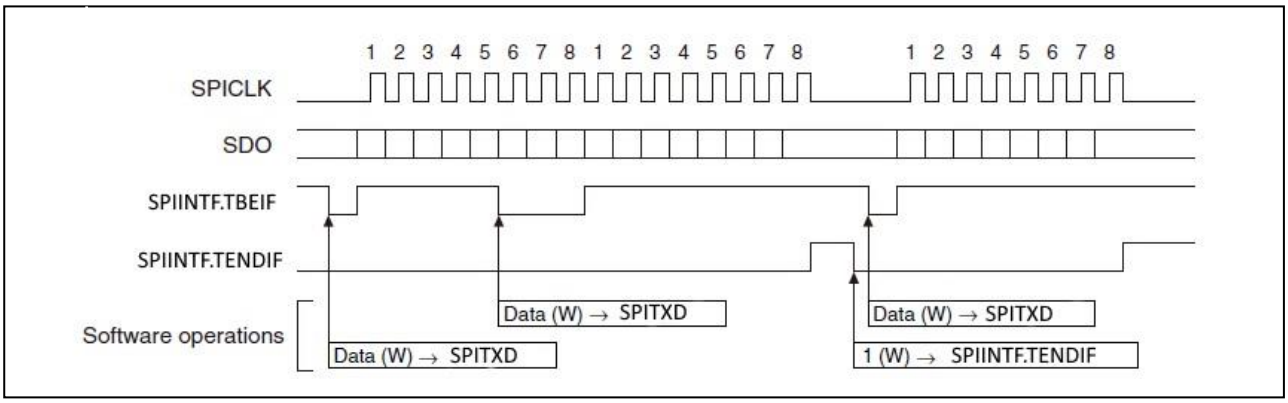


Figure 15.6 Example of Data Sending Operations in Master Mode (SPIMOD.CHLN[3:0] bits = 0x7)

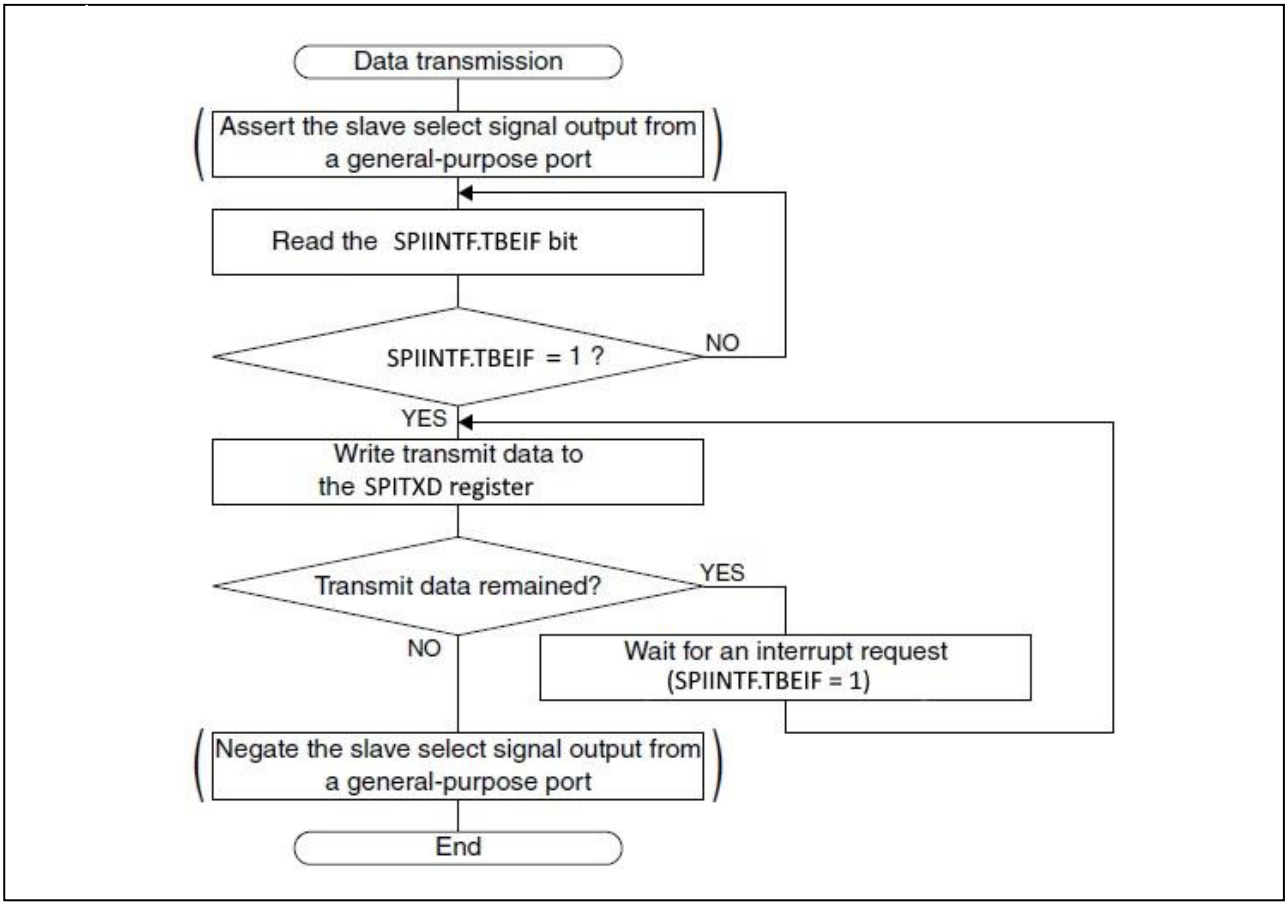


Figure 15.7 Data Transmission Flowchart in Master Mode

Data transmission using DMA

By setting the SPITBEDMAEN.TBEDMAENx bit to 1 (DMA transfer request enabled), a DMA transfer request is sent to the DMA controller and transmit data is transferred from the specified memory to the SPITXD register via DMA Ch.x when the SPIINTF.TBEIF bit is set to 1 (transmit buffer empty).
This automates the procedure from Step 2 to Step 5 described above.
The transfer source/destination and control data must be set for the DMA controller and the relevant DMA channel must be enabled to start a DMA transfer in advance so that transmit data will be transferred to the SPITXD register. For more information on DMA, refer to the "DMA Controller" chapter.

Table 15.4 DMA Data Structure Configuration Example (for 16-bit Data Transmission)

Item		Setting example
End pointer	Transfer source	Memory address in which the last transmit data is stored
	Transfer destination	SPITXD register address
Control data	dst_inc	0x3 (no increment)
	dst_size	0x1 (halfword)
	src_inc	0x1 (+2)
	src_size	0x1 (halfword)
	R_power	0x0 (arbitrated for every transfer)
	n_minus_1	Number of transfer data
	cycle_ctrl	0x1 (basic transfer)

15.5.3 Data Reception in Master Mode

A data receiving procedure and operations in master mode are shown below. Figure 15.8 and Figure 15.9 show a timing chart and flowcharts, respectively.

Data receiving procedure

1. Assert the slave select signal by controlling the general-purpose output port (if necessary).
2. Check to see if the SPIINTF.TBEIF bit is set to 1 (transmit buffer empty).
3. Write dummy data (or transmit data) to the SPITXD register.
4. Wait for a transmit buffer empty interrupt (SPIINTF.TBEIF bit = 1).
5. Write dummy data (or transmit data) to the SPITXD register.
6. Wait for a receive buffer full interrupt (SPIINTF.RBFIF bit = 1).
7. Read the received data from the SPIRXD register.
8. Repeat Steps 5 to 7 until the end of data reception.
9. Negate the slave select signal by controlling the general-purpose output port (if necessary).

Note: To perform continuous data reception without stopping SPICLK, Steps 7 and 5 operations must be completed within the SPICLK cycles equivalent to "Data bit length - 1" after Step 6.

Data receiving operations

SPI starts data receiving operations simultaneously with data sending operations when transmit data (may be dummy data if data transmission is not required) is written to the SPITXD register.
The SPICLK pin outputs clocks of the number of the bits specified by the SPIMOD.CHLN[3:0] bits. The transmit data bits are output in sequence from the SDOn pin in sync with these clocks and the receive data bits input from the SDI pin are shifted into the shift register.
When the last clock is output from the SPICLK pin and receive data bits are all shifted into the shift register, the received data is transferred to the receive data buffer and the SPIINTF.RBFIF bit is set to 1. At the same time SPI issues a receive buffer full interrupt request if the SPIINTE.RBFIE bit = 1. After that, the received data in the receive data buffer can be read through the SPIRXD register.

Note: If data of the number of the bits specified by the SPIMOD.CHLN[3:0] bits is received when the SPIINTF.RBFIF bit is set to 1, the SPIRXD register is overwritten with the newly received data and the previously received data is lost. In this case, the SPIINTF.OEIF bit is set.

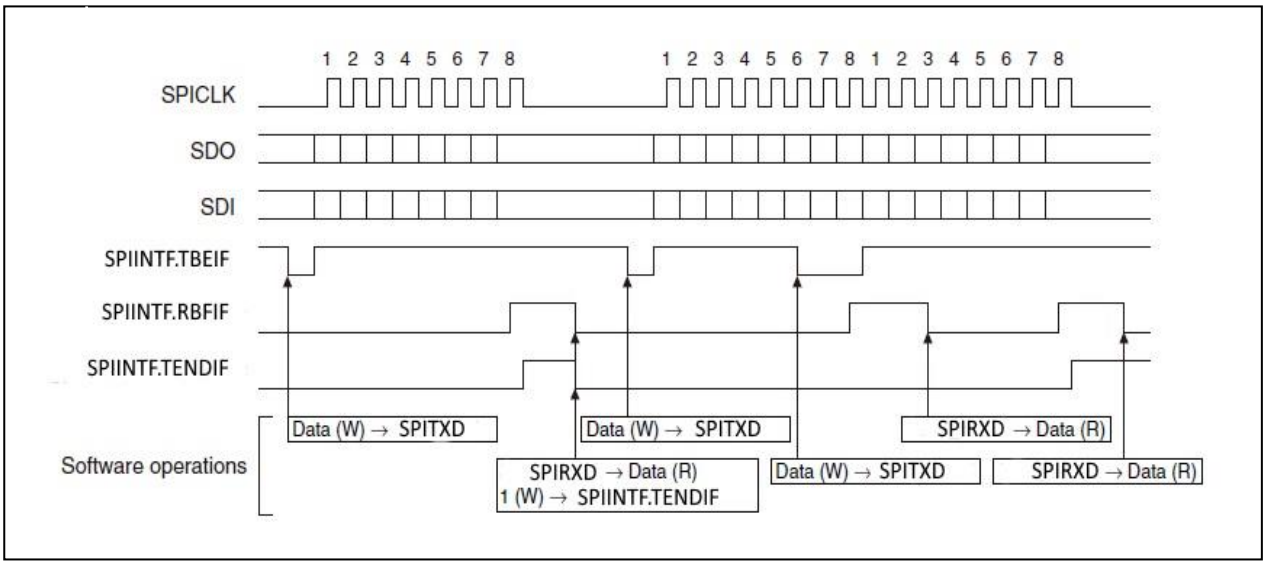


Figure 15.8 Example of Data Receiving Operations in Master Mode (SPIMOD.CHLN[3:0] bits = 0x7)

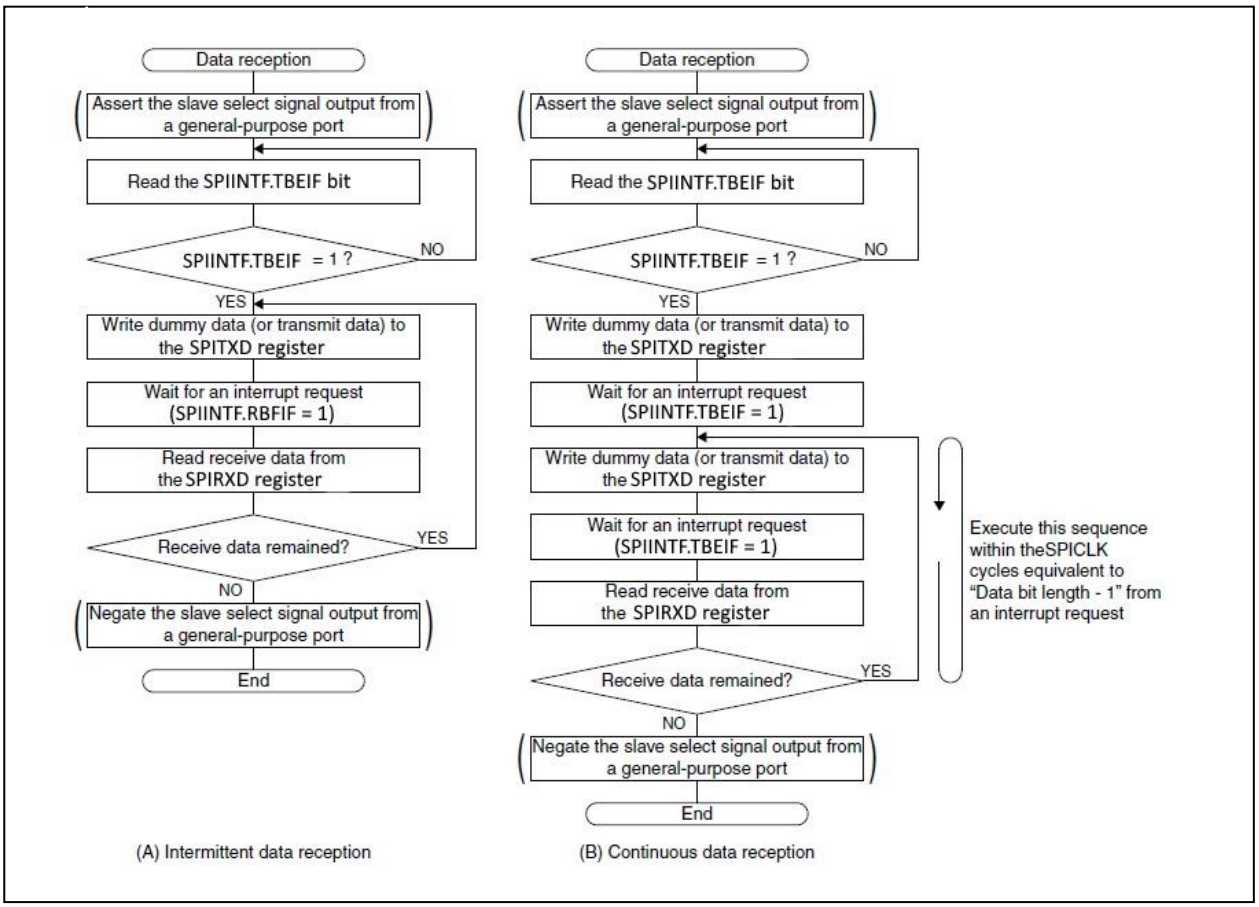


Figure 15.9 Data Reception Flowcharts in Master Mode

Data reception using DMA

For data reception, two DMA controller channels should be used to write dummy data to the SPITXD register as a reception start trigger and to read the received data from the SPIRXD register.

By setting the SPITBEDMAEN.TBEDMAENx1 bit to 1 (DMA transfer request enabled), a DMA transfer request is sent to the DMA controller and dummy data is transferred from the specified memory to the SPITXD register via DMA Ch.x1 when the SPIINTF.TBEIF bit is set to 1 (transmit buffer empty).

By setting the SPIRBFDMAEN.RBFDMAENx2 bit to 1 (DMA transfer request enabled), a DMA transfer

request is sent to the DMA controller and the received data is transferred from the SPIRXD register to the specified memory via DMA Ch.x2 when the SPIINTF.RBFIF bit is set to 1 (receive buffer full).
This automates the procedure from Step 2 to Step 8 described above.
The transfer source/destination and control data must be set for the DMA controller and the relevant DMA channel must be enabled to start a DMA transfer in advance. For more information on DMA, refer to the "DMA Controller" chapter.

Table 15.5 DMA Data Structure Configuration Example (for Writing 16-bit Dummy Transmit Data)

Item		Setting example
End pointer	Transfer source	Memory address in which dummy data is stored
	Transfer destination	SPITXD register address
Control data	dst_inc	0x3 (no increment)
	dst_size	0x1 (halfword)
	src_inc	0x3 (no increment)
	src_size	0x1 (halfword)
	R_power	0x0 (arbitrated for every transfer)
	n_minus_1	Number of transfer data
	cycle_ctrl	0x1 (basic transfer)

Table 15.6 DMA Data Structure Configuration Example (for 16-bit Data Reception)

Item		Setting example
End pointer	Transfer source	SPIRXD register address
	Transfer destination	Memory address to which the last received data is stored
Control data	dst_inc	0x1 (+2)
	dst_size	0x1 (halfword)
	src_inc	0x3 (no increment)
	src_size	0x1 (halfword)
	R_power	0x0 (arbitrated for every transfer)
	n_minus_1	Number of transfer data
	cycle_ctrl	0x1 (basic transfer)

15.5.4 Terminating Data Transfer in Master Mode

A procedure to terminate data transfer in master mode is shown below.

1. Wait for an end-of-transmission interrupt (SPIINTF.TENDIF bit = 1).
2. Set the SPICTL.MODEN bit to 0 to disable the SPI operations.
3. Stop the 16-bit timer to disable the clock supply to SPI.

15.5.5 Data Transfer in Slave Mode

A data sending/receiving procedure and operations in slave mode are shown below. Figure 15.10 and Figure 15.11 show a timing chart and flowcharts, respectively.

Data sending procedure

1. Check to see if the SPIINTF.TBEIF bit is set to 1 (transmit buffer empty).
2. Write transmit data to the SPITXD register.
3. Wait for a transmit buffer empty interrupt (SPIINTF.TBEIF bit = 1).
4. Repeat Steps 2 and 3 until the end of transmit data.

Note: Transmit data must be written to the SPITXD register after the SPIINTF.TBEIF bit is set to 1 by the time the sending SPITXD register data written is completed. If no transmit data is written during this period, the data bits input from the SDI pin are shifted and output from the SDO pin without being modified.

Data receiving procedure

1. Wait for a receive buffer full interrupt (SPIINTF.RBFIF bit = 1).
2. Read the received data from the SPIRXD register.
3. Repeat Steps 1 and 2 until the end of data reception.

Data transfer operations

The following shows the slave mode operations different from master mode:

- Slave mode operates with the SPI clock supplied from the external SPI master to the SPICLK pin. The data transfer rate is determined by the SPICLK frequency. It is not necessary to control the 16-bit timer.
- SPI can operate as a slave device only when the slave select signal input from the external SPI master to the #SPISS pin is set to the active (low) level.
  If #SPISS = high, the software transfer control, the SPICLK pin input, and the SDI pin input are all ineffective. If the #SPISS signal goes high during data transfer, the transfer bit counter is cleared and data in the shift register is discarded.
- Slave mode starts data transfer when SPICLK is input from the external SPI master after the #SPISS signal is asserted. Writing transmit data is not a trigger to start data transfer. Therefore, it is not necessary to write dummy data to the transmit data buffer when performing data reception only.

NOTES:

- If data of the number of bits specified by the SPIMOD.CHLN[3:0] bits is received when the SPIINTF.RBFIF bit is set to 1, the SPIRXD register is overwritten with the newly received data and the previously received data is lost. In this case, the SPIINTF.OEIF bit is set.
- When the clock for the first bit is input from the SPICLK pin, SPI starts sending the data currently stored in the shift register even if the SPIINTF.TBEIF bit is set to 1.

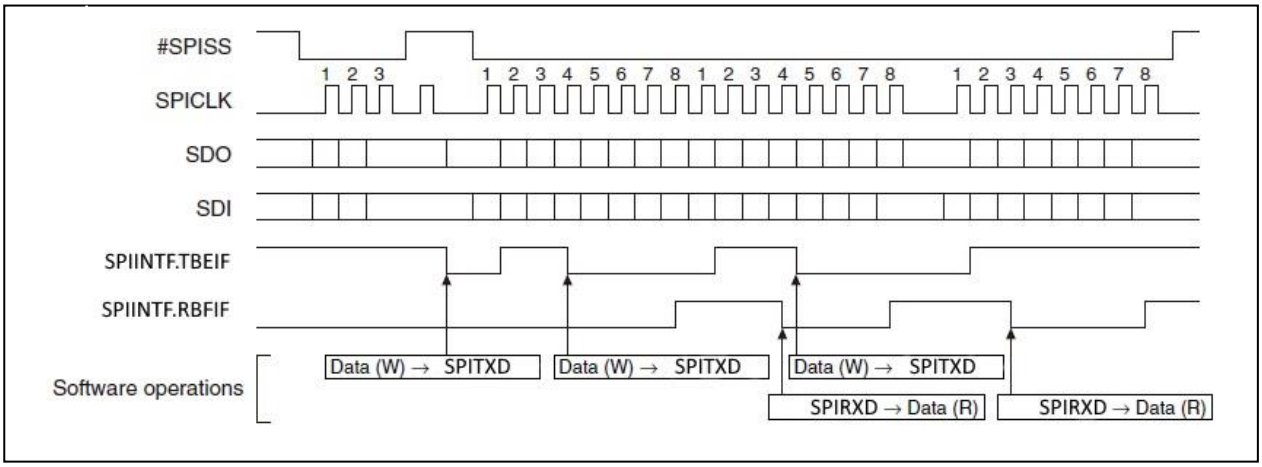


Figure 15.10 Example of Data Transfer Operations in Slave Mode (SPIMOD.CHLN[3:0] bits = 0x7)

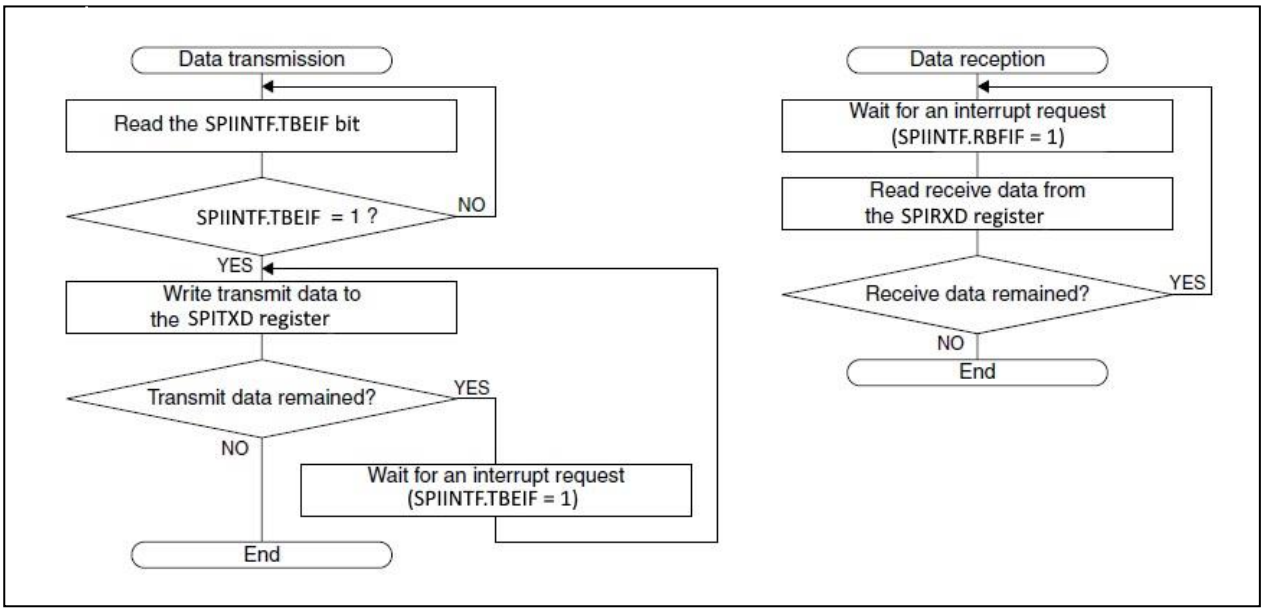


Figure 15.11 Data Transfer Flowcharts in Slave Mode

15.5.6 Terminating Data Transfer in Slave Mode

A procedure to terminate data transfer in slave mode is shown below.

1. Wait for an end-of-transmission interrupt (SPIINTF.TENDIF bit = 1). Or determine end of transfer via the received data.
2. Set the SPICTL.MODEN bit to 0 to disable the SPI operations.

15.6 Interrupts

SPI has a function to generate the interrupts shown in Table 15.7.

Table 15.7 SPI Interrupt Function

Interrupt	Interrupt flag	Set condition	Clear condition
End of transmission	SPIINTF.TENDIF	When the SPIINTF.TBEIF bit = 1 after data of the specified bit length (defined by the SPIMOD.CHLN[3:0] bits) has been sent	Writing 1
Receive buffer full	SPIINTF.RBFIF	When data of the specified bit length is received and the received data is transferred from the shift register to the received data buffer	Reading the SPIRXD register
Transmit buffer empty	SPIINTF.TBEIF	When transmit data written to the transmit data buffer is transferred to the shift register	Writing to the SPITXD register
Overrun error	SPIINTF.OEIF	When the receive data buffer is full (when the received data has not been read) at the point that receiving data to the shift register has completed	Writing 1

SPI provides interrupt enable bits corresponding to each interrupt flag. An interrupt request is sent to the Host MCU only when the interrupt flag, of which interrupt has been enabled by the interrupt enable bit, is set.
The SPIINTF register also contains the BSY bit that indicates the SPI operating status.
Figure 15.12 shows the SPIINTF.BSY and SPIINTF.TENDIF bit set timings.

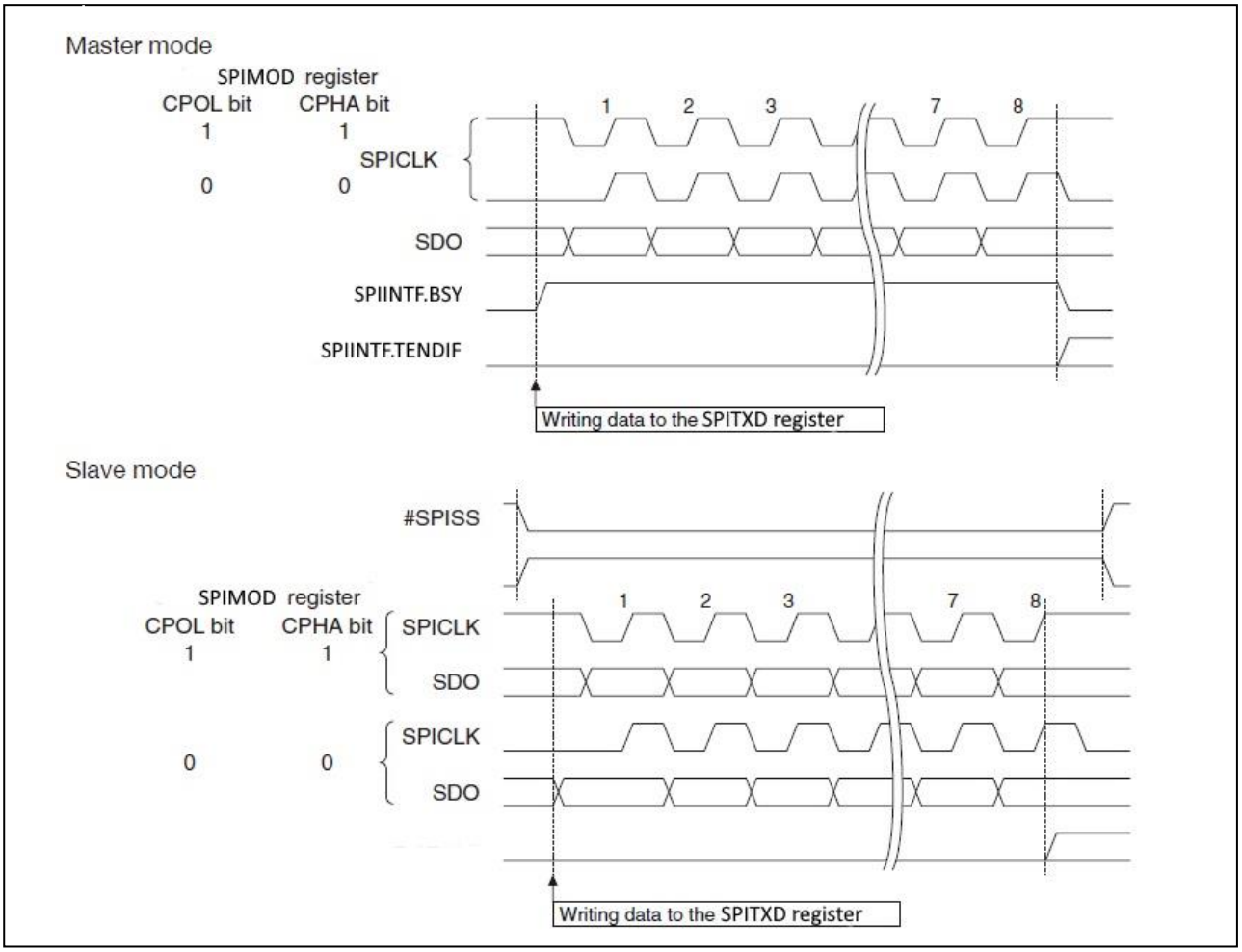


Figure 15.12 SPIINTF.BSY and SPIINTF.TENDIF Bit Set Timings (when SPIMOD.CHLN[3:0] bits = 0x7)

Figure 15.13 shows a diagram of the SPIINT interrupt signal. The SPIINT signal is logical ORed with interrupt signals from other peripherals to generate the HIFIRQ interrupt signal to the host MCU.
See Section 22 (“Interrupts”) for more details on the HIFIRQ output.

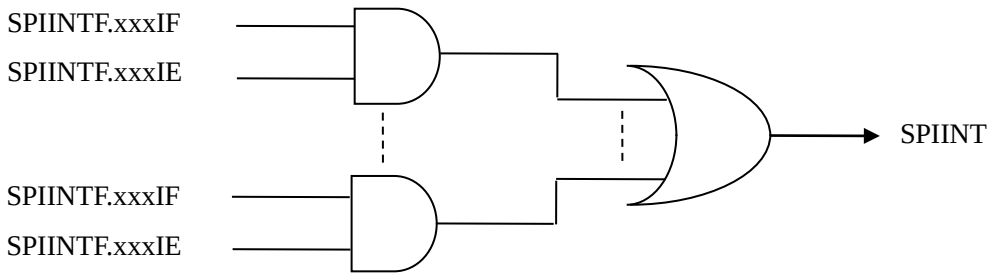


Figure 15.13 SPIINT Interrupt Circuit

15.7 DMA Transfer Requests

The SPI has a function to generate DMA transfer requests from the causes shown in Table 15.8.

Table 15.8 DMA Transfer Request Causes of SPI

Cause to request DMA transfer	DMA transfer request flag	Set condition	Clear condition
Receive buffer full	Receive buffer full flag (SPIINTF.RBFIF)	When data of the specified bit length is received and the received data is transferred from the shift register to the received data buffer	Reading the SPIRXD register
Transmit buffer empty	Transmit buffer empty flag (SPIINTF.TBEIF)	When transmit data written to the transmit data buffer is transferred to the shift register	Writing to the SPITXD register

The SPI provides DMA transfer request enable bits corresponding to each DMA transfer request flag shown above for the number of DMA channels. A DMA transfer request is sent to the pertinent channel of the DMA controller only when the DMA transfer request flag, of which DMA transfer has been enabled by the DMA transfer request enable bit, is set. The DMA transfer request flag also serves as an interrupt flag, therefore, both the DMA transfer request and the interrupt cannot be enabled at the same time. After a DMA transfer has completed, disable the DMA transfer to prevent unintended DMA transfer requests from being issued. For more information on the DMA control, refer to the “DMA Controller” chapter.

15.8 Control Registers

See Section 10.5 for descriptions of the control registers for SPI.

16. I2C Interface

16.1 Overview

The I2C is a subset of the I2C bus interface. The features of the I2C are listed below.

- Functions as an I2C bus master (single master) or a slave device.
- Supports standard mode (up to 100 kbit/s) and fast mode (up to 400 kbit/s).
- Supports 7-bit and 10-bit address modes.
- Supports clock stretching.
- Includes a baud rate generator for generating the clock in master mode.
- No clock source is required to run the I2C in slave mode, as it can run with the I2C bus signals only and generate an interrupt when data is received.
- Master mode supports automatic bus clear sending function.
- Can generate receive buffer full, transmit buffer empty, and other interrupts.
- Can issue a DMA transfer request when a receive buffer full or a transmit buffer empty occurs.

Figure 16.1 shows the I2C configuration.

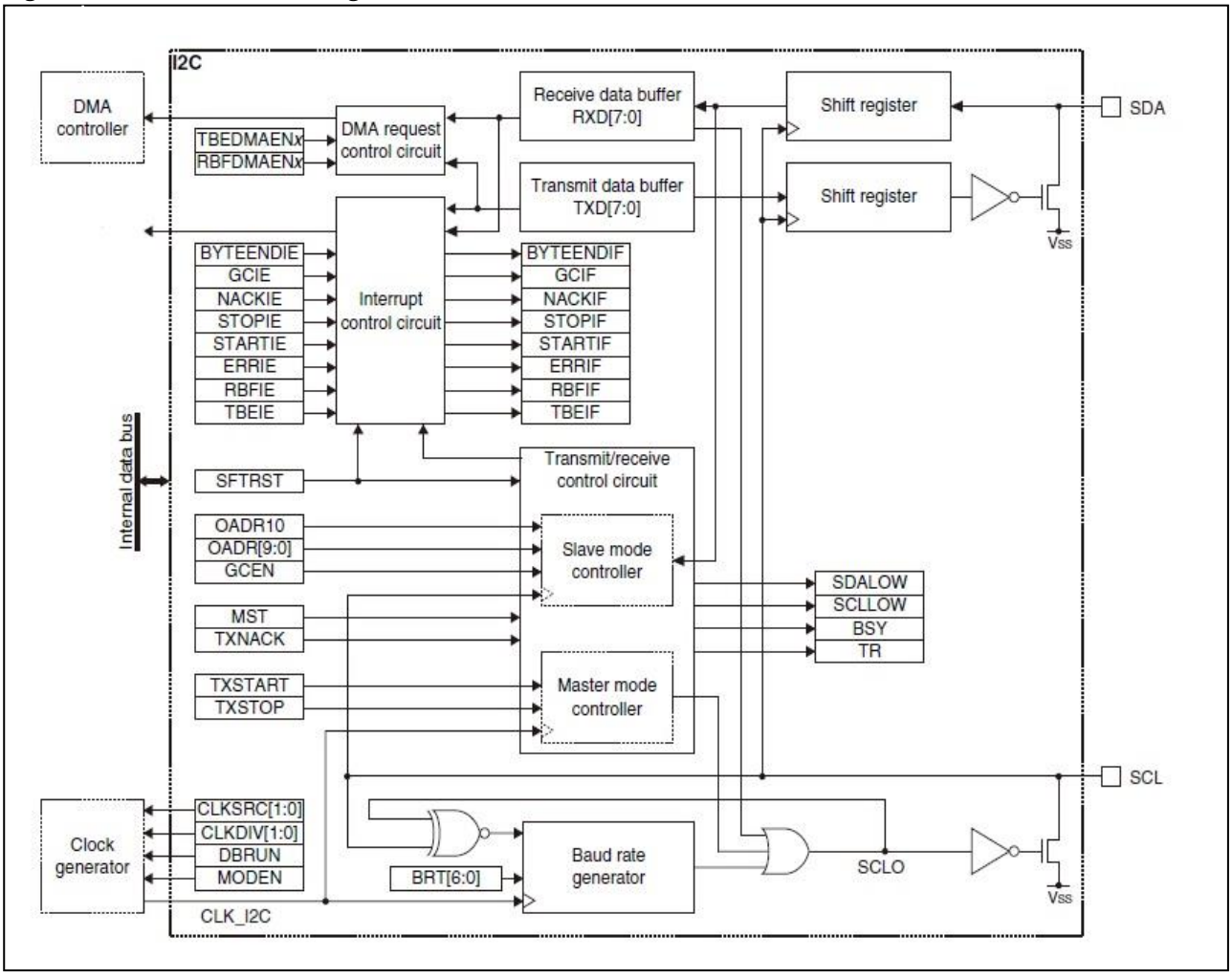


Figure 16.1 I2C Configuration

16.2 Input/Output Pins and External Connections

16.2.1 List of Input/Output Pins

Table 16.1 lists the I2C pins.

Table 16.1 List of I2C Pins

Pin name	I/O*	Initial status*	Function
SDA	I/O	I	I2C bus serial data input/output pin
SCL	I/O	I	I2C bus clock input/output pin

* Indicates the status when the pin is configured for the I2C.

If the port is shared with the I2C pin and other functions, the I2C input/output function must be assigned to the port before activating the I2C. For more information, refer to the "GPIO Ports" chapter.

16.2.2 External Connections

Figure 16.2 shows a connection diagram between the I2C in this IC and external I2C devices.
The serial data (SDA) and serial clock (SCL) lines must be pulled up with an external resistor.
When the I2C is set into master mode, one or more slave devices that have a unique address may be connected to the I2C bus. When the I2C is set into slave mode, one or more master and slave devices that have a unique address may be connected to the I2C bus.

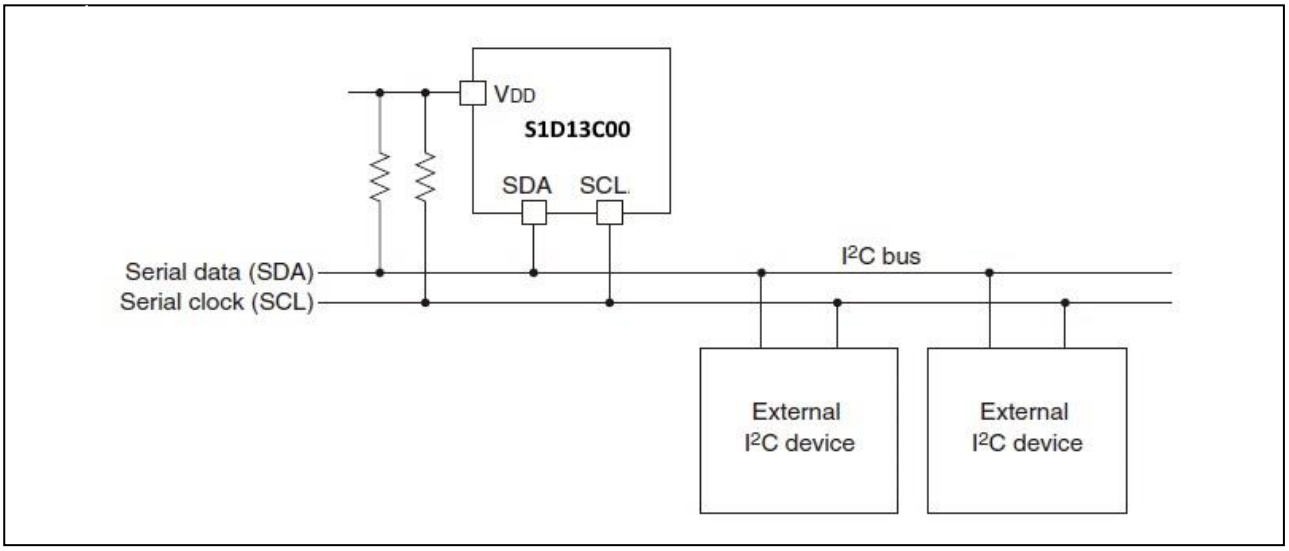


Figure 16.2 Connections between I2C and External I2C Devices

Note:

- The SDA and SCL lines must be pulled up to a VDD of this IC or lower voltage. However, if the I2C input/output ports are configured with the over voltage tolerant fail-safe type I/O, these lines can be pulled up to a voltage exceeding the VDD of this IC but within the recommended operating voltage range of this IC.
- The internal pull-up resistors for the I/O ports cannot be used for pulling up SDA and SCL.
- When the I2C is set into master mode, no other master device can be connected to the I2C bus.

16.3 Clock Settings

16.3.1 I2C Operating Clock

Master mode operating clock
When using the I2C in master mode, the I2C operating clock CLK_I2C must be supplied to the I2C from the clock generator. The CLK_I2C supply should be controlled as in the procedure shown below.

1. Enable the clock source in the clock generator if it is stopped.
2. Set the following I2CCLK register bits:
   - I2CCLK.CLKSRC[1:0] bits (Clock source selection)
   - I2CCLK.CLKDIV[1:0] bits (Clock division ratio selection = Clock frequency setting)

   The I2C operating clock should be selected so that the baud rate generator will be configured easily.

Slave mode operating clock
The I2C set to slave mode uses the SCL supplied from the I2C master as its operating clock. The clock setting by the I2CCLK register is ineffective.
The I2C keeps operating using the clock supplied from the external I2C master even if all the internal clocks are stopped, so the I2C can receive data and can generate receive buffer full interrupts.

16.3.2 Baud Rate Generator

The I2C includes a baud rate generator to generate the serial clock SCL used in master mode. The I2C set to slave mode does not use the baud rate generator, as it operates with the serial clock input from the SCL pin.

Setting data transfer rate (for master mode)
The transfer rate is determined by the I2CBR.BRT[6:0] bit settings. Use the following equations to calculate the setting values for obtaining the desired transfer rate.

$bps = f_{CLK\_I2C} / ((BRT + 3) \times 2)$
$BRT = (f_{CLK\_I2C} / (bps \times 2)) - 3$

Where

bps:	Data transfer rate [bit/s]
f_{CLK_I2C}:	I2C operating clock frequency [Hz]
BRT:	I2CBRT[6:0] bits setting value (1 to 127)

* The equations above do not include SCL rising/falling time and delay time by clock stretching.

Note: The I2C bus transfer rate is limited to 100 kbit/s in standard mode or 400 kbit/s in fast mode. Do not set a transfer rate exceeding the limit.

Baud rate generator clock output and operations for supporting clock stretching

Figure 16.3 shows the clock generated by the baud rate generator and the clock waveform on the I2C bus.

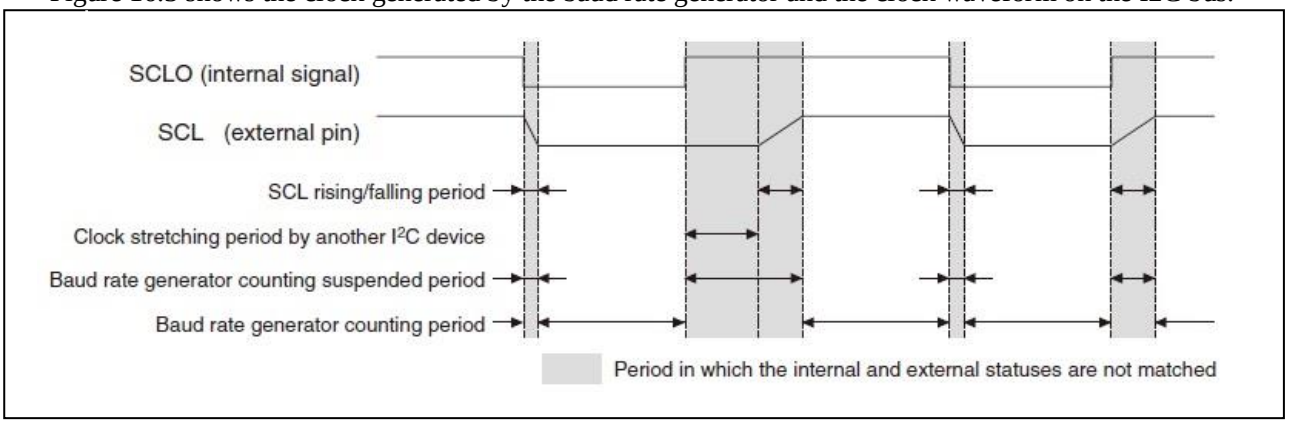


Figure 16.3 Baud Rate Generator Output Clock and SCL Output Waveform

The baud rate generator output clock SCLO is compared with the SCL pin status and the results are returned to the baud rate generator. If a mismatch has occurred between SCLO and SCL pin levels, the baud rate generator suspends counting. This extends the clock to control data transfer during the SCL signal rising/falling period and clock stretching period in which SCL is fixed at low by a slave device.

16.4 Operations

16.4.1 Initialization

The I2C should be initialized with the procedure shown below.

When using the I2C in master mode

1. Configure the operating clock and the baud rate generator using the I2CCLK and I2CBR registers.
2. Assign the I2C input/output function to the ports. (Refer to the "GPIO Ports" chapter.)
3. Set the following bits when using the interrupt:
   - Write 1 to the interrupt flags in the I2CINTF register. (Clear interrupt flags)
   - Set the interrupt enable bits in the I2CINTE register to 1. (Enable interrupts)
4. Set the following I2CCTL register bits:
   - Set the I2CCTL.MST bit to 1. (Set master mode)
   - Set the I2CCTL.SFTRST bit to 1. (Execute software reset)
   - Set the I2CCTL.MODEN bit to 1. (Enable I2C operations)

When using the I2C in slave mode

1. Set the following I2CMOD register bits:
   - I2CMOD.OADR10 bit (Set 10/7-bit address mode)
   - I2CMOD.GCEN bit (Enable response to general call address)
2. Set its own address to the I2COADR.OADR[9:0] (or OADR[6:0]) bits.
3. Assign the I2C Ch.n input/output function to the ports. (Refer to the "GPIO Ports" chapter.)
4. Set the following bits when using the interrupt:
   - Write 1 to the interrupt flags in the I2CINTF register. (Clear interrupt flags)
   - Set the interrupt enable bits in the I2CINTE register to 1. (Enable interrupts)
5. Set the following I2C_nCTL register bits:
   - Set the I2CCTL.MST bit to 0. (Set slave mode)
   - Set the I2CCTL.SFTRST bit to 1. (Execute software reset)
   - Set the I2CCTL.MODEN bit to 1. (Enable I2C operations)

16.4.2 Data Transmission in Master Mode

A data sending procedure in master mode and the I2C operations are shown below. Figure 16.4 and Figure 16.5 show an operation example and a flowchart, respectively.

Data sending procedure

1. Issue a START condition by setting the I2CCTL.TXSTART bit to 1.
2. Wait for a transmit buffer empty interrupt (I2CINTF.TBEIF bit = 1) or a START condition interrupt (I2CINTF.STARTIF bit = 1).
   Clear the I2CINTF.STARTIF bit by writing 1 after the interrupt has occurred.
3. Write the 7-bit slave address to the I2CTXD.TXD[7:1] bits and 0 that represents WRITE as the data transfer direction to the I2CTXD.TXD0 bit.
4. (When DMA is used) Configure the DMA controller and set a DMA transfer request enable bit in the I2CTBEDMAEN register to 1 (DMA transfer request enabled). (This automates the data sending procedure Steps 5, 6, and 8.)
5. (When DMA is not used) Wait for a transmit buffer empty interrupt (I2CINTF.TBEIF bit = 1) generated when an ACK is received.
6. (When DMA is not used) Write transmit data to the I2CTXD register.
7. If a NACK reception interrupt (I2CINTF.NACKIF bit = 1) has occurred, go to Step 9 or 1 after clearing the I2CINTF.NACKIF bit.
8. (When DMA is not used) Repeat Steps 5 and 6 until the end of transmit data.
9. Issue a STOP condition by setting the I2CCTL.TXSTOP bit to 1.
10. Wait for a STOP condition interrupt (I2CINTF.STOPIF bit = 1).
    Clear the I2CINTF.STOPIF bit by writing 1 after the interrupt has occurred.

Data sending operations

Generating a START condition

The I2C starts generating a START condition when the I2CCTL.TXSTART bit is set to 1. When the generating operation has completed, the I2C clears the I2CCTL.TXSTART bit to 0 and sets both the I2CINTF.STARTIF and I2CINTF.TBEIF bits to 1.

Sending slave address and data

If the I2CINTF.TBEIF bit = 1, a slave address or data can be written to the I2CTXD register. The I2C pulls down SCL to low and enters standby state until data is written to the I2CTXD register. The writing operation triggers the I2C to send the data to the shift register automatically and to output eight clock pulses and data bits to the I2C bus.
When the slave device returns an ACK as the response, the I2CINTF.TBEIF bit is set to 1. After this interrupt occurs, the subsequent data may be sent or a STOP/repeated START condition may be issued to terminate transmission. If the slave device returns NACK, the I2CINTF.NACKIF bit is set to 1 without setting the I2CINTF.TBEIF bit.

Generating a STOP/repeated START condition

After the I2CINTF.TBEIF bit is set to 1 (transmit buffer empty) or the I2CINTF.NACKIF bit is set to 1 (NACK received), setting the I2CCTL.TXSTOP bit to 1 generates a STOP condition. When the bus free time (tBUF defined in the I2C Specifications) has elapsed after the STOP condition has been generated, the I2CCTL.TXSTOP bit is cleared to 0 and the I2CINTF.STOPIF bit is set to 1.
When setting the I2CCTL.TXSTART bit to 1 while the I2CINTF.TBEIF bit = 1 (transmit buffer empty) or the I2CINTF.NACKIF bit = 1 (NACK received), the I2C generates a repeated START condition.
When the repeated START condition has been generated, the I2CINTF.STARTIF and I2CINTF.TBEIF bits are both set to 1 same as when a START condition has been generated.

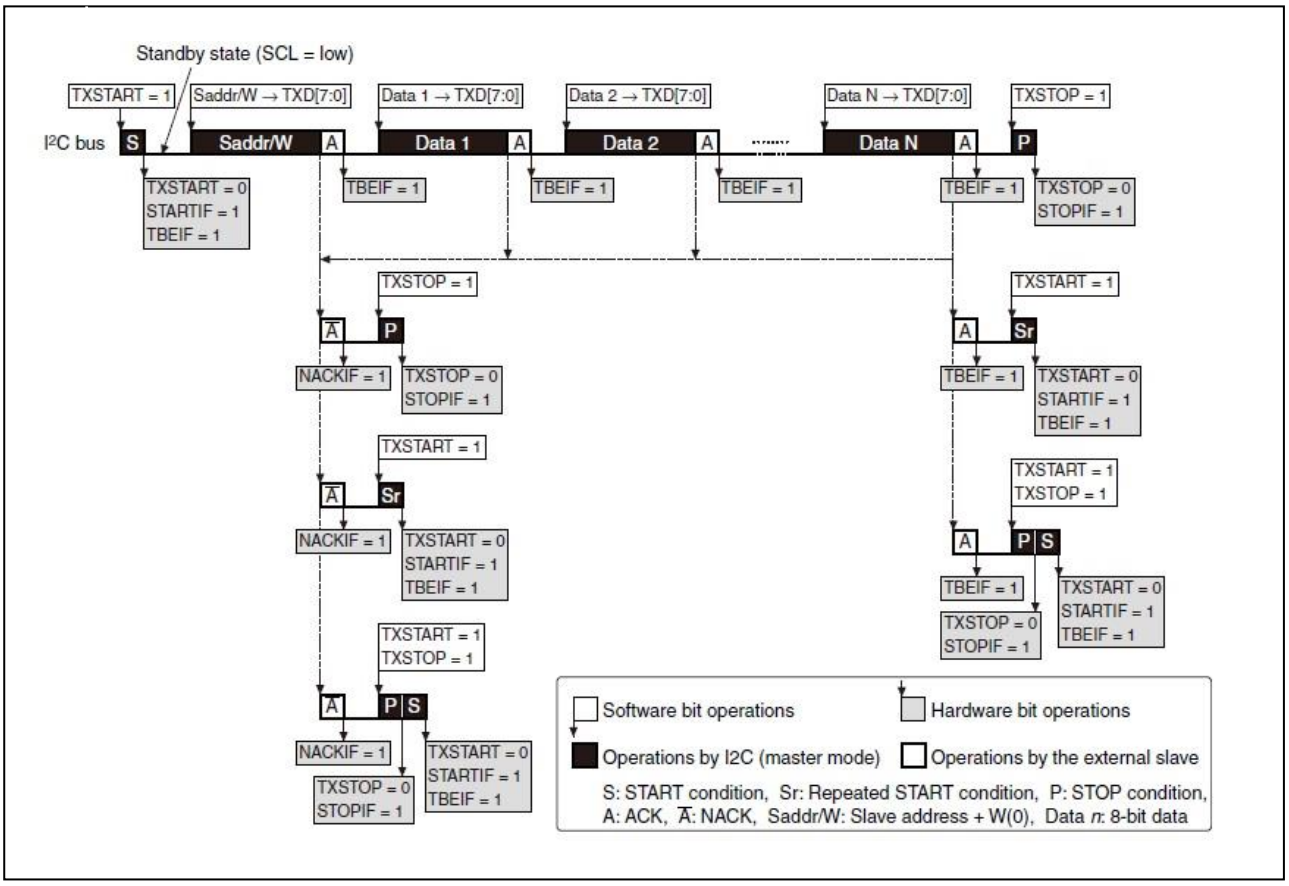


Figure 16.4 Example of Data Sending Operations in Master Mode

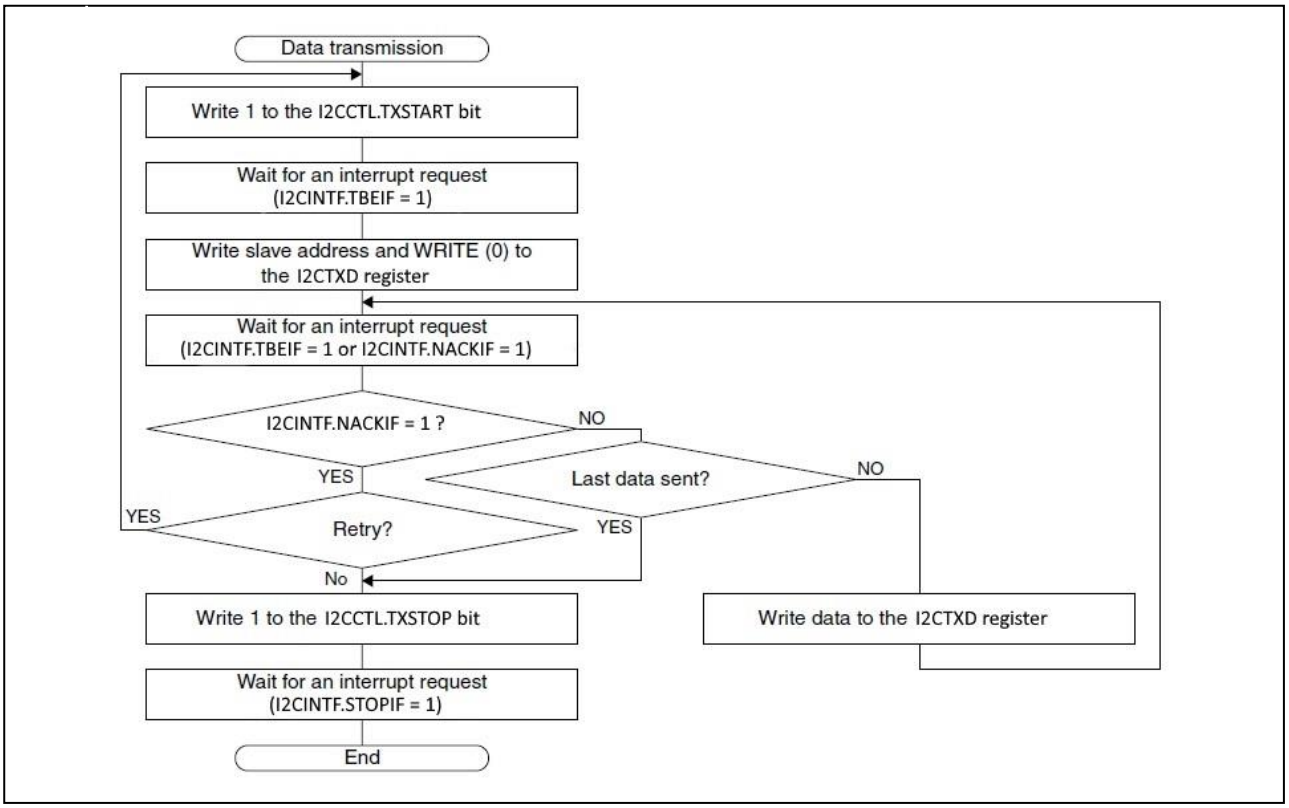


Figure 16.5 Master Mode Data Transmission Flowchart

Data transmission using DMA

By setting the I2CTBEDMAEN.TBEDMAENx bit to 1 (DMA transfer request enabled), a DMA transfer request is sent to the DMA controller and transmit data is transferred from the specified memory to the I2CTXD register via DMA Ch.x when the I2CINTF.TBEIF bit is set to 1 (transmit buffer empty).
This automates the data sending procedure from Steps 5, 6, and 8 described above.
The transfer source/destination and control data must be set for the DMA controller and the relevant DMA channel must be enabled to start a DMA transfer in advance so that transmit data will be transferred to the I2CTXD register. For more information on DMA, refer to the "DMA Controller" chapter.

Table 16.2 DMA Data Structure Configuration Example (for Data Transmission)

Item		Setting example
End pointer	Transfer source	Memory address in which the last transmit data is stored
	Transfer destination	I2CTXD register address
Control data	dst_inc	0x3 (no increment)
	dst_size	0x0 (byte)
	src_inc	0x0 (+1)
	src_size	0x0 (byte)
	R_power	0x0 (arbitrated for every transfer)
	n_minus_1	Number of transfer data
	cycle_ctrl	0x1 (basic transfer)

16.4.3 Data Reception in Master Mode

A data receiving procedure in master mode and the I2C operations are shown below. Figure 16.6 and Figure 16.7 show an operation example and a flowchart, respectively.

Data receiving procedure

1. Issue a START condition by setting the I2CCTL.TXSTART bit to 1.
2. Wait for a transmit buffer empty interrupt (I2C_nINTF.TBEIF bit = 1) or a START condition interrupt (I2CINTF.STARTIF bit = 1).
   Clear the I2CINTF.STARTIF bit by writing 1 after the interrupt has occurred.
3. Write the 7-bit slave address to the I2CTXD.TXD[7:1] bits and 1 that represents READ as the data transfer direction to the I2CTXD.TXD0 bit.
4. (When DMA is used) Configure the DMA controller and set a DMA transfer request enable bit in the I2CRBFDMAEN register to 1 (DMA transfer request enabled). (This automates the data receiving procedure Steps 5, 7, and 9.)
5. (When DMA is not used) Wait for a receive buffer full interrupt (I2CINTF.RBFIF bit = 1) generated when a one-byte reception has completed.
6. Perform one of the operations below when the last or next-to-last data is received.
   i. When the next-to-last data is received, write 1 to the I2CCTL.TXNACK bit to send a NACK after the last data is received, and then go to Step 7.
   ii. When the last data is received, read the received data from the I2CRXD register and set the I2CCTL.TXSTOP to 1 to generate a STOP condition. Then go to Step 10.
7. (When DMA is not used) Read the received data from the I2CRXD register.
8. If a NACK reception interrupt (I2CINTF.NACKIF bit = 1) has occurred, clear the I2CINTF.NACKIF bit and issue a STOP condition by setting the I2CCTL.TXSTOP bit to 1. Then go to Step 10 or Step 1 if making a retry.
9. (When DMA is not used) Repeat Steps 5 to 7 until the end of data reception.
10. Wait for a STOP condition interrupt (I2CINTF.STOPIF bit = 1).
    Clear the I2CINTF.STOPIF bit by writing 1 after the interrupt has occurred.

Data receiving operations

Generating a START condition

It is the same as the data transmission in master mode.

Sending slave address

It is the same as the data transmission in master mode. Note, however, that the I2CTXD.TXD0 bit must be set to 1 that represents READ as the data transfer direction to issue a request to the slave to send data.

Receiving data

After the slave address has been sent, the slave device sends an ACK and the first data. The I2C sets the I2CINTF.RBFIF bit to 1 after the data reception has completed. Furthermore, the I2C returns an ACK. To return a NACK, such as for a response after the last data has been received, write 1 to the I2CCTL.TXNACK bit before the I2CINTF.RBFIF bit is set to 1.

The received data can be read out from the I2CRXD register after a receive buffer full interrupt has occurred. The I2C pulls down SCL to low and enters standby state until data is read out from the I2CRXD register.

This reading triggers the I2C to start subsequent data reception.

Generating a STOP or repeated START condition

It is the same as the data transmission in master mode.

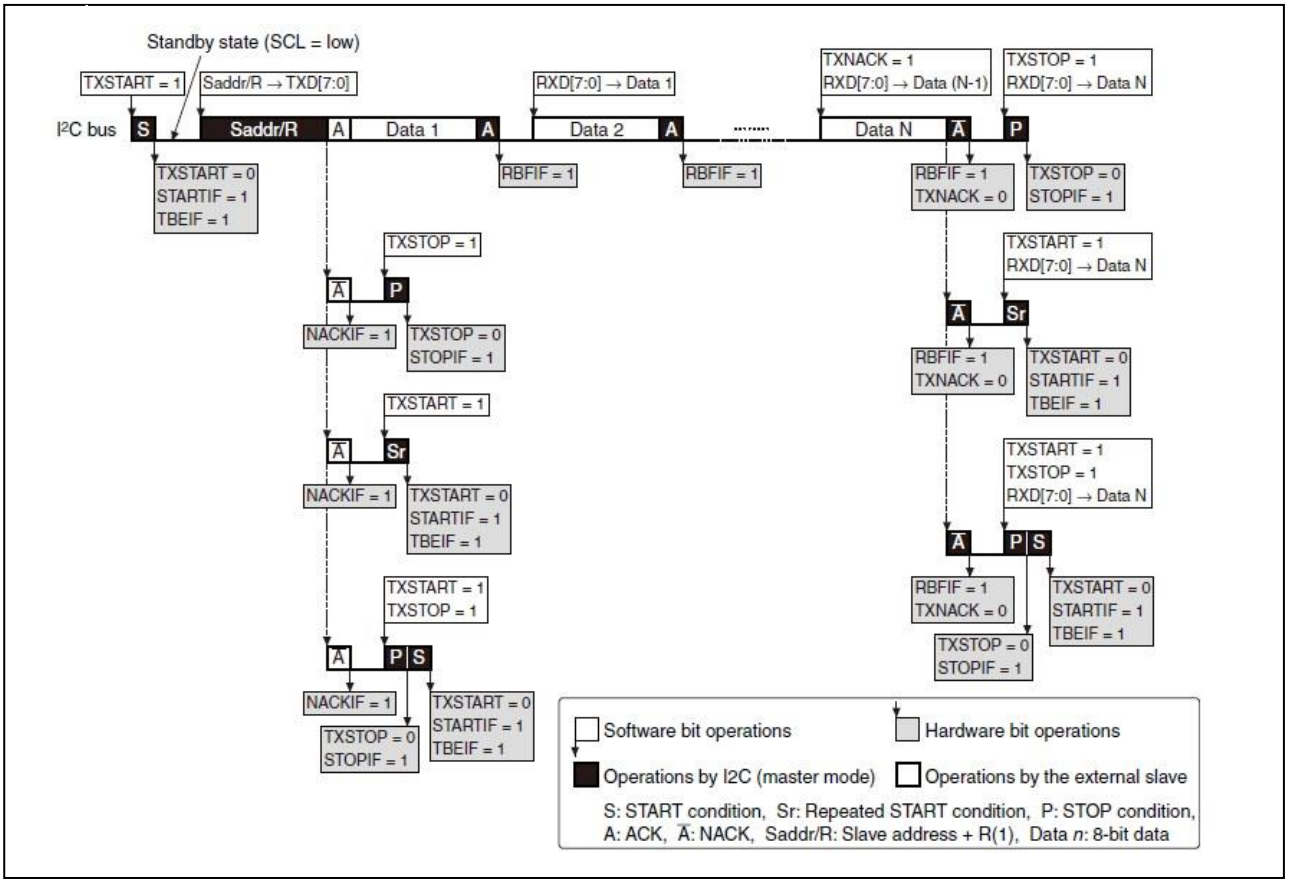


Figure 16.6 Example of Data Receiving Operations in Master Mode

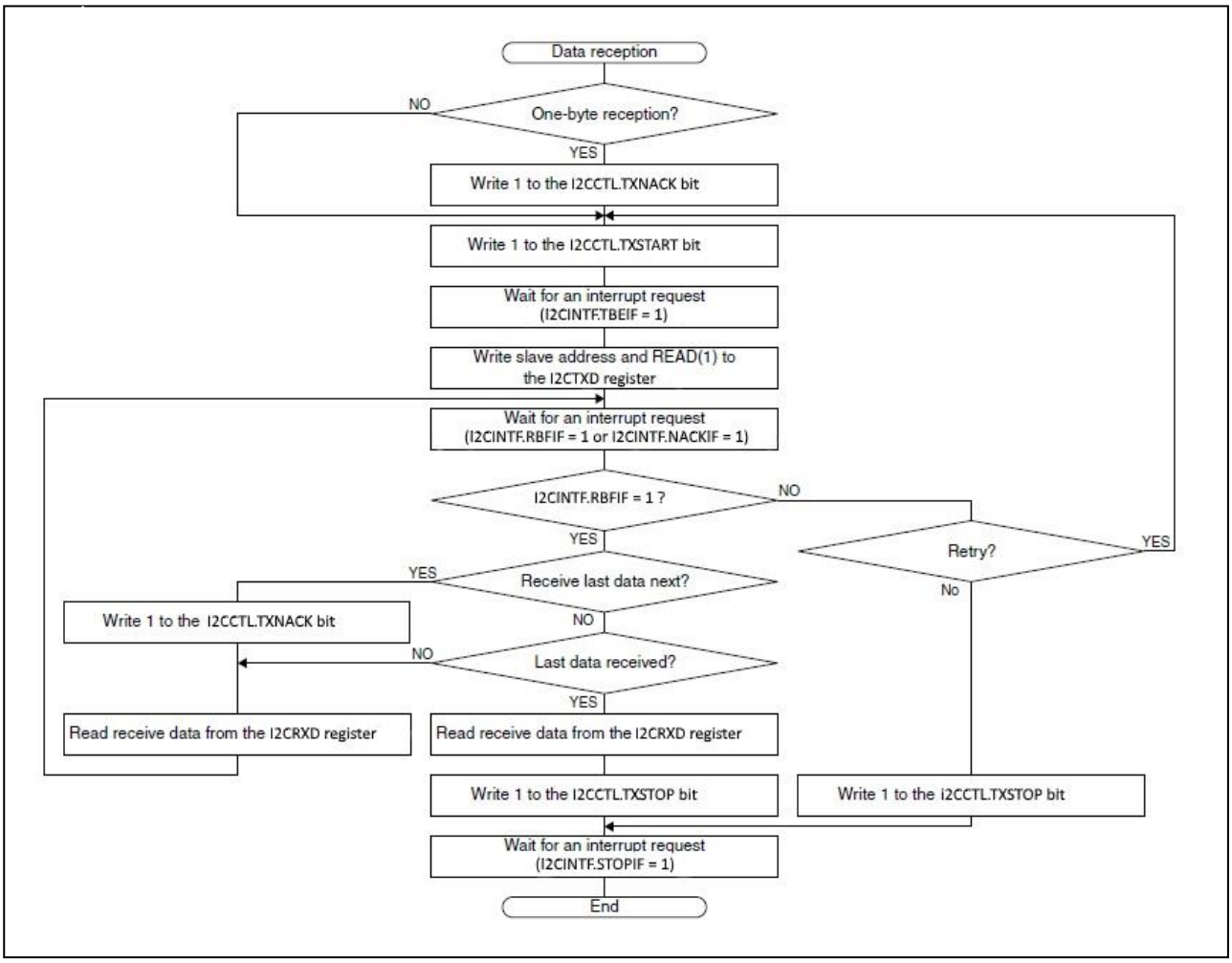


Figure 16.7 Master Mode Data Reception Flowchart

Data reception using DMA

By setting the I2CRBFDMAEN.RBFDMAENx bit to 1 (DMA transfer request enabled), a DMA transfer request is sent to the DMA controller and the received data is transferred from the I2CRXD register to the specified memory via DMA Ch.x when the I2CINTF.RBFIF bit is set to 1 (receive buffer full).
This automates the data receiving procedure Steps 5, 7, and 9 described above.
The transfer source/destination and control data must be set for the DMA controller and the relevant DMA channel must be enabled to start a DMA transfer in advance. For more information on DMA, refer to the "DMA Controller" chapter.

Table 16.3 DMA Data Structure Configuration Example (for Data Reception)

Item		Setting example
End pointer	Transfer source	I2CRXD register address
	Transfer destination	Memory address to which the last received data is stored
Control data	dst_inc	0x0 (+1)
	dst_size	0x0 (byte)
	src_inc	0x3 (no increment)
	src_size	0x0 (byte)
	R_power	0x0 (arbitrated for every transfer)
	n_minus_1	Number of transfer data
	cycle_ctrl	0x1 (basic transfer)

16.4.4 10-bit Addressing in Master Mode

A 10-bit address consists of the first address that contains two high-order bits and the second address that contains eight low-order bits.

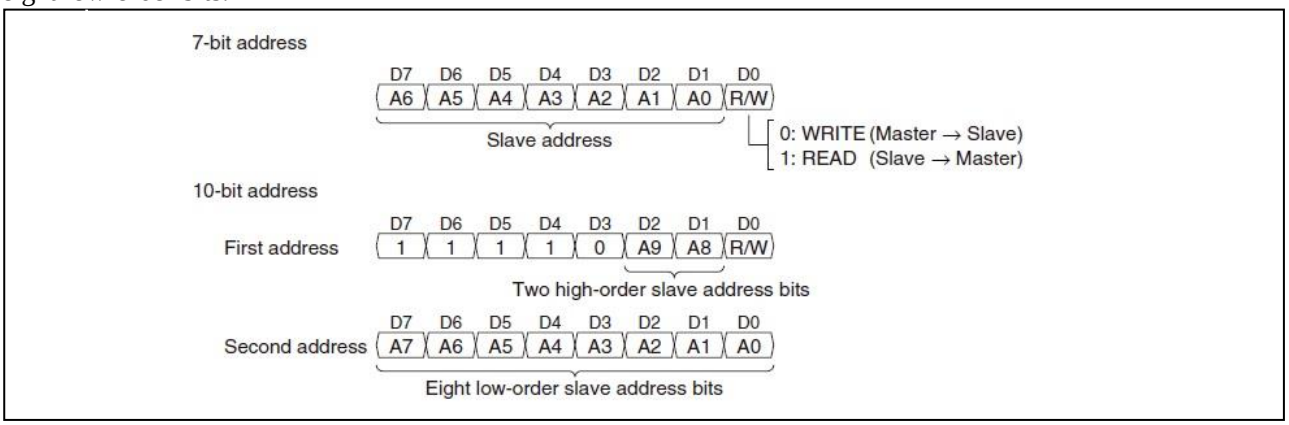


Figure 16.8 10-bit Address Configuration

The following shows a procedure to start data transfer in 10-bit address mode when the I2C is placed into master mode (see the 7-bit mode descriptions above for control procedures when a NACK is received or sending/ receiving data). Figure 16.4.4.2 shows an operation example.

Starting data transmission in 10-bit address mode

1. Issue a START condition by setting the I2CCTL.TXSTART bit to 1.
2. Wait for a transmit buffer empty interrupt (I2CINTF.TBEIF bit = 1) or a START condition interrupt (I2CINTF.STARTIF bit = 1).
   Clear the I2CINTF.STARTIF bit by writing 1 after the interrupt has occurred.
3. Write the first address to the I2CTXD.TXD[7:1] bits and 0 that represents WRITE as the data transfer direction to the I2CTXD.TXD0 bit.
4. Wait for a transmit buffer empty interrupt (I2CINTF.TBEIF bit = 1).
5. Write the second address to the I2CTXD.TXD[7:0] bits.
6. Wait for a transmit buffer empty interrupt (I2CINTF.TBEIF bit = 1).
7. Perform data transmission.

Starting data reception in 10-bit address mode

1 to 6. These steps are the same as the data transmission starting procedure described above.

7. Issue a repeated START condition by setting the I2CCTL.TXSTART bit to 1.
8. Wait for a transmit buffer empty interrupt (I2CINTF.TBEIF bit = 1) or a START condition interrupt (I2CINTF.STARTIF bit = 1).
   Clear the I2CINTF.STARTIF bit by writing 1 after the interrupt has occurred.
9. Write the first address to the I2CTXD.TXD[7:1] bits and 1 that represents READ as the data transfer direction to the I2CTXD.TXD0 bit.
10. Perform data reception.

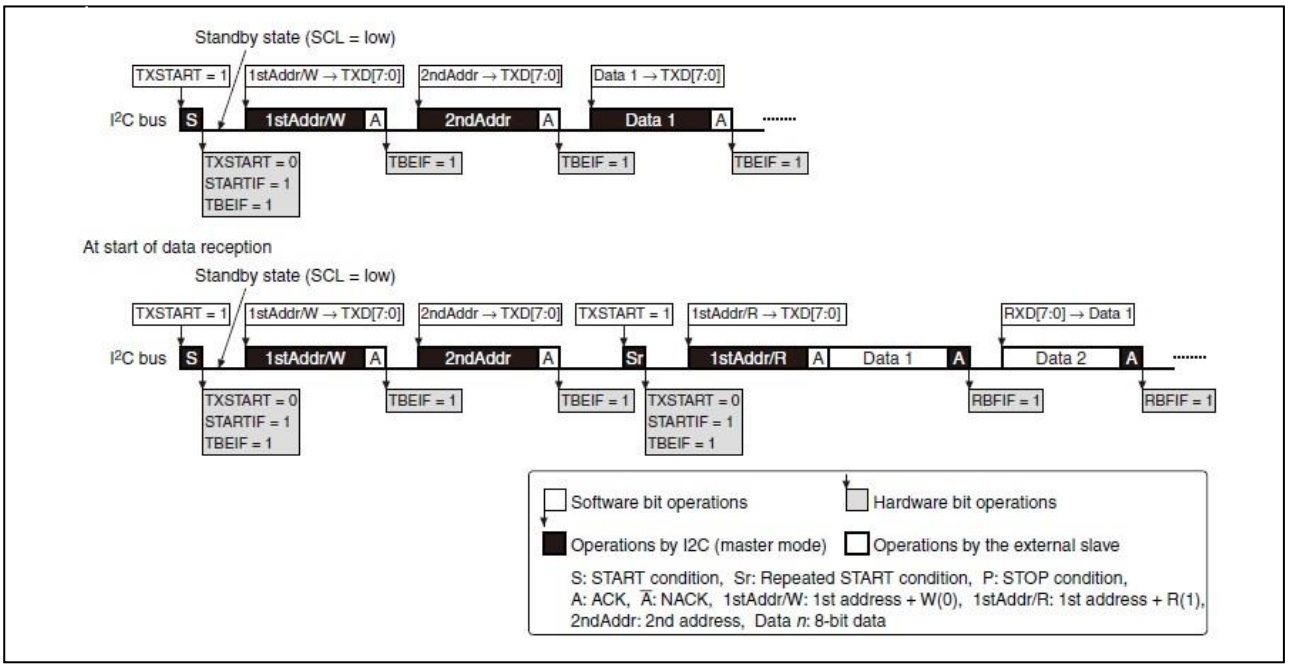


Figure 16.9 Example of Data Transfer Starting Operations in 10-bit Address Mode (Master Mode)

16.4.5 Data Transmission in Slave Mode

A data sending procedure in slave mode and the I2C operations are shown below. Figure 16.10 and Figure 16.11 show an operation example and a flowchart, respectively.

Data sending procedure

1. Wait for a START condition interrupt (I2CINTF.STARTIF bit = 1).
   Clear the I2CINTF.STARTIF bit by writing 1 after the interrupt has occurred.
2. Check to see if the I2CINTF.TR bit = 1 (transmission mode).
   (Start a data receiving procedure if the I2CINTF.TR bit = 0.)
3. Write transmit data to the I2CTXD register.
4. Wait for a transmit buffer empty interrupt (I2CINTF.TBEIF bit = 1), a NACK reception interrupt (I2CINTF.NACKIF bit = 1), or a STOP condition interrupt (I2CINTF.STOPIF bit = 1).
   i. Go to Step 3 when a transmit buffer empty interrupt has occurred.
   ii. Go to Step 5 after clearing the I2CINTF.NACKIF bit when a NACK reception interrupt has occurred.
   iii. Go to Step 6 when a STOP condition interrupt has occurred.
5. Wait for a STOP condition interrupt (I2CINTF.STOPIF bit = 1) or a START condition interrupt (I2CINTF.STARTIF bit = 1).
   i. Go to Step 6 when a STOP condition interrupt has occurred.
   ii. Go to Step 2 when a START condition interrupt has occurred.
6. Clear the I2CINTF.STOPIF bit and then terminate data sending operations.

Data sending operations

START condition detection and slave address check

While the I2CCTL.MODEN bit = 1 and the I2CCTL.MST bit = 0 (slave mode), the I2C monitors the I2C bus. When the I2C detects a START condition, it starts receiving of the slave address sent from the master. If the received address is matched with the own address set to the I2COADR.OADR[6:0] bits (when the I2CMOD.OADR10 bit = 0 (7-bit address mode)) or the I2COADR.OADR[9:0] bits (when the I2CMOD.OADR10 bit = 1 (10-bit address mode)), the I2CINTF.STARTIF bit and the I2CINTF.BSY bit are both set to 1. The I2C sets the I2CINTF.TR bit to the R/W bit value in the received address. If this value is 1, the I2C sets the I2CINTF.TBEIF bit to 1 and starts data sending operations.

Sending the first data byte

After the valid slave address has been received, the I2C pulls down SCL to low and enters standby state until data is written to the I2CTXD register. This puts the I2C bus into clock stretching state and the external master into standby state. When transmit data is written to the I2CTXD register, the I2C clears the I2CINTF.TBEIF bit and sends an ACK to the master. The transmit data written in the I2CTXD register is automatically transferred to the shift register and the I2CINTF.TBEIF bit is set to 1. The data bits in the shift register are output in sequence to the I2C bus.

Sending subsequent data

If the I2CINTF.TBEIF bit = 1, subsequent transmit data can be written during data transmission. If the I2CINTF.TBEIF bit is still set to 1 when the data transmission from the shift register has completed, the I2C pulls down SCL to low (sets the I2C bus into clock stretching state) until transmit data is written to the I2CTXD register.

If the next transmit data already exists in the I2CTXD register or data has been written after the above, the I2C sends the subsequent eight-bit data when an ACK from the external master is received. At the same time, the I2CINTF.BYTEENDIF bit is set to 1. If a NACK is received, the I2CINTF.NACKIF bit is set to 1 without sending data.

STOP/repeated START condition detection

While the I2CCTL.MST bit = 0 (slave mode) and the I2CINTF.BSY = 1, the I2C monitors the I2C bus. When the I2C detects a STOP condition, it terminates data sending operations. At this time, the I2CINTF.BSY bit is cleared to 0 and the I2CINTF.STOPIF bit is set to 1. Also when the I2C detects a repeated START condition, it terminates data sending operations. In this case, the I2CINTF.STARTIF bit is set to 1.

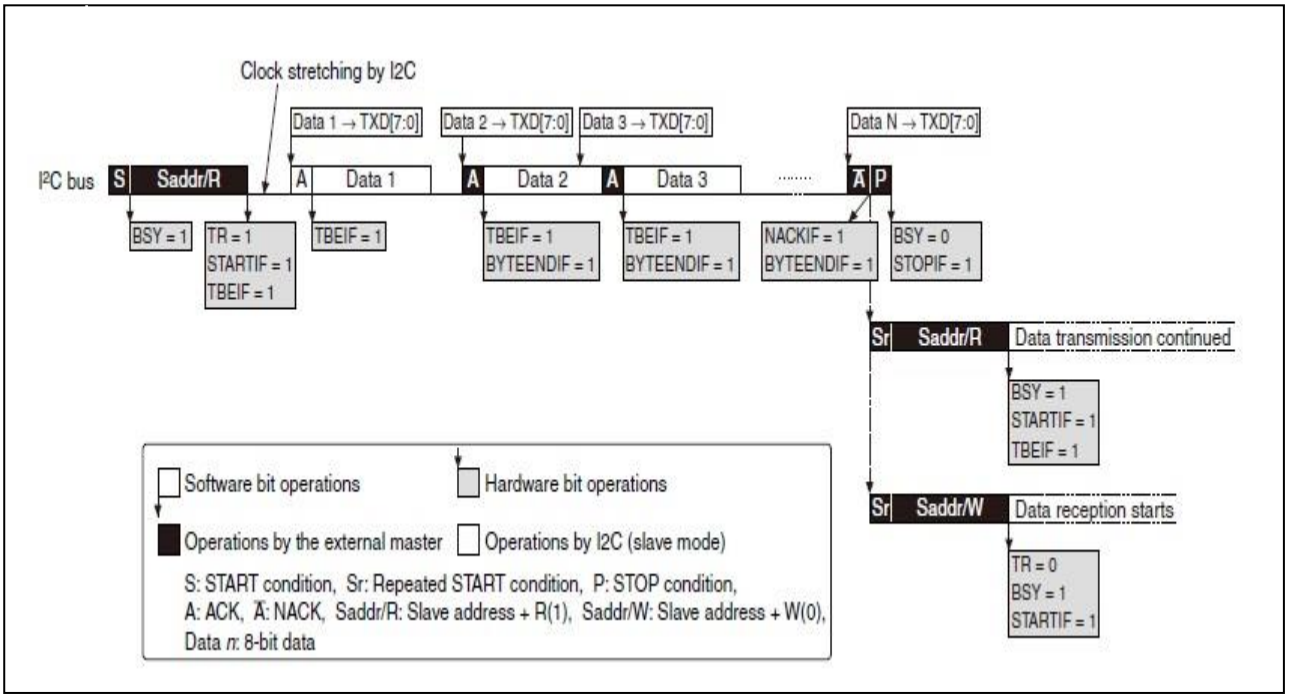


Figure 16.10 Example of Data Sending Operations in Slave Mode

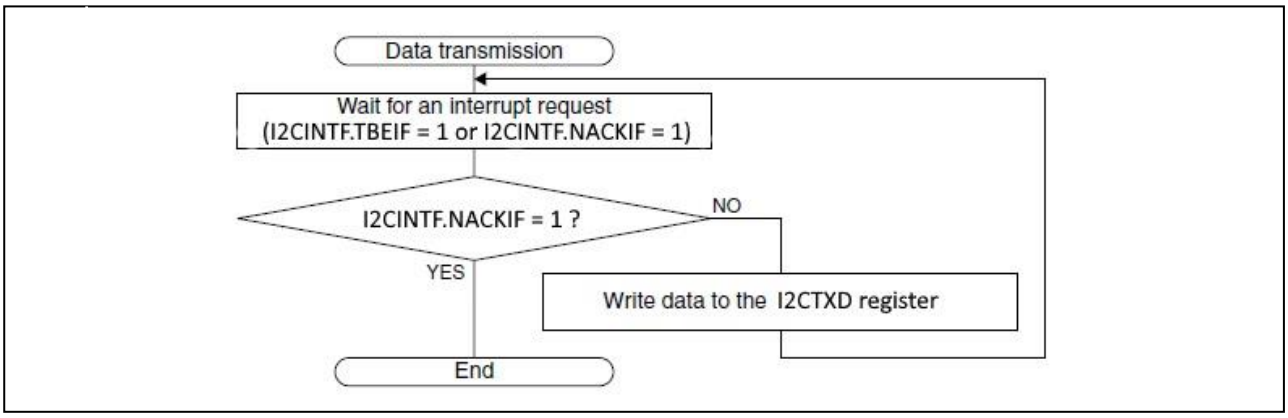


Figure 16.11 Slave Mode Data Transmission Flowchart

16.4.6 Data Reception in Slave Mode

A data receiving procedure in slave mode and the I2C operations are shown below. Figure 16.12 and Figure 16.13 show an operation example and a flowchart, respectively.

Data receiving procedure

1. When receiving one-byte data, write 1 to the I2CCTL.TXNACK bit.
2. Wait for a START condition interrupt (I2CINTF.STARTIF bit = 1).
3. Check to see if the I2CINTF.TR bit = 0 (reception mode).
   (Start a data sending procedure if I2CINTF.TR bit = 1.)
4. Clear the I2CINTF.STARTIF bit by writing 1.
5. Wait for a receive buffer full interrupt (I2CINTF.RBFIF bit = 1) generated when a one-byte reception has completed or an end of transfer interrupt (I2CINTF.BYTEENDIF bit = 1).
   Clear the I2CINTF.BYTEENDIF bit by writing 1 after the interrupt has occurred.
6. If the next receive data is the last one, write 1 to the I2CCTL.TXNACK bit to send a NACK after it is received.
7. Read the received data from the I2CRXD register.
8. Repeat Steps 5 to 7 until the end of data reception.
9. Wait for a STOP condition interrupt (I2CINTF.STOPIF bit = 1) or a START condition interrupt (I2CINTF.STARTIF bit = 1).
   i. Go to Step 10 when a STOP condition interrupt has occurred.
   ii. Go to Step 3 when a START condition interrupt has occurred.
10. Clear the I2CINTF.STOPIF bit and then terminate data receiving operations.

Data receiving operations

START condition detection and slave address check

It is the same as the data transmission in slave mode.
However, the I2CINTF.TR bit is cleared to 0 and the I2CINTF.TBEIF bit is not set.
If the I2CMOD.GCEN bit is set to 1 (general call address response enabled), the I2C starts data receiving operations when the general call address is received.

Receiving the first data byte

After the valid slave address has been received, the I2C sends an ACK and pulls down SCL to low until 1 is written to the I2CINTF.STARTIF bit. This puts the I2C bus into clock stretching state and the external master into standby state. When 1 is written to the I2CINTF.STARTIF bit, the I2C releases SCL and receives data sent from the external master into the shift register. After eight-bit data has been received, the I2C sends an ACK and pulls down SCL to low. The received data in the shift register is transferred to the receive data buffer and the I2CINTF.RBFIF and I2CINTF.BYTEENDIF bits are both set to 1. After that, the received data can be read out from the I2CRXD register.

Receiving subsequent data

When the received data is read out from the I2CRXD register after the I2CINTF.RBFIF bit has been set to 1, the I2C clears the I2CINTF.RBFIF bit to 0, releases SCL, and receives subsequent data sent from the external master. After eight-bit data has been received, the I2C sends an ACK and pulls down SCL to low. The received data in the shift register is transferred to the receive data buffer and the I2CINTF.RBFIF and I2CINTF.BYTEENDIF bits are both set to 1.
To return a NACK after eight-bit data is received, such as when terminating data reception, write 1 to the I2CCTL.TXNACK bit before the data reception is completed. The I2CCTL.TXNACK bit is automatically cleared to 0 after a NACK has been sent.

STOP/repeated START condition detection

It is the same as the data transmission in slave mode.

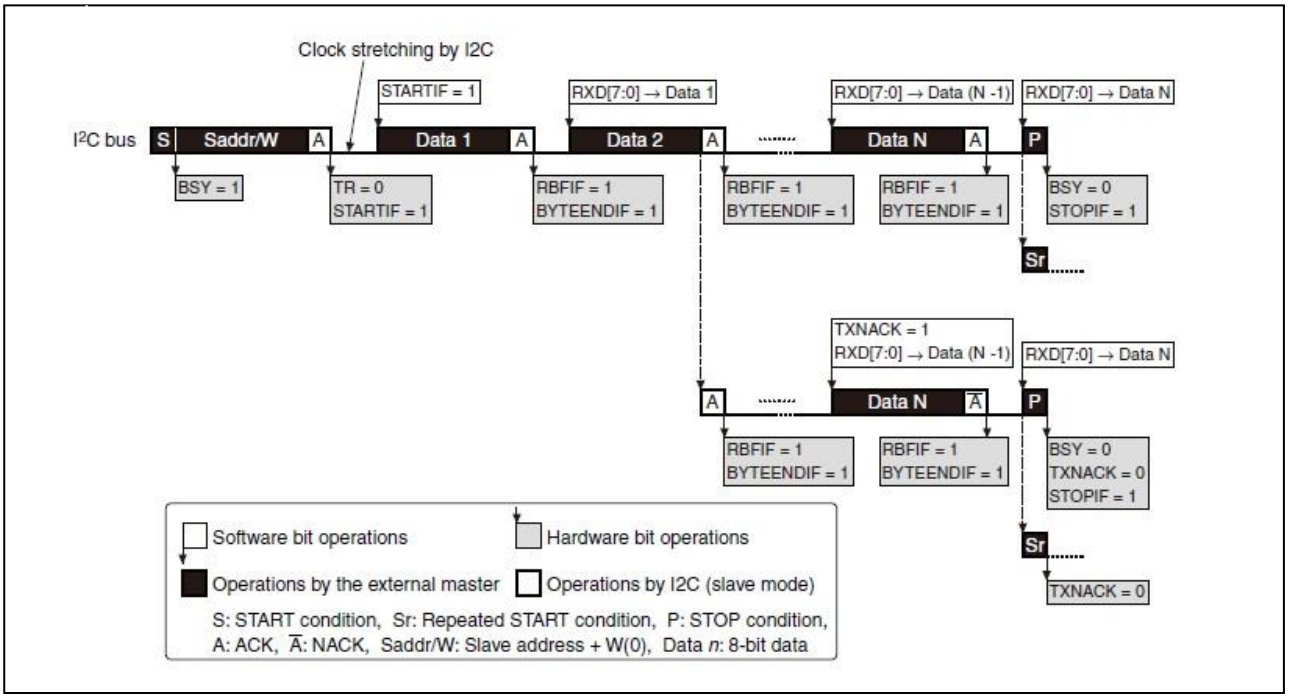


Figure 16.12 Example of Data Receiving Operations in Slave Mode

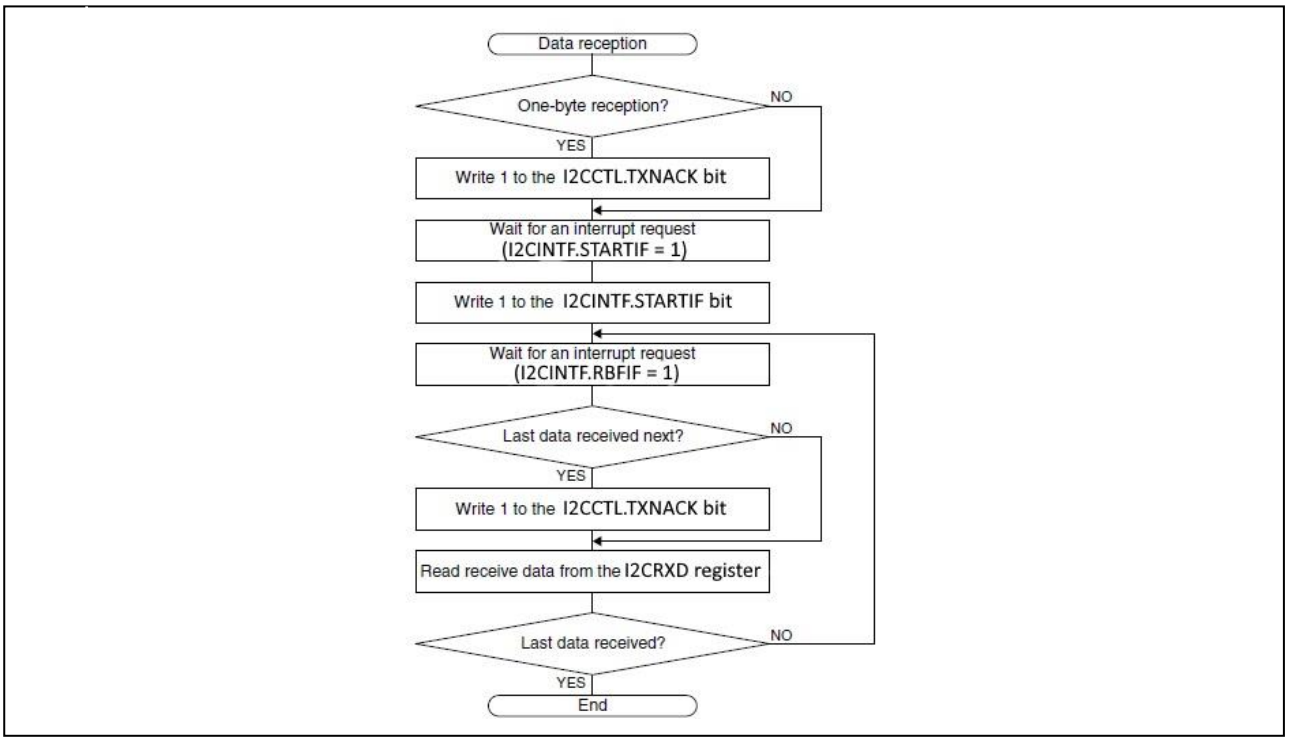


Figure 16.13 Slave Mode Data Reception Flowchart

16.4.7 Slave Operations in 10-bit Address Mode

The I2C functions as a slave device in 10-bit address mode when the I2CCTL.MST bit = 0 and the I2CMOD.OADR10 bit = 1.
The following shows the address receiving operations in 10-bit address mode. Figure 16.14 shows an operation example. See Figure 16.4.4.1 for the 10-bit address configuration.

10-bit address receiving operations

After a START condition is issued, the master sends the first address that includes the two high-order slave address bits and the R/W bit (= 0). If the received two high-order slave address bits are matched with the I2COADR.OADR[9:8] bits, the I2C returns an ACK. At this time, other slaves may return an ACK as the two high-order bits may be matched.
Then the master sends the eight low-order slave address bits as the second address. If this address is matched with the I2COADR.OADR[7:0] bits, the I2C returns an ACK and starts data receiving operations.
If the master issues a request to the slave to send data (data reception in the master), the master generates a repeated START condition and sends the first address with the R/W bit set to 1. This reception switches the I2C to data sending mode.

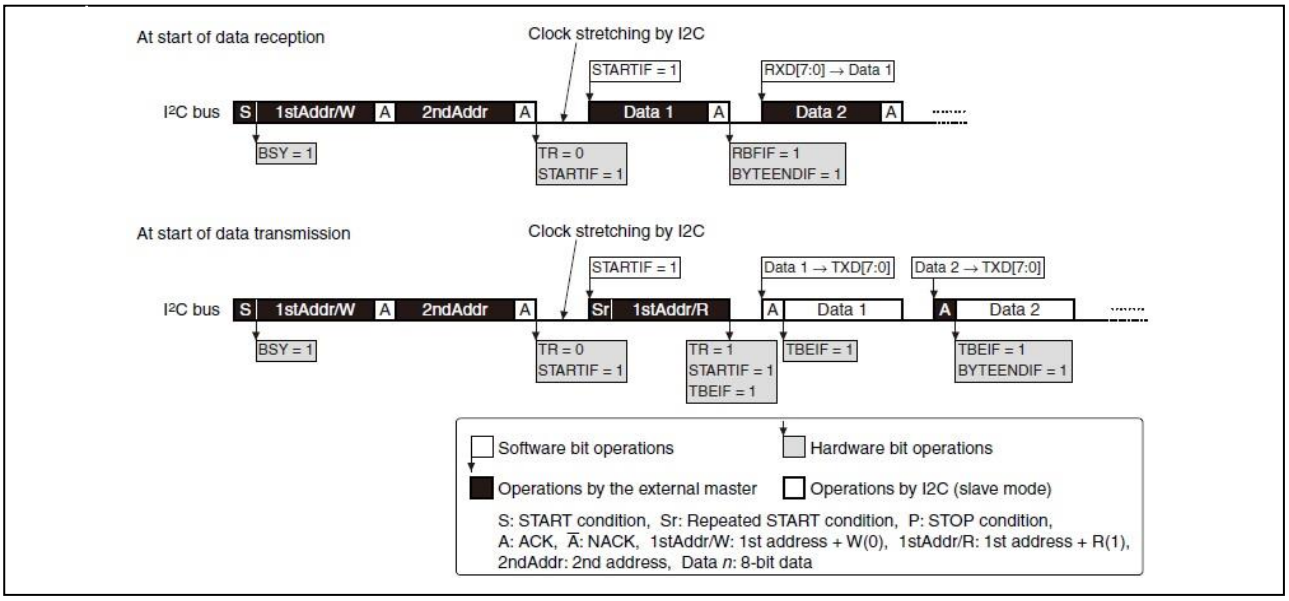


Figure 16.14 Example of Data Transfer Starting Operations in 10-bit Address Mode (Slave Mode)

16.4.8 Automatic Bus Clearing Operation

The I2C set into master mode checks the SDA state immediately before generating a START condition. If SDA is set to a low level at this time, the I2C automatically executes bus clearing operations that output up to ten clocks from the SCL pin with SDA left free state.
When SDA goes high from low within nine clocks, the I2C issues a START condition and starts normal operations. If SDA does not change from low when the I2C outputs the ninth clock, it is regarded as an automatic bus clearing failure. In this case, the I2C clears the I2CCTL.TXSTART bit to 0 and sets both the I2CINTF.ERRIF and I2CINTF.STARTIF bits to 1.

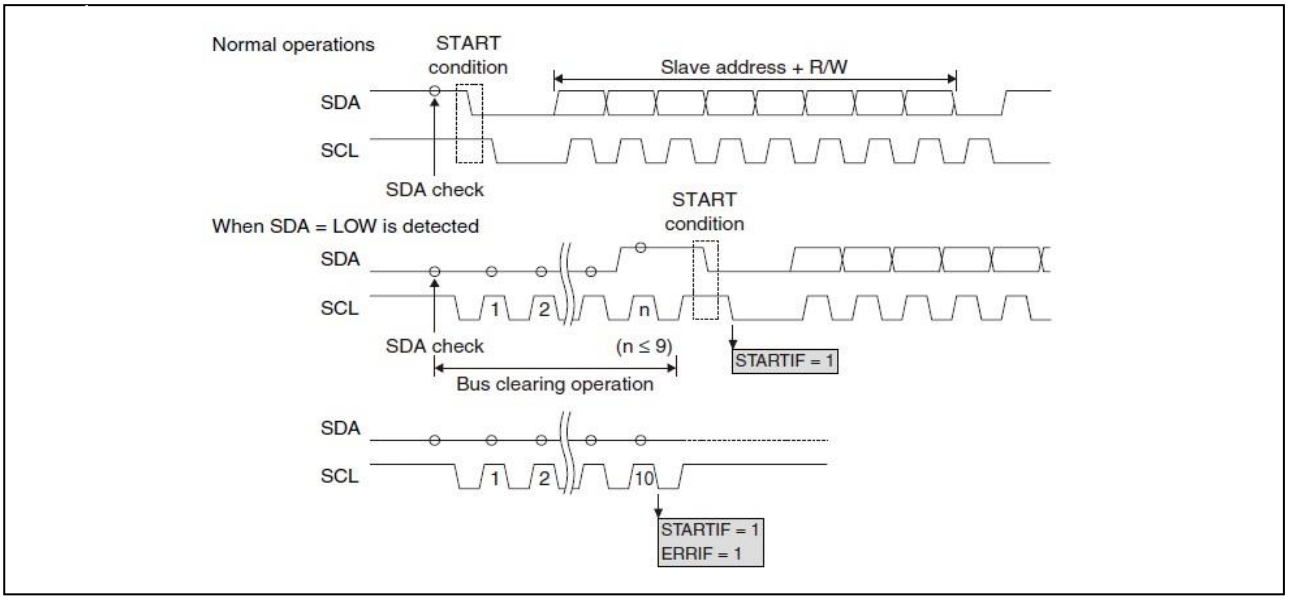


Figure 16.15 Automatic Bus Clearing Operation

16.4.9 Error Detection

The I2C includes a hardware error detection function.
Furthermore, the I2CINTF.SDALOW and I2CINTF.SCLLOW bits are provided to allow software to check whether the SDA and SCL lines are fixed at low. If unintended low level is detected on SDA or SCL, a software recovery processing, such as I2C software reset, can be performed.
The table below lists the hardware error detection conditions and the notification method.

Table 16.4 Hardware Error Detection Function

No.	Error detecting period/timing	I2C bus line monitored and error condition	Notification method
1	While the I2C controls SDA to high for sending address, data, or a NACK	SDA = low	I2CINTF.ERRIF = 1
2	<Master mode only> When 1 is written to the I2CCTL.TXSTART bit while the I2CINTF.BSY bit = 0	SCL = low	I2CINTF.ERRIF = 1 I2CCTL.TXSTART = 0 I2CINTF.STARTIF = 1
3	<Master mode only> When 1 is written to the I2CCTL.TXSTOP bit while the I2CINTF.BSY bit = 0	SCL = low	I2CINTF.ERRIF = 1 I2CCTL.TXSTOP = 0 I2CINTF.STOPIF = 1
4	<Master mode only> When 1 is written to the I2CCTL.TXSTART bit while the I2CINTF.BSY bit = 0 (Refer to “Automatic Bus Clearing Operation.”)	SDA Automatic bus clearing failure	I2CINTF.ERRIF = 1 I2CCTL.TXSTART = 0 I2CINTF.STARTIF = 1

16.5 Interrupts

The I2C has a function to generate the interrupts shown in Table 16.5.

Table 16.5 I2C Interrupt Function

Interrupt	Interrupt flag	Set condition	Clear condition
End of data transfer	I2CINTF.BYTEENDIF	When eight-bit data transfer and the following ACK/NACK transfer are completed	Writing 1, software reset
General call address reception	I2CINTF.GCIF	Slave mode only: When the general call address is received	Writing 1, software reset
NACK reception	I2CINTF.NACKIF	When a NACK is received	Writing 1, software reset
STOP condition	I2CINTF.STOPIF	Master mode: When a STOP condition is generated and the bus free time (tBUF) between STOP and START conditions has elapsed Slave mode: When a STOP condition is detected while the I2C is selected as the slave currently accessed	Writing 1, software reset
START condition	I2CINTF.STARTIF	Master mode: When a START condition is issued Slave mode: When an address match is detected (including general call)	Writing 1, software reset
Error detection	I2CINTF.ERRIF	Refer to “Error Detection”	Writing 1, software reset
Receive buffer full	I2CINTF.RBFIF	When received data is loaded to the receive data buffer	Reading received data (to empty the receive data buffer), software reset
Transmit buffer empty	I2CINTF.TBEIF	Master mode: When a START condition is issued or when an ACK is received from the slave Slave mode: When transmit data written to the transmit data buffer is transferred to the shift register or when an address match is detected with R/W bit set to 1	Writing transmit data

The I2C provides interrupt enable bits corresponding to each interrupt flag. An interrupt request is sent to the host MCU only when the interrupt flag, of which interrupt has been enabled by the interrupt enable bit, is set.

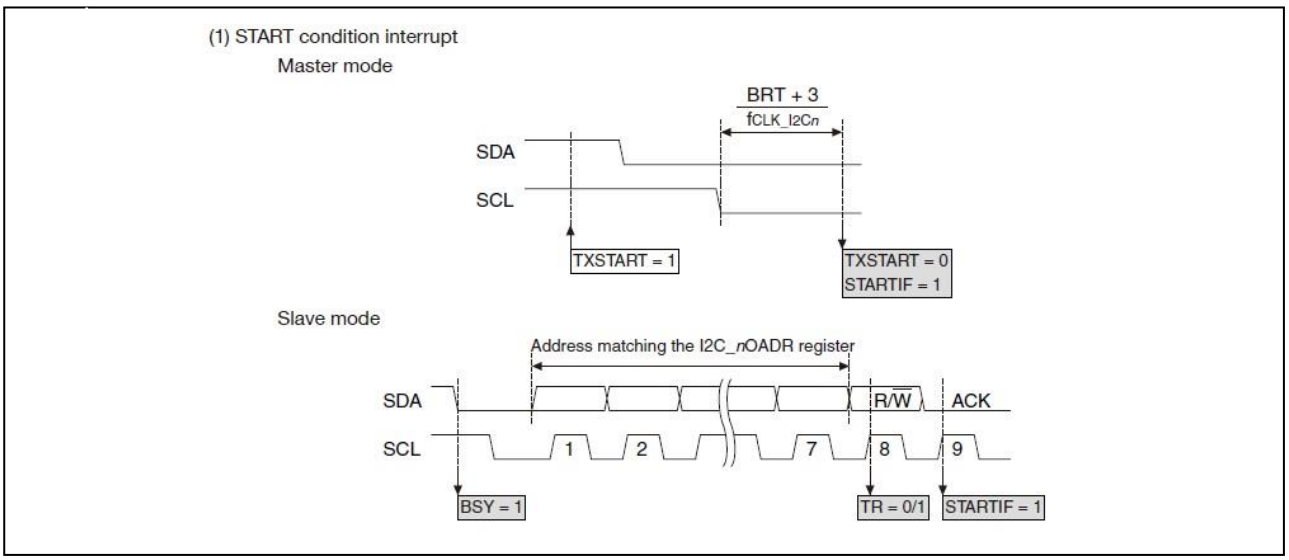


Figure 16.16 START Condition Interrupt Timings

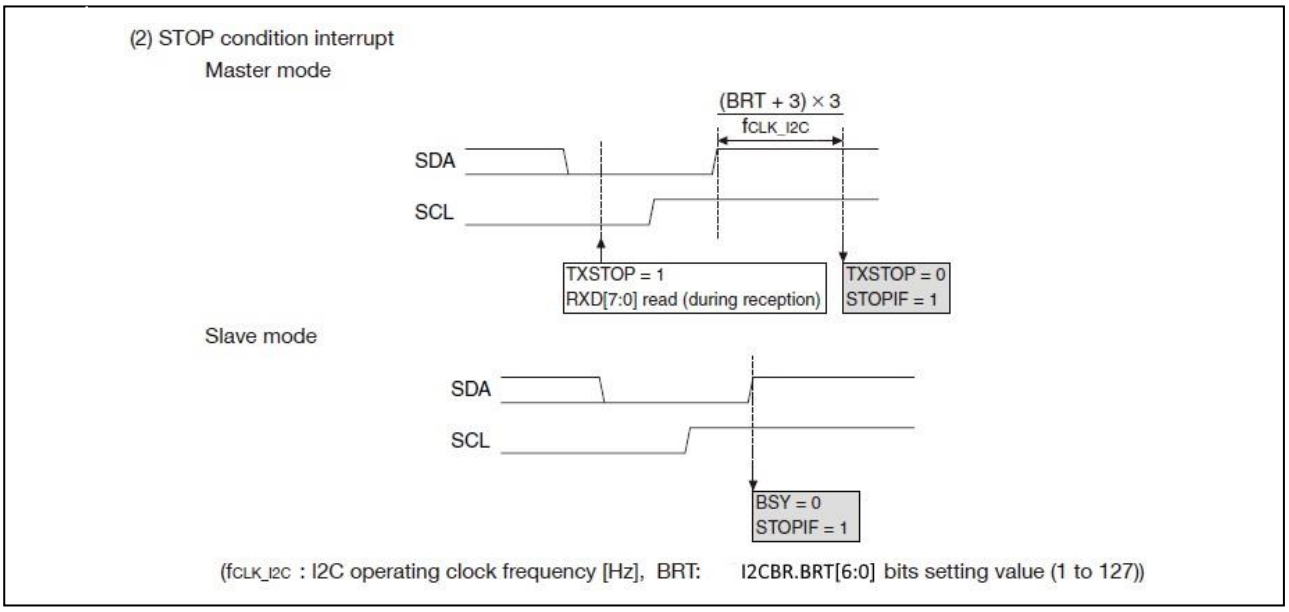


Figure 16.17 STOP Condition Interrupt Timings

Figure 16.18 shows a diagram of the I2CINT interrupt signal. The I2CINT signal is logical ORed with interrupt signals from other peripherals to generate the HIFIRQ interrupt signal to the host MCU.
See Section 22 ("Interrupts") for more details on the HIFIRQ output.

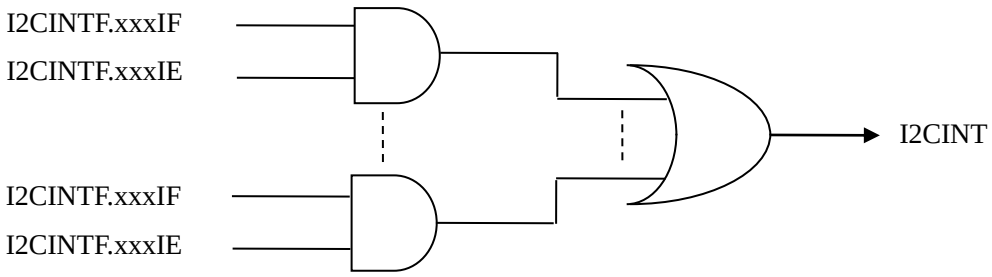


Figure 16.18 I2CINT Interrupt Circuit

16.6 DMA Transfer Requests

The I2C has a function to generate DMA transfer requests from the causes shown in Table 16.6.

Table 16.6 DMA Transfer Request Causes of I2C

Cause to request DMA transfer	DMA transfer request flag	Set condition	Clear condition
Receive buffer full	Receive buffer full flag (I2CINTF.RBFIF)	When received data is loaded to the receive data buffer	Reading received data (to empty the receive data buffer), software reset
Transmit buffer empty	Transmit buffer empty flag (I2CINTF.TBEIF)	Master mode: When a START condition is issued or when an ACK is received from the slave Slave mode: When transmit data written to the transmit data buffer is transferred to the shift register or when an address match is detected with R/W bit set to 1	Writing transmit data

The I2C provides DMA transfer request enable bits corresponding to each DMA transfer request flag shown above for the number of DMA channels. A DMA transfer request is sent to the pertinent channel of the DMA controller only when the DMA transfer request flag, of which DMA transfer has been enabled by the DMA transfer request enable bit, is set. The DMA transfer request flag also serves as an interrupt flag, therefore, both the DMA transfer request and the interrupt cannot be enabled at the same time. After a DMA transfer has completed, disable the DMA transfer to prevent unintended DMA transfer requests from being issued. For more information on the DMA control, refer to the "DMA Controller" chapter.

16.7 Control Registers

See Section 10.6 for descriptions of the control registers for I2C.

17. QSPI Interface

17.1 Overview

The QSPI is a quad synchronous serial interface. The features of the QSPI are listed below.

- Supports both master and slave modes.
- Supports single, dual, and quad transfer modes.
- Data length: 2 to 16 clocks programmable.
- Data line drive length: 1 to 16 clocks programmable (for output direction only).
- Either MSB first or LSB first can be selected for the data format.
- Clock phase and polarity are configurable.
- Supports full-duplex communications.
- Includes separated transmit data buffer and receive data buffer registers.
- Can generate receive buffer full, transmit buffer empty, end of transmission, and overrun interrupts.
- Master mode allows use of a 16-bit timer to set baud rate.
- Slave mode is capable of being operated with the external input clock QSPICLK only.
- Input pins can be pulled up/down with an internal resistor.
- Low CPU overhead memory mapped access mode that can access the external Flash memory with XIP (eXecute-In-Place) mode in the same manner as the embedded system memory.
  - Memory mapped access size: 8, 16, and 32-bit access.
  - 1M-byte external Flash memory mapping area starting at 0x0008_0000 that allows programmable re-mapping.
  - Configurable 3 or 4-byte address cycle length.
  - Single, dual, or quad transfer mode is configurable for each address, mode byte/dummy, and data cycle.
  - Programmable mode bytes for both XIP mode activation and termination.
  - Configurable mode byte/dummy output cycle length.
- Can issue a DMA transfer request when a receive buffer full, a transmit buffer empty, or a memory mapped access (32-bit read) occurs.

Figure 17.1 shows the configuration of the QSPI module. It uses either the clock source or the CLK_T16 output of the T16 CH2 timer as its clock source.

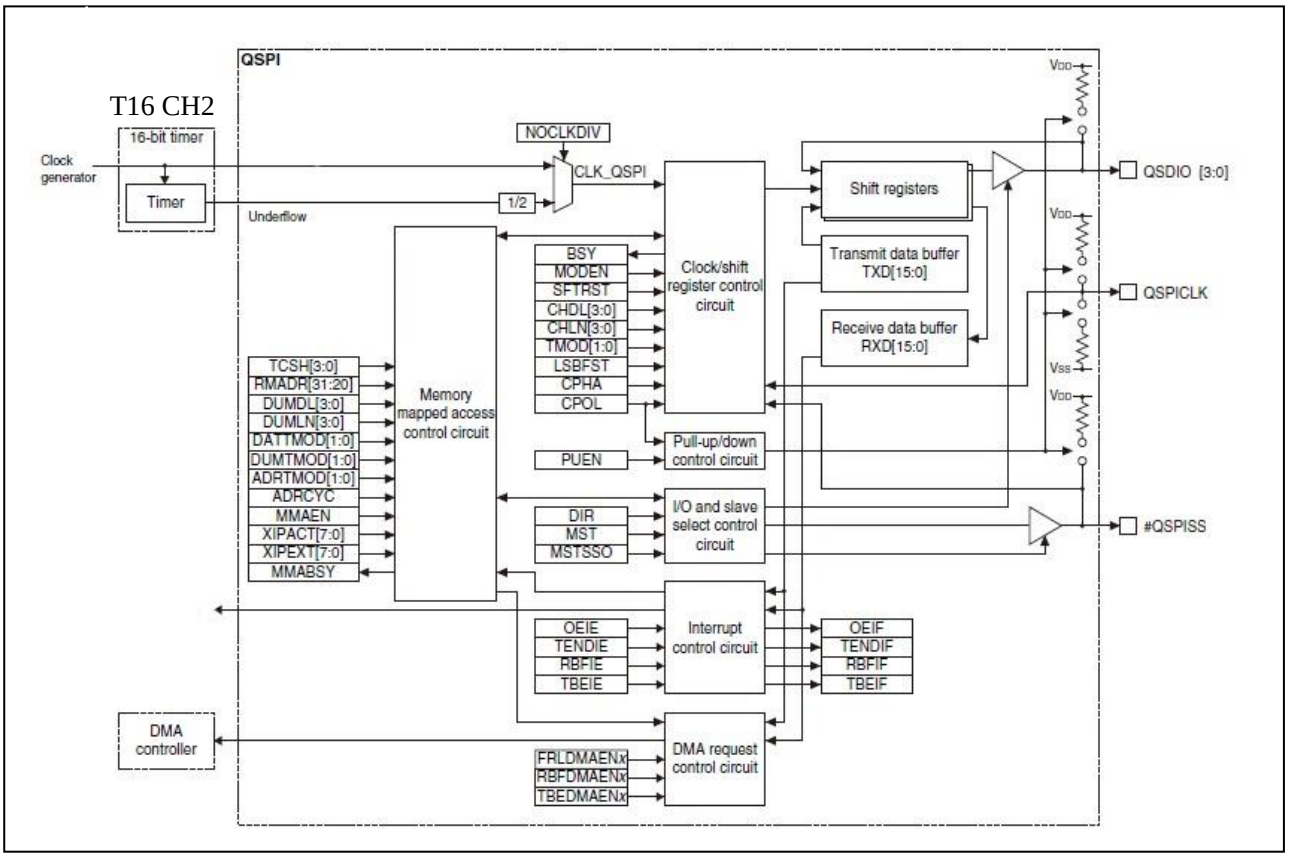


Figure 17.1 QSPI Configuration

17.2 Input/Output Pins and External Connections

17.2.1 List of Input/Output Pins

Table 17.1 lists the QSPI pins.

Table 17.1 List of QSPI Pins

Pin name	I/O*	Initial status*	Function
QSDIO[3:0]	I or O	I (Hi-Z)	QSPI data input/output pins
QSPICLK	I or O	I (Hi-Z)	QSPI external clock input/output pin
#QSPISS	I or O	I (Hi-Z)	QSPI slave select signal input/output pin

* Indicates the status when the pin is configured for the QSPI.

If the port is shared with the QSPI pin and other functions, the QSPI input/output function must be assigned to the port before activating QSPI. For more information, refer to the “GPIO Ports” chapter.

17.2.2 External Connections

The QSPI operates in master or slave mode. The memory mapped access mode is available only in master mode. When QSPI is operating in memory mapped access mode, the #QSPISS output is controlled by the internal state machine. In this case, only one external QSPI device can be connected.
When QSPI is operating in register access master mode, the #QSPISS output is directly controlled by a register bit. In this case, GPIO pins other than #QSPISS can also be used as the slave select output ports to connect the QSPI to more than one external QSPI device.
to show connection diagrams between the QSPI in each mode and external QSPI devices.

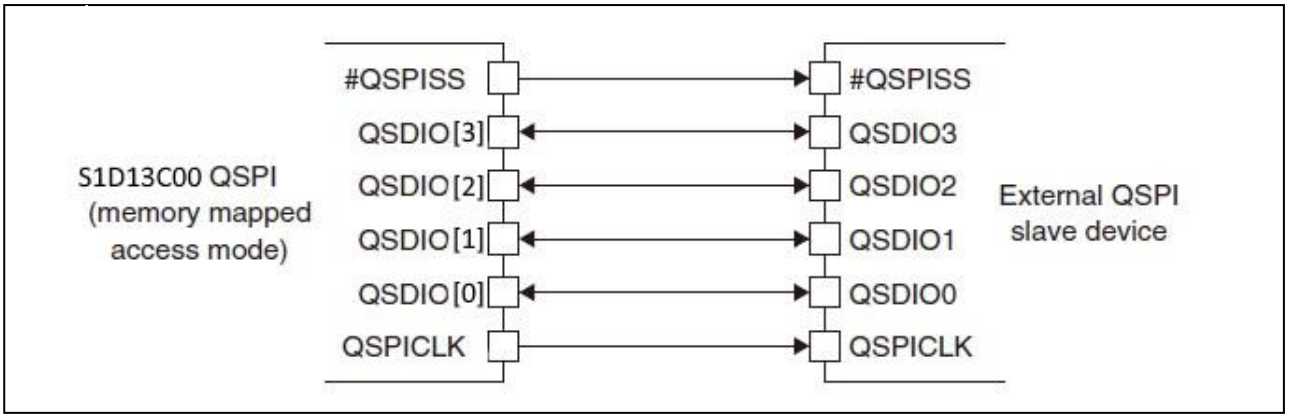


Figure 17.2 Connections between QSPI in Memory Mapped Access Mode and an External QSPI Slave Device

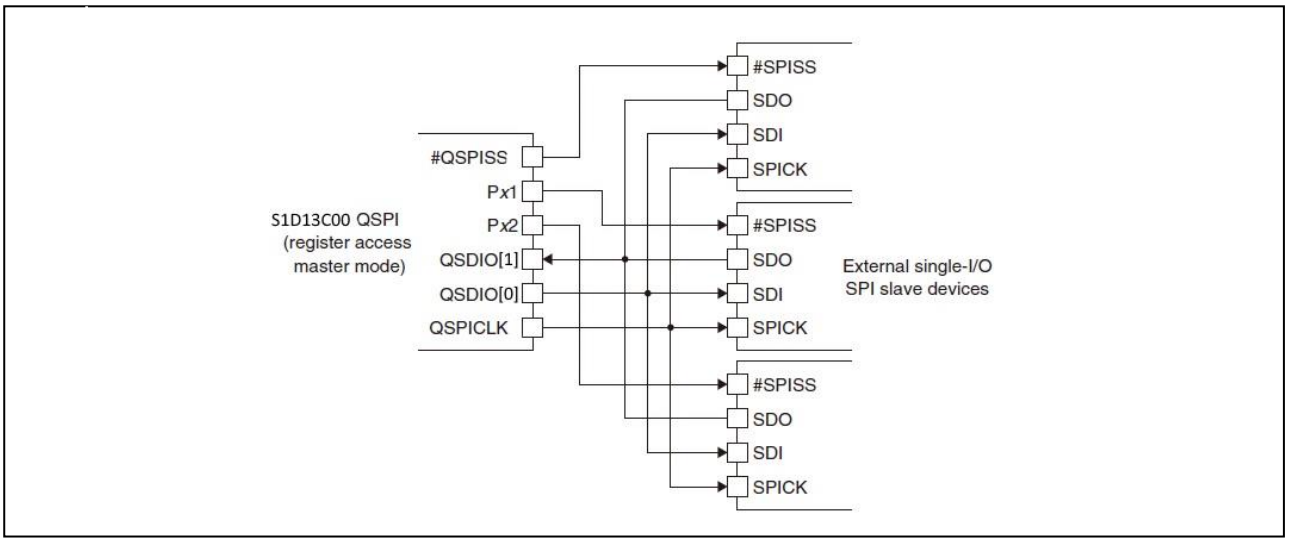


Figure 17.3 Connections between QSPI in Register Access Master Mode and External Single-I/O SPI (Legacy SPI) Slave Devices

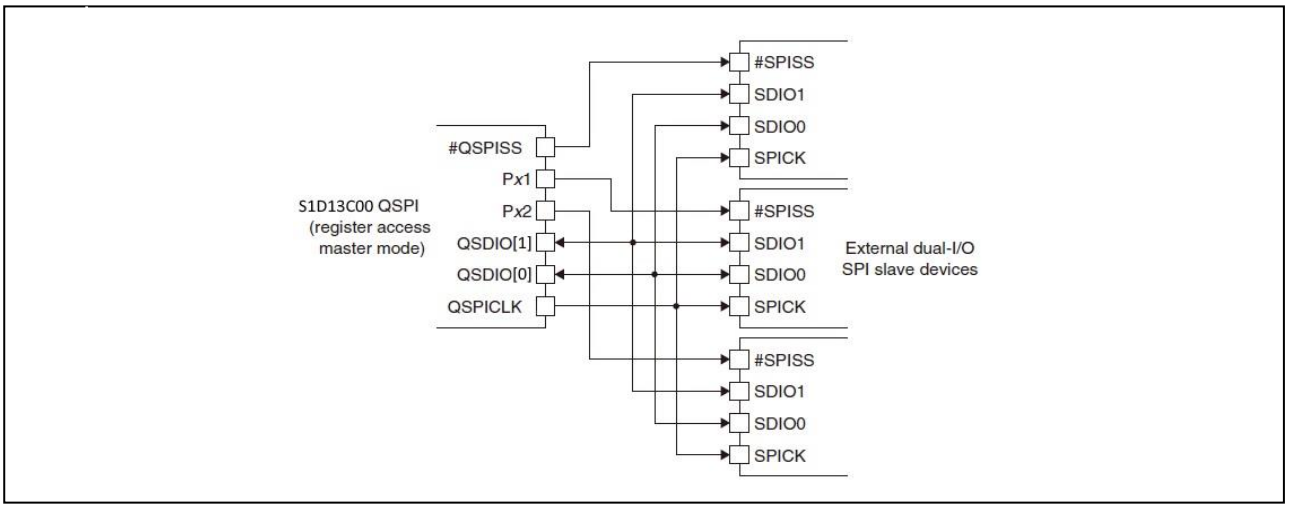


Figure 17.4 Connections between QSPI in Register Access Master Mode and External Dual-I/O SPI Slave Devices

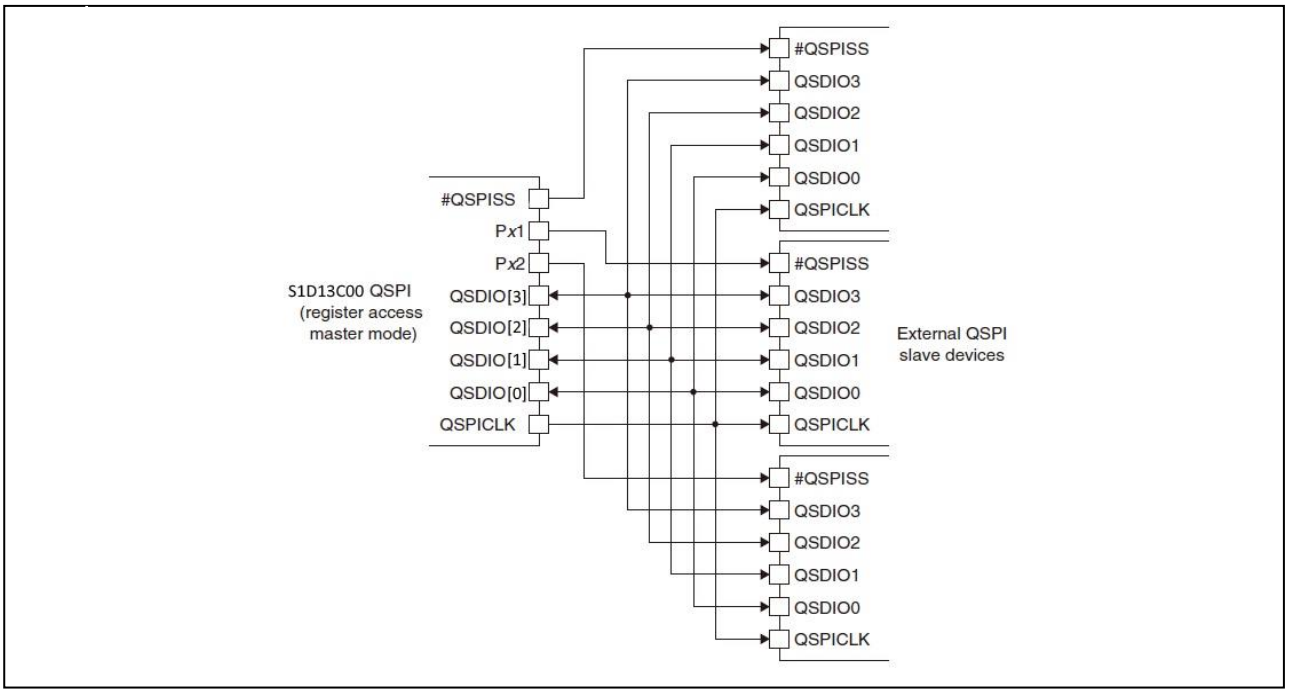


Figure 17.5 Connections between QSPI in Register Access Master Mode and External QSPI Slave Devices

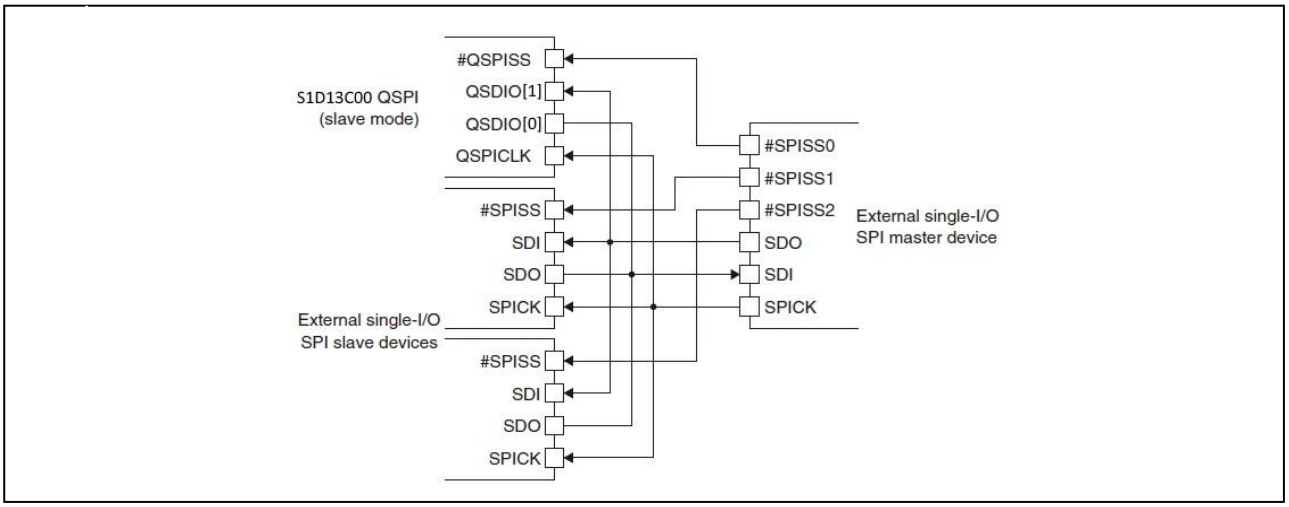


Figure 17.6 Connections between QSPI in Slave Mode and External Single-I/O SPI (Legacy SPI) Master Device

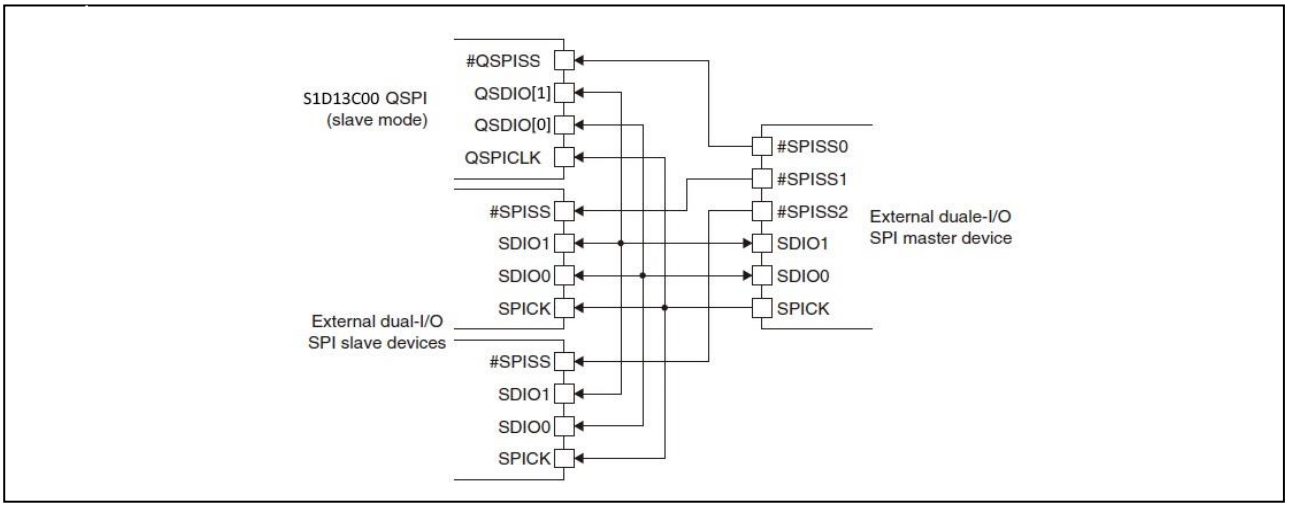


Figure 17.7 Connections between QSPI in Slave Mode and External Dual-I/O SPI Master Device

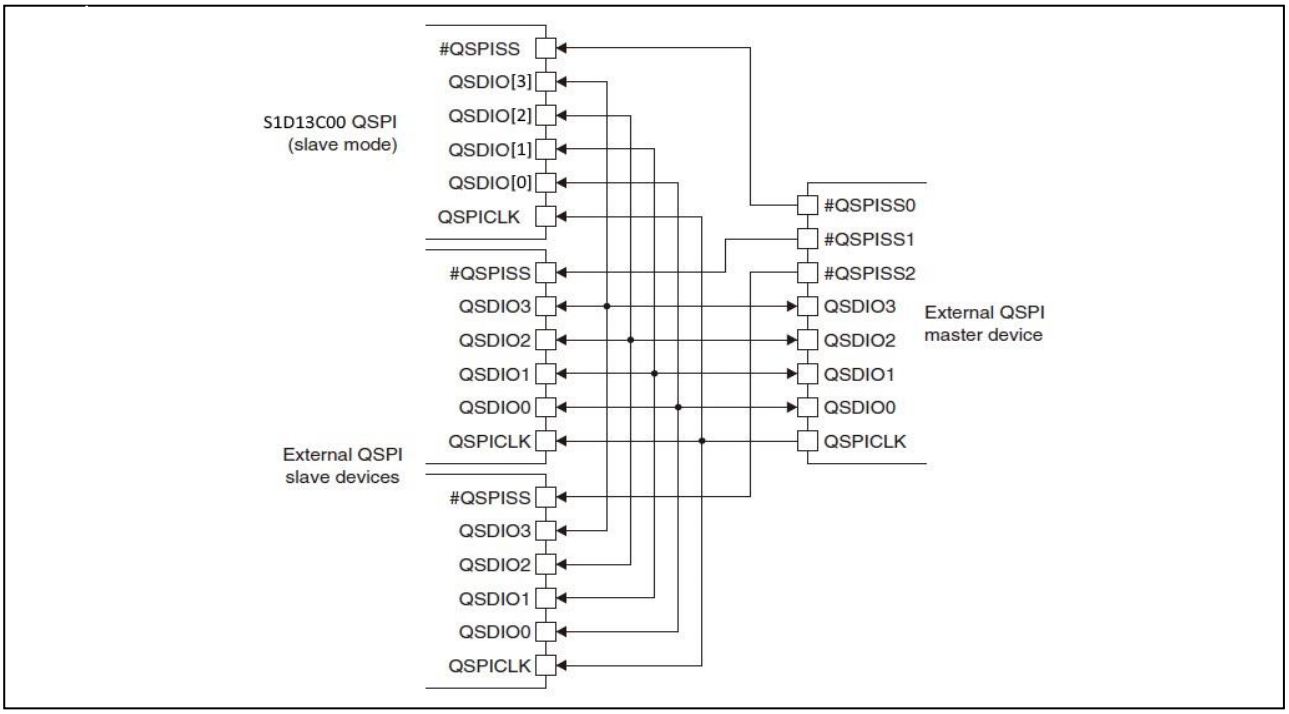


Figure 17.8 Connections between QSPI in Slave Mode and External QSPI Master Device

17.2.3 Pin Functions in Master Mode and Slave Mode

The pin functions are changed according to the transfer direction, transfer mode, and master/slave mode selections. The differences in pin functions between the modes are shown in Table 17.2.

Table 17.2 Pin Function Differences between Modes

<table>
<tr><th rowspan="2">Pin</th><th colspan="3">Function in master mode</th><th colspan="3">Function in slave mode</th></tr>
<tr><th>Single Transfer Mode</th><th>Dual Transfer Mode</th><th>Quad Transfer Mode</th><th>Single Transfer Mode</th><th>Dual Transfer Mode</th><th>Quad Transfer Mode</th></tr>
<tr><td>QSDIO[3:2]</td><td colspan="2">Always placed into Hi-Z state</td><td rowspan="3">These pins are placed into input or output state according to the QSPICTL.DIR bit setting.</td><td colspan="2">Always placed into Hi-Z state</td><td rowspan="3"></td></tr>
<tr><td>QSDIO[1]</td><td>Always placed into input state</td><td rowspan="2">These pins are placed into input or output state according to the QSPICTL.DIR bit setting.</td><td>Always placed into input state</td><td rowspan="2">These pins are placed into output state while a low level is applied to the #QSPISS pin and the QSPICTL.DIR bit is set to 0 (output), or placed into Hi-Z state while a high level is applied to the #QSPISS pin or the QSPICTL.DIR bit is set to 1 (input).</td><td rowspan="2">These pins are placed into output state while a low level is applied to the #QSPISS pin and the QSPICTL.DIR bit is set to 0 (output), or placed into Hi-Z state while a high level is applied to the #QSPISS pin or the QSPICTL.DIR bit is set to 1 (input).</td></tr>
<tr><td>QSDIO[0]</td><td>Always placed into output state</td><td>This pin is placed into output state while a low level is applied to the #QSPISS pin or placed into Hi-Z state while a high level is applied to the #QSPISS pin.</td></tr>
<tr><td>QSPICLK</td><td colspan="3">Outputs the QSPI clock to external devices.
Output clock polarity and phase can be configured if necessary.</td><td colspan="3">Inputs an external QSPI clock.
Clock polarity and phase can be designated according to the input clock.</td></tr>
<tr><td>#QSPISS</td><td colspan="3">This pin is used to output the slave select signal in master mode. In memory mapped access mode, this pin is controlled by the internal state machine. In register access mode, this pin is controlled by a register bit. When connecting more than one external slave device, general purpose I/O ports can be used to output the extra slave select signals.</td><td colspan="3">Applying a low level to the #QSPISS pin enables the QSPI to transmit/receive data. While a high level is applied to this pin, the QSPI is not selected as a slave device. Data input to the QSDIO pins and the clock input to the QSPICLK pin are ignored. When a high level is applied, the transmit/receive bit count is cleared to 0 and the already received bits are discarded.</td></tr>
</table>

17.2.4 Input Pin Pull-Up/Pull-Down Function

The QSPI pins (QSDIO[3:0] pins in master mode or QSDIO[3:0] pins, QSPICLK, and #QSPISS pins in slave mode) have a pull-up or pull-down function as shown in Table 17.3. This function is enabled by setting the QSPIMOD.PUEN bit to 1.

Table 17.3 Pull-Up or Pull-Down of QSPI Pins

Pin	Master mode	Slave mode
QSDIO[3:0]	Pull-up	Pull-up
QSPICLK	-	QSPIMOD.CPOL bit = 1: Pull-up QSPIMOD.CPOL bit = 0: Pull-down
#QSPISS	-	Pull-up

17.3 Clock Settings

17.3.1 QSPI Operating Clock

Operating clock in master mode

In master mode, the QSPI operating clock is supplied from the 16-bit timer. The following two options are provided for the clock configuration.

Use the 16-bit timer operating clock without dividing

By setting the QSPIMOD.NOCLKDIV bit to 1, the operating clock CLK_T16_2, which is configured by selecting a clock source and a division ratio, for the 16-bit timer channel corresponding to the QSPI is input to the QSPI as CLK_QSPI. Since this clock is also used as the QSPI clock QSPICLK without changing, the CLK_QSPI frequency becomes the baud rate. To supply CLK_QSPI to the QSPI, the 16-bit timer clock source must be enabled in the clock generator. It does not matter how the T16_2CTL.MODEN and T16_2CTL.PRUN bits of the corresponding 16-bit timer channel are set (1 or 0). When setting this mode, the timer function of the corresponding 16-bit timer channel may be used for another purpose.

Use the 16-bit timer as a baud rate generator

By setting the QSPIMOD.NOCLKDIV bit to 0, the QSPI inputs the underflow signal generated by the corresponding 16-bit timer channel and converts it to the QSPICLK. The 16-bit timer must be run with an appropriate reload data set. The QSPICLK frequency (baud rate) and the 16-bit timer reload data are calculated by the equations shown below.

$$f_{QSPICLK} = \frac{f_{CLK\_QSPI}}{2 \times (RLD + 1)} \qquad RLD = \frac{f_{CLK\_QSPI}}{f_{QSPICLK} \times 2} - 1$$

Where

$f_{QSPICLK}$: QSPICLK frequency [Hz] (= baud rate [bps])
f_{CLK_QSPI}: QSPI operating clock frequency [Hz]
RLD: 16-bit timer reload data value

Operating clock in slave mode

The QSPI set in slave mode operates with the clock supplied from the external SPI/QSPI master to the QSPICLK pin. The 16-bit timer channel (including the clock source selector and the divider) corresponding to the QSPI is not used. Furthermore, the QSPIMOD.NOCLKDIV bit setting becomes ineffective.
QSPI keeps operating using the clock supplied from the external SPI/QSPI master even if all the internal clocks are halted, so QSPI can receive data and can generate receive buffer full interrupts.

17.3.2 QSPI Clock (QSPICLK) Phase and Polarity

The QSPICLK phase and polarity can be configured separately using the QSPIMOD.CPHA bit and the QSPIMOD.CPOL bit, respectively. Figure 17.9 shows the clock waveform and data input/output timing in each setting.

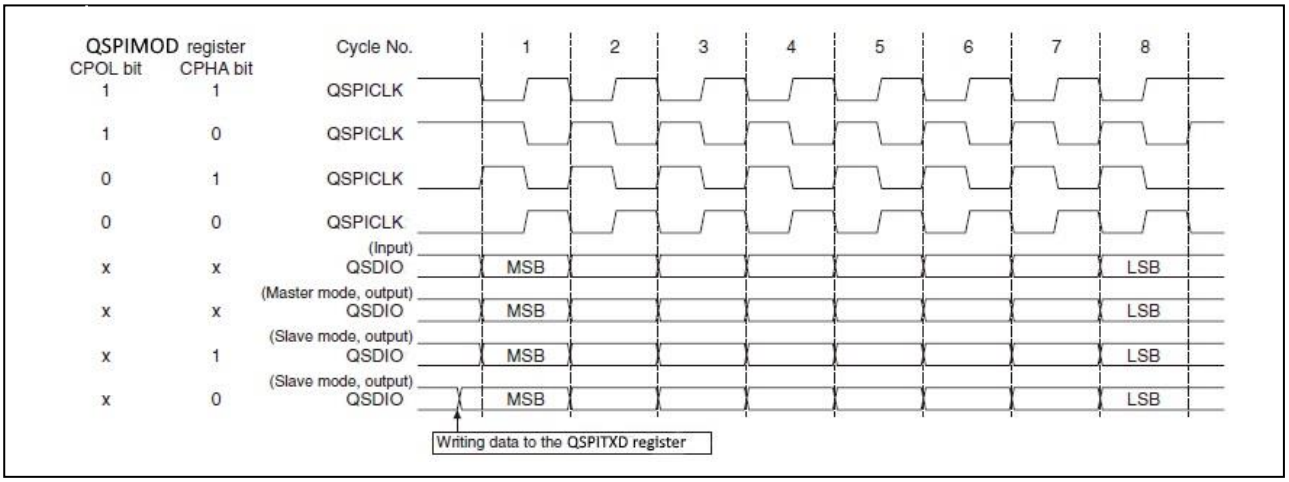


Figure 17.9 QSPI Clock Phase and Polarity (QSPIMOD.LSBFST bit = 0, QSPIMOD.CHLN[3:0] bits = 0x7)

17.4 Data Format

The QSPI data length can be selected from 2 to 16 clocks by setting the QSPIMOD.CHLN[3:0] bits. The input/output permutation is configurable to MSB first or LSB first using the QSPIMOD.LSBFST bit. Figure 17.10 to Figure 17.12 show data format examples in different transfer modes (QSPIMOD.TMOD[1:0]) when the QSPIMOD.CPOL bit = 0 and the QSPIMOD.CPHA bit = 0.

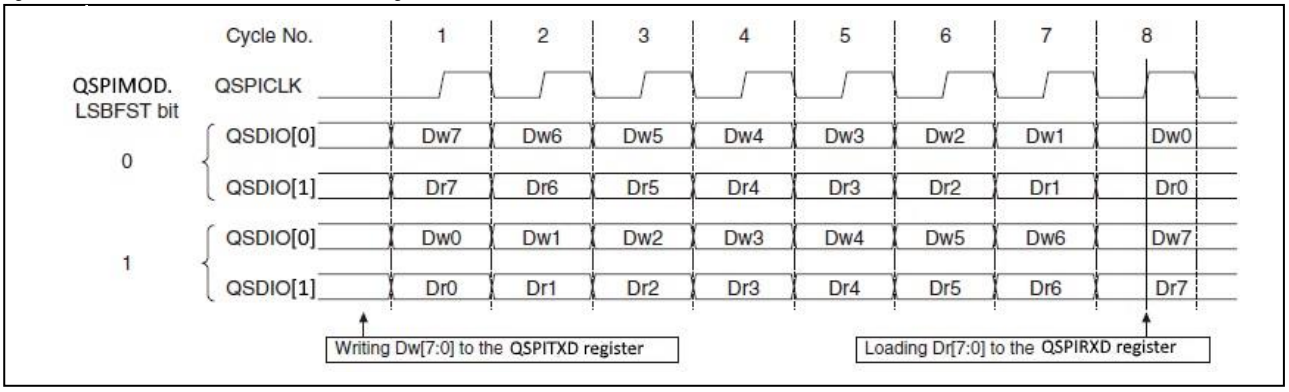


Figure 17.10 Data Format Selection for Single Transfer Mode Using the QSPIMOD.LSBFST Bit (QSPIMOD.TMOD[1:0] bits = 0x0, QSPIMOD.CHLN[3:0] bits = 0x7, QSPIMOD.CPOL bit = 0, QSPIMOD.CPHA bit = 0)

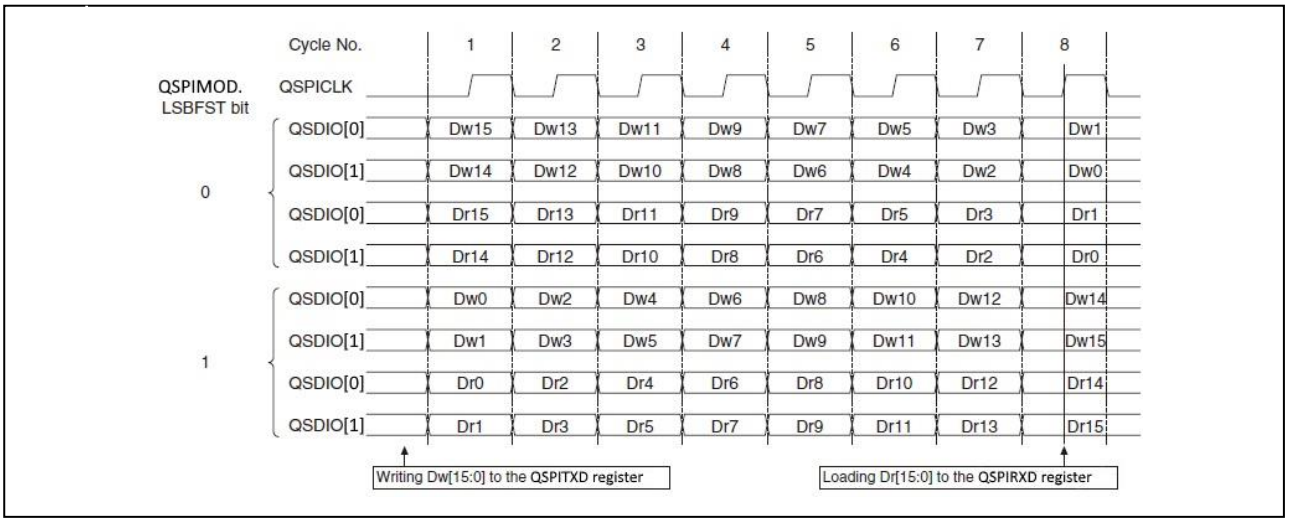


Figure 17.11 Data Format Selection for Dual Transfer Mode Using the QSPIMOD.LSBFST Bit (QSPIMOD.TMOD[1:0] bits = 0x1, QSPIMOD.CHLN[3:0] bits = 0x7, QSPIMOD.CPOL bit = 0, QSPIMOD.CPHA bit = 0)

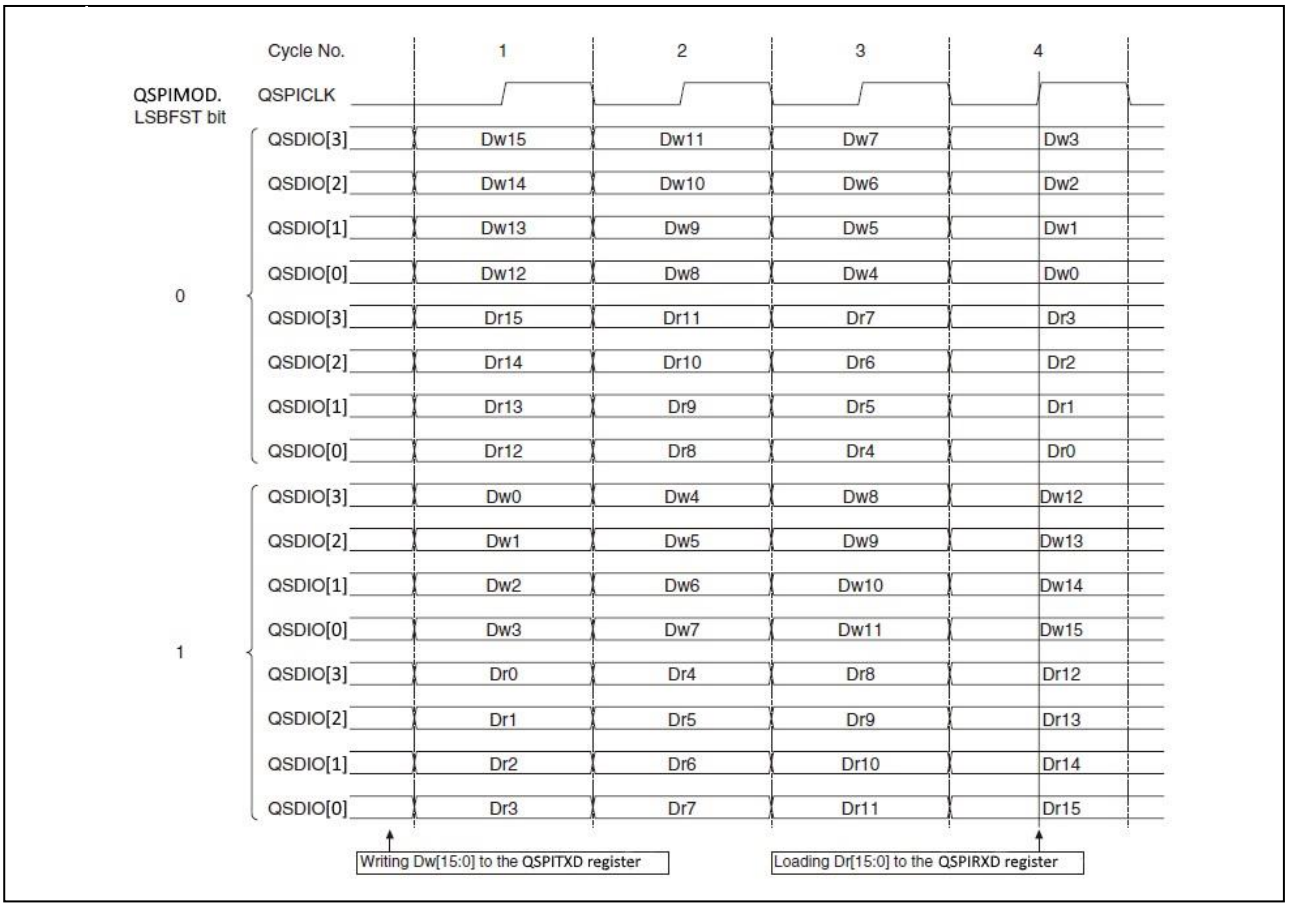


Figure 17.12 Data Format Selection for Quad Transfer Mode Using the QSPIMOD.LSBFST Bit (QSPIMOD.TMOD[1:0] bits = 0x2, QSPIMOD.CHLN[3:0] bits = 0x3, QSPIMOD.CPOL bit = 0, QSPIMOD.CPHA bit = 0)

17.5 Operations

17.5.1 Register Access Mode

Data can be read from or written to the external SPI/QSPI device by accessing the registers in both master and slave modes.
In single transfer mode, transmit data are always output from the QSDIO[0] pin and receive data are always input to the QSDIO[1] pin (the QSDIO[3:2] pins are not used). The operations are backward compatible with legacy SPI (e.g., synchronous serial interface of this MCU).
In dual transfer mode, transmit data are output from the QSDIO[1:0] pins when the transfer direction is set to output (QSPICTL.DIR bit = 0). Receive data are input from the QSDIO[1:0] pins when the transfer direction is set to input (QSPICTL.DIR bit = 1). The QSDIO[3:2] pins are not used. The number of data transfer clocks is configured using the QSPIMOD.CHLN[3:0] bits. Since two data lines are used for data transfer, the data bit length (number of clocks) is obtained by dividing the number of transfer data bits by two.
In quad transfer mode, transmit data are output from the QSDIO[3:0] pins when the transfer direction is set to output (QSPICTL.DIR bit = 0). Receive data are input from the QSDIO[3:0] pins when the transfer direction is set to input (QSPICTL.DIR bit = 1). The number of data transfer clocks is configured with the QSPIMOD.CHLN[3:0] bits. Since four data lines are used for data transfer, the data bit length (number of clocks) is obtained by dividing the number of transfer data bits by four.

LENGTH = (BIT / N) [clocks]

Where,

LENGTH:	Data bit length [clocks]
BIT:	Number of transfer data bits
N:	1 (single transfer mode), 2 (dual transfer mode), or 4 (quad transfer mode)

17.5.2 Memory Mapped Access Mode

Memory mapped access mode is a low CPU overhead operation mode used with master mode to read data from an external Flash memory, which supports XIP (eXecute-In-Place) mode. Once the external Flash memory enters XIP mode and a read command is executed, the same read command operation can be performed by controlling the slave select signal (inactive to active) and sending a new address to be accessed without the command being resent. This may reduce command re-execution overhead and random access time.
An XIP session consists of a command cycle, an address cycle, a dummy cycle, and consecutive data cycles, and it begins with an XIP specific read command similar to a general read command. Unlike a general read command, one or more data lines must be driven to send XIP activation or termination confirmation bit(s) at the beginning of the dummy cycle of an XIP session.
In an XIP session, to start reading from a non-sequential Flash memory address, which is not continuous to the previous read address, assert the slave signal again after negating it once. After that, just send an address cycle to specify the new read start address and a dummy cycle including an XIP activation (continuation) confirmation bit(s), as the command cycle is not needed in this XIP session. The Flash memory performs read operations the same as the read command previously executed to execute a data cycle that includes a given number of data stored from the newly specified address.
To terminate an XIP session, first assert the slave signal again after negating it once. Then, send an address cycle with the address bits set to all high (suggested by most Flash memory manufacturers) and a dummy cycle including an XIP termination confirmation bit(s) at the beginning of the cycle on one or more data lines. After that, negate the slave select signal.
Figure 17.13 and Figure 17.14 show Spansion S25FL128S Quad I/O Read command sequences as XIP operation examples.

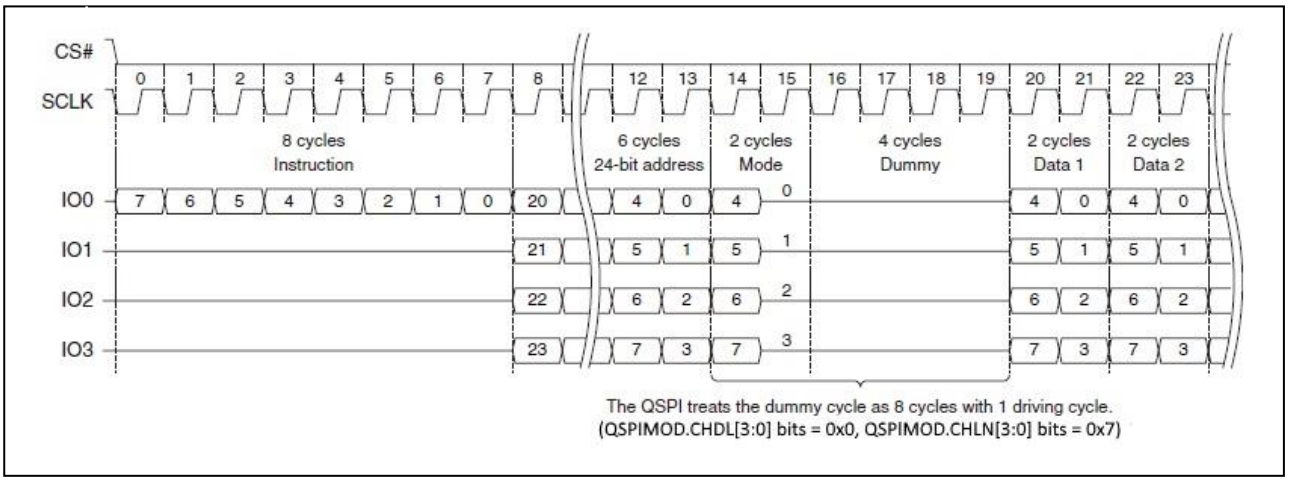


Figure 17.13 XIP Example - Spansion S25FL128S Quad I/O Read Command Sequence (3-byte address, 0xeb [ExtAdd = 0], LC = 0b00)

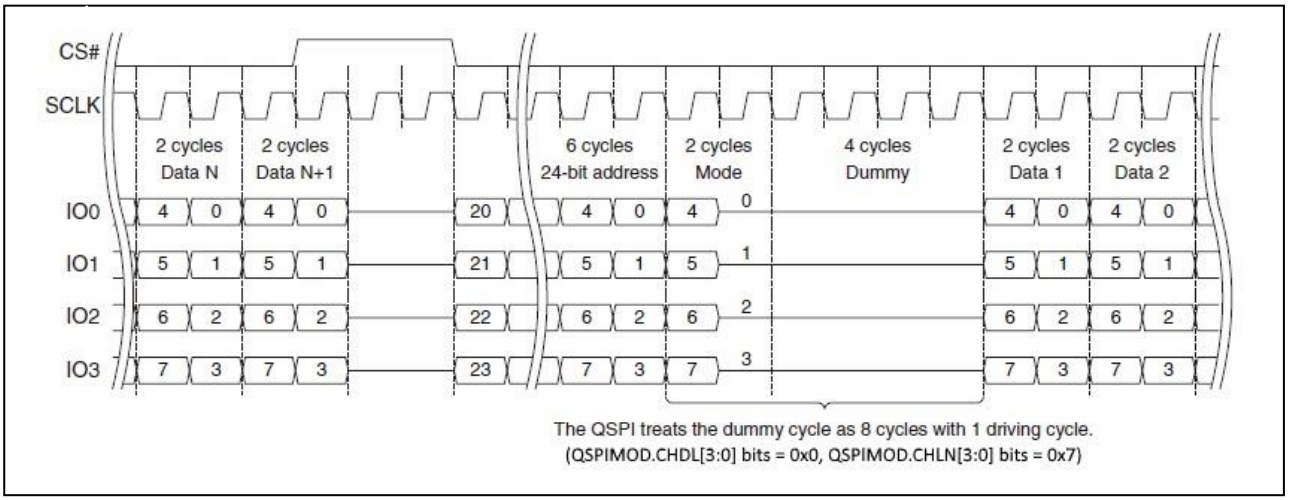


Figure 17.14 XIP Example - Spansion S25FL128S Continuous Quad I/O Read Command Sequence (3-byte address, LC = 0b00)

In memory mapped access mode, the QSPI automates toggling of the slave select signal and executing address, dummy, and data cycles so that the CPU will be able to read the external Flash memory mapped to the system memory area. This further reduces CPU overhead.
The transfer mode can be configured for address, dummy, and data cycles individually. The address cycle supports 24 and 32-bit addresses. The QSPI considers that the mode cycle (or XIP activation/termination confirmation) is a part of the dummy cycle, so a mode cycle is sent out on the I/O data line in a dummy cycle.
The 1M-byte system memory area starting at address 0x40000 is used to map the external Flash memory and to access from the CPU. Up to 4G-byte Flash memory can be accessed from this area using a remapping register.
Once the external Flash memory is set into XIP mode and a read command is sent in register access mode, the CPU can directly read external Flash memory data through this area. When a read access to a non-sequential address occurs in memory mapped access mode, the QSPI automatically executes a new address and dummy cycles.
When memory mapped access mode is disabled by setting a register, the QSPI executes an address cycle and a dummy cycle including a mode byte that specifies to terminate XIP mode.
Memory mapped access mode supports 8, 16, and 32-bit read accesses.
The 32-bit access is mainly used to read data in a large memory block sequentially. In this access, up to two 32-bit data are prefetched into the internal FIFO. Therefore, zero-wait read access is possible if the desired data has already been fetched in the FIFO.
The 8 and 16-bit accesses are mainly used to read data in a small memory block or to read data from non-sequential

addresses. Prefetching is not performed as it is unnecessary in non-sequential read. Therefore, overhead of a couple of clocks occurs between accesses.
The QSPI allows incorporating 8 and 16-bit accesses into 32- bit accesses. Prefetching data into FIFO is only performed immediately after a 32-bit read. An 8 or 16-bit read at the sequential address after a 32-bit read allows zerowait read if the desired data has already been fetched in the FIFO.

17.5.3 Initialization

QSPI should be initialized with the procedure shown below.
1. <Master mode only> Generate a clock by controlling the 16-bit timer and supply it to QSPI.
2. Configure the following QSPIMOD register bits:
   - QSPIMOD.PUEN bit (Enable input pin pull-up/down)
   - QSPIMOD.NOCLKDIV bit (Select master mode operating clock)
   - QSPIMOD.LSBFST bit (Select MSB first/LSB first)
   - QSPIMOD.CPHA bit (Select clock phase)
   - QSPIMOD.CPOL bit (Select clock polarity)
   - QSPIMOD.MST bit (Select master/slave mode)
3. Configure the following register bits when using memory mapped access mode:
   - QSPIMMACFG1.TCSH[3:0] bits (Set slave select signal negation period)
   - QSPIRMADRH.RMADR[31:20] bits (Set remapping address)
   - QSPIMMACFG2.DUMDL[3:0] bits (Select dummy cycle drive length)
   - QSPIMMACFG2.DUMLN[3:0] bits (Select dummy cycle length)
   - QSPIMMACFG2.DATTMOD[1:0] bits (Select data cycle transfer mode)
   - QSPIMMACFG2.DUMTMOD[1:0] bits (Select dummy cycle transfer mode)
   - QSPIMMACFG2.ADRTMOD[1:0] bits (Select address cycle transfer mode)
   - QSPIMMACFG2.ADRCYC bit (Select 24 or 32-bit address cycle)
   - QSPIMB.XIPACT[7:0] bits (Set XIP activation mode byte)
   - QSPIMB.XIPEXT[7:0] bits (Set XIP termination mode byte)
4. Assign the QSPI input/output function to the ports. (Refer to the "GPIO Ports" chapter.)
5. Set the following QSPICTL register bits:
   - Set the QSPICTL.SFTRST bit to 1. (Execute software reset)
   - Set the QSPICTL.MODEN bit to 1. (Enable QSPI Ch.n operations)
6. Set the following bits when using the interrupt:
   - Write 1 to the interrupt flags in the QSPIINTF register. (Clear interrupt flags)
   - Set the interrupt enable bits in the QSPIINTE register to 1. * (Enable interrupts)
   * The initial value of the QSPIINTF.TBEIF bit is 1, therefore, an interrupt will occur immediately after the QSPIINTE.TBEIE bit is set to 1.
7. Configure the DMA controller and set the following QSPI control bits when using DMA transfer:
   - Write 1 to the DMA transfer request enable bits in the QSPITBEDMAEN, QSPIRBFDMAEN, and QSPIFRLDMAEN registers. (Enable DMA transfer requests)

17.5.4 Data Transmission in Master Mode

A data sending procedure and operations in master mode are shown below. Figure 17.15 and Figure 17.16 show a timing chart and a flowchart, respectively.

Data sending procedure
1. Set the QSPICTL.DIR bit to 0 when QSPI is set to dual or quad transfer mode. (This setting is not necessary in single transfer mode.)
2. Assert the slave select signal for the external slave device to be accessed by controlling the QSPICTL.MSTSSO bit or the general-purpose output port used for an extra slave select signal output (if necessary).
3. Check to see if the QSPIINTF.TBEIF bit is set to 1 (transmit buffer empty).
4. Write transmit data to the QSPITXD register.
5. Wait for a QSPI interrupt when using interrupt.
6. Repeat Steps 3 to 5 (or 3 and 4) until the end of transmit data.

7. Negate the slave select signal that has been asserted in Step 2 by controlling the QSPICTL.MSTSSO bit or the general-purpose output port (if necessary).

Data sending operations

QSPI starts data sending operations when transmit data is written into the QSPITXD register.
The transmit data in the QSPITXD register is automatically transferred to the shift register and the QSPIINTF.TBEIF bit is set to 1. If the QSPIINTE.TBEIE bit = 1 (transmit buffer empty interrupt enabled), a transmit buffer empty interrupt occurs at the same time.
The QSPICLK pin outputs clocks for the number of cycles specified by the QSPIMOD.CHLN[3:0] bits and the transmit data bits are output in sequence from the QSDIO pins, according to the transfer mode specified by the QSPIMOD.TMOD[1:0] bits, in sync with these clocks.
Even if the clock is being output from the QSPICLK pin, the next transmit data can be written to the QSPITXD register after making sure the QSPIINTF.TBEIF bit is set to 1.
If transmit data has not been written to the QSPITXD register after the last clock is output from the QSPICLK pin, the clock output halts and the QSPIINTF.TENDIF bit is set to 1. At the same time QSPI issues an end-of-transmission interrupt request if the QSPIINTE.TENDIE bit = 1.

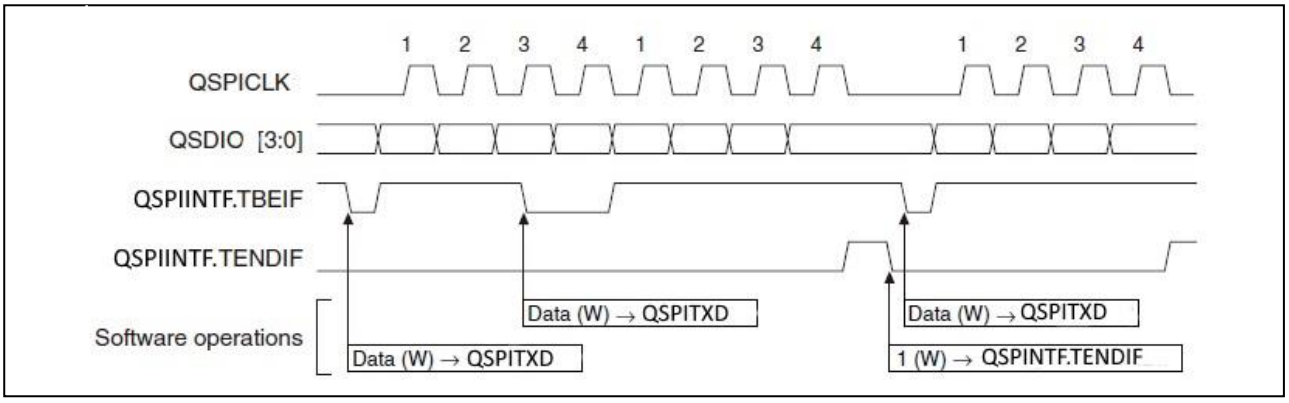


Figure 17.15 Example of Data Sending Operations in Master Mode

(QSPIMOD.CHDL[3:0] bits = QSPIMOD.CHLN[3:0] bits = 0x3)

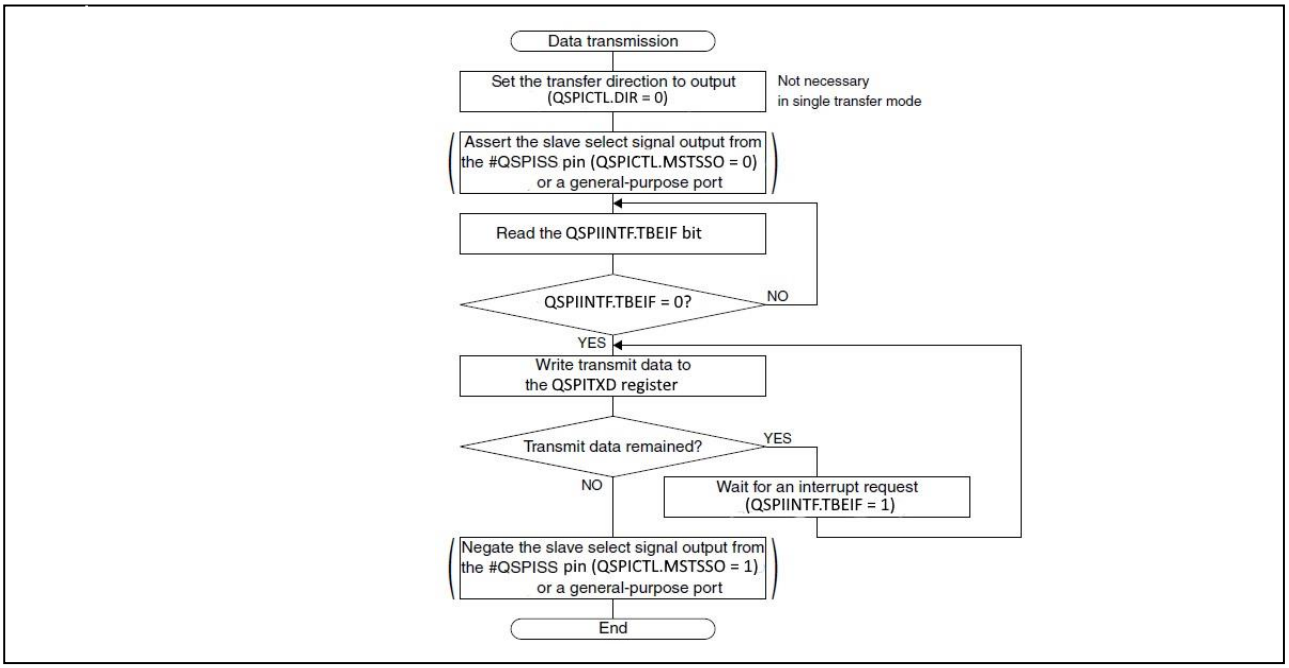


Figure 17.16 Data Transmission Flowchart in Master Mode

Data transmission using DMA

By setting the QSPITBEDMAEN.TBEDMAENx bit to 1 (DMA transfer request enabled), a DMA transfer request is sent to the DMA controller and transmit data is transferred from the specified memory to the QSPITXD register via DMA Ch.x when the QSPIINTF.TBEIF bit is set to 1 (transmit buffer empty). This automates the procedure from Step 3 to Step 6 described above.
The transfer source/destination and control data must be set for the DMA controller and the relevant DMA channel must be enabled to start a DMA transfer in advance so that transmit data will be transferred to the QSPITXD register. For more information on DMA, refer to the "DMA Controller" chapter.

Table 17.4 DMA Data Structure Configuration Example (for 16-bit Data Transmission)

Item		Setting example
End pointer	Transfer source	Memory address in which the last transmit data is stored
	Transfer destination	QSPITXD register address
Control data	dst_inc	0x3 (no increment)
	dst_size	0x1 (halfword)
	src_inc	0x1 (+2)
	src_size	0x1 (halfword)
	R_power	0x0 (arbitrated for every transfer)
	n_minus_1	Number of transfer data
	cycle_ctrl	0x1 (basic transfer)

17.5.5 Data Reception in Register Access Master Mode

A data receiving procedure and operations in register access master mode are shown below. Figure 17.17 and Figure 17.18 show a timing chart and flowcharts, respectively.

Data receiving procedure

1. Set the QSPICTL.DIR bit to 1 when QSPI is set to dual or quad transfer mode. (This setting is not necessary in single transfer mode.)
2. Assert the slave select signal for the external slave device to be accessed by controlling the QSPICTL.MSTSSO bit or the general-purpose output port used for an extra slave select signal output (if necessary).
3. Check to see if the QSPIINTF.TBEIF bit is set to 1 (transmit buffer empty).
4. Write dummy data (or transmit data) to the QSPITXD register.
5. Wait for a transmit buffer empty interrupt (QSPIINTF.TBEIF bit = 1).
6. Write dummy data (or transmit data) to the QSPITXD register.
7. Wait for a receive buffer full interrupt (QSPIINTF.RBFIF bit = 1).
8. Read the received data from the QSPIRXD register.
9. Repeat Steps 6 to 8 until the end of data reception.
10. Negate the slave select signal that has been asserted in Step 2 by controlling the QSPICTL.MSTSSO bit or the general-purpose output port (if necessary).

Note: To perform continuous data reception without stopping QSPICLK, Steps 8 and 6 operations must be completed within the QSPICLK cycles equivalent to "Data bit length - 1" after Step 7.

Data receiving operations

In single transfer mode (QSPIMOD.TMOD[1:0] bits = 0), QSPI operates similar to legacy SPI devices.
The data receiving operation starts simultaneously with a data sending operation when transmit data (may be dummy data if data transmission is not required) is written to the QSPITXD register. Transmit data are output from the QSDIO[0] pin and receive data are input from the QSDIO[1] pin.
In dual or quad transfer mode (QSPIMOD.TMOD[1:0] bits = 1 or 2), transmit data are not sent at data reception. Writing dummy data to the QSPITXD register triggers the QSPI to start supplying the data transfer clock from the QSPICLKn pin to the slave device.

The QSPICLK pin outputs the number of clocks specified by the QSPIMOD.CHLN[3:0] bits. The receive data bits input from the QSDIO pins, according to the transfer mode specified by the QSPIMOD.TMOD[1:0] bits, are shifted into the shift register in sync with these clocks.
When the last clock is output from the QSPICLK pin and receive data bits are all shifted into the shift register, the received data is transferred to the receive data buffer and the QSPIINTF.RBFIF bit is set to 1. At the same time QSPI issues a receive buffer full interrupt request if the QSPIINTE.RBFIE bit = 1. After that, the received data in the receive data buffer can be read through the QSPIRXD register.

Note: If data of the number of the bits specified by the QSPIMOD.CHLN[3:0] bits and QSPIMOD.TMOD[1:0] bits is received when the QSPIINTF.RBFIF bit is set to 1, the QSPIRXD register is overwritten with the newly received data and the previously received data is lost. In this case, the QSPIINTF.OEIF bit is set.

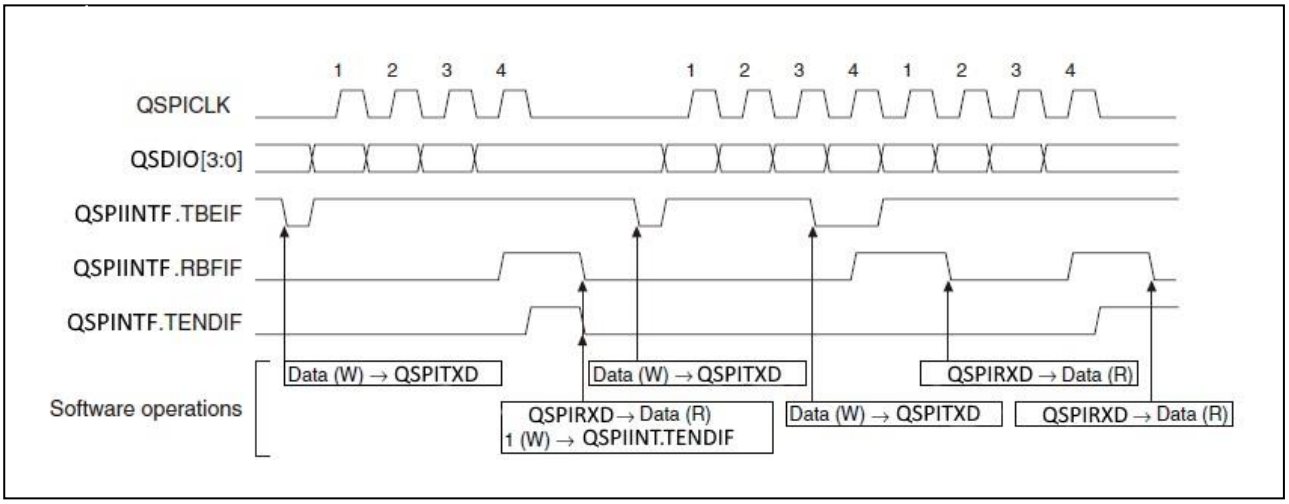


Figure 17.17 Example of Data Receiving Operations in Register Access Master Mode (QSPIMOD.CHDL[3:0] bits = QSPIMOD.CHLN[3:0] bits = 0x3)

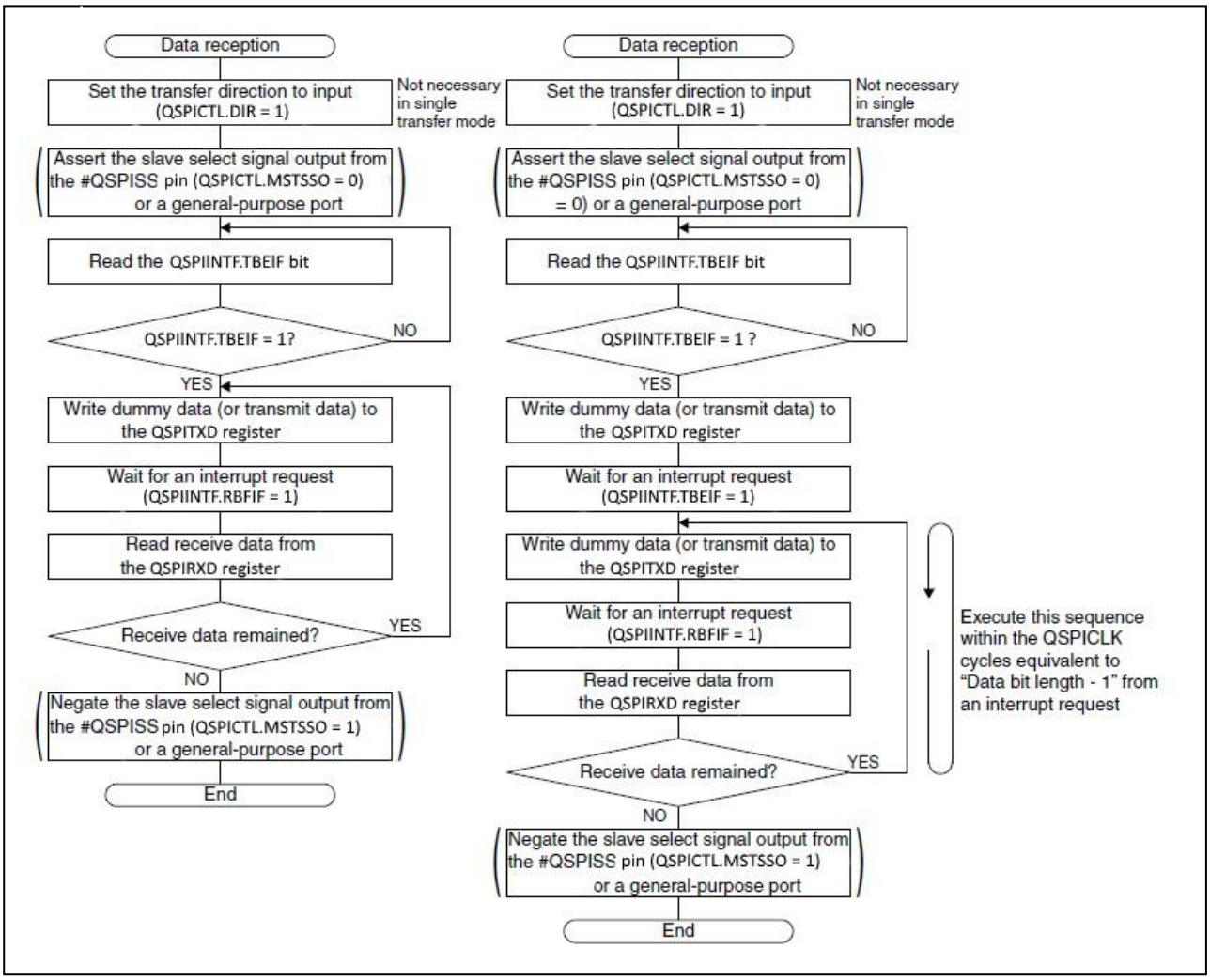


Figure 17.18 Data Reception Flowcharts in Register Access Master Mode

Data reception using DMA

For data reception, two DMA controller channels should be used to write dummy data to the QSPITXD register as a reception start trigger and to read the received data from the QSPIRXD register.
By setting the QSPITBEDMAEN.TBEDMAENx1 bit to 1 (DMA transfer request enabled), a DMA transfer request is sent to the DMA controller and dummy data is transferred from the specified memory to the QSPITXD register via DMA Ch.x1 when the QSPIINTF.TBEIF bit is set to 1 (transmit buffer empty).

By setting the QSPIRBFDMAEN.RBFDMAENx2 bit to 1 (DMA transfer request enabled), a DMA transfer request is sent to the DMA controller and the received data is transferred from the QSPIRXD register to the specified memory via DMA Ch.x2 when the QSPIINTF.RBFIF bit is set to 1 (receive buffer full).
This automates the procedure from Step 3 to Step 9 described above.
The transfer source/destination and control data must be set for the DMA controller and the relevant DMA channel must be enabled to start a DMA transfer in advance. For more information on DMA, refer to the "DMA Controller" chapter.

Table 17.5 DMA Data Structure Configuration Example (for Writing 16-bit Dummy Transmit Data)

Item		Setting example
End pointer	Transfer source	Memory address in which dummy data is stored
	Transfer destination	QSPITXD register address
Control data	dst_inc	0x3 (no increment)
	dst_size	0x1 (halfword)
	src_inc	0x3 (no increment)
	src_size	0x1 (halfword)
	R_power	0x0 (arbitrated for every transfer)
	n_minus_1	Number of transfer data
	cycle_ctrl	0x1 (basic transfer)

Table 17.6 DMA Data Structure Configuration Example (for 16-bit Data Reception)

Item		Setting example
End pointer	Transfer source	QSPIRXD register address
	Transfer destination	Memory address to which the last received data is stored
Control data	dst_inc	0x1 (+2)
	dst_size	0x1 (halfword)
	src_inc	0x3 (no increment)
	src_size	0x1 (halfword)
	R_power	0x0 (arbitrated for every transfer)
	n_minus_1	Number of transfer data
	cycle_ctrl	0x1 (basic transfer)

The following shows an example of the control procedure including the DMA controller operations:

1. Configure the primary data structure for the DMA channel (Ch.x) used for writing dummy bytes to the QSPITXD register as shown in Table 17.5.
2. Configure the primary data structure for the DMA channel (Ch.y) used for reading data from the QSPIRXD register as shown in Table 17.6.
3. Enable both the DMA channels using the DMA controller register.
4. Increase the priority of the DMA channel used for reading data using the DMA controller register.
5. Clear the channel request masks for both the DMA channels using the DMA controller register.
6. Clear the DMA transfer completion interrupt flags using the DMA controller register.
7. Enable only the DMA transfer completion interrupt of the DMA channel used for reading using the DMA controller register.
8. Clear pending DMA interrupts in the host MCU.
9. Enable pending DMA interrupts in the host MCU.
10. Enable the QSPI to issue DMA transfer requests to both the DMA channels using the QSPITBEDMAEN.TBEDMAENx and QSPIRBFDMAEN.RBFDMAENy bits.
11. Assert the slave select signal by controlling the QSPICTL.MSTSSO bit, or the general-purpose output port used for an extra slave select signal output (if necessary).
12. Issue a software DMA transfer request to the DMA channel used for writing dummy bytes by setting the DMA controller register. This operation is required to read the first data and to set the receive buffer full status flag. Once the receive buffer full status flag is set, a hardware DMA request is generated, and the DMA controller transfers data from the QSPI_nRXD register and then writes another dummy byte to the QSPITXD register, allowing the QSPI to read the next data.
13. Wait for a DMA interrupt.

14. Disable the DMA requests to be sent to both the DMA channels using the QSPITBEDMAEN.TBEDMAENx and QSPIRBFDMAEN.RBFDMAENy bits.
15. Set the channel request masks for both the DMA channels using the DMA controller register.
16. Disable both the DMA channels using the DMA controller register.
17. Negate the slave select signal by controlling the QSPICTL.MSTSSO bit or the general-purpose output port (if necessary).

17.5.6 Data Reception in Memory Mapped Access Mode

A data receiving procedure, and 32-bit and 8/16-bit received data read operations in memory mapped access mode are shown below. Figure 17.19 to Figure 17.21 show their timing charts and a flowchart.

Data receiving procedure

QSPI Flash memories of different manufacturers have a different XIP operation mode setup procedure. The procedure described below assumes that the external Flash memory has already been placed into XIP operation mode.

1. Send a read command that supports XIP mode to the external Flash memory.
   For the sending procedure, see Steps 1 to 5 of the data sending procedure described in Section 17.5.4, "Data Transmission in Master Mode." The slave select signal that has been asserted should be left unchanged.
2. Set the QSPIMADRH.RMADR[31:20] bits. (Remap external Flash memory)
3. Write 1 to the QSPIMMACFG2.MMAEN bit. (Enable memory mapped access mode)
4. Read the memory mapped access area with an 8, 16, or 32-bit memory read instruction.
   This operation directly reads data within the 1M-byte Flash memory area remapped to the memory mapped access area (1M-byte system memory area starting at address 0x40000) at Step 2.
5. Repeat Step 4 as needed.
   When reading an address outside the remapped area, start from Step 2 again after setting the QSPIMMACFG2.MMAEN bit to 0 once.

Data receiving operations (32-bit read)

In memory mapped access mode, the internal state machine detects the address in the memory mapped access area from which data is read. If it is the first read operation after the QSPI has entered memory mapped access mode, the state machine generates an address cycle and a dummy cycle (including the XIP activation confirmation bit(s)). At the same time, it pulls the HREADY signal on the internal system bus down to low.
The address cycle can be configured for 24 or 32-bit addresses and it consists of two transfer cycles. The state machine determines actual Flash memory address from the memory mapped access area start address, the read address in that area, and the external Flash memory remapping start address set using the QSPIRMADRH[31:20] bits. The first transfer cycle is an 8-bit transfer that sends the high-order 8 bits of the address (when 24-bit address cycle is configured) or a 16-bit transfer that sends the high-order 16 bits of the address (when 32-bit address cycle is configured). The second cycle is fixed at 16-bit transfer that sends the low order 16 bits of the address.
A dummy cycle follows. The XIP activation confirmation byte set in the QSPIMB.XIPACT[7:0] bits is sent at the beginning of the cycle.
Then, the state machine starts fetching data from the external Flash memory. Once 32-bit data has been fetched into the internal FIFO, the FIFO read level is incremented (FIFO data ready). At this time, the HREADY signal reverts to high and the data fetched into the FIFO is sent to the internal system bus. The state machine prefetches two more 32-bit data from the continuous address and stores it into the FIFO.
If the address in the memory mapped access area that is continuous to the previous read address is read when the FIFO contains the prefetched data (FIFO data ready status), the prefetched data is sent to the internal system bus with the HREADY signal held high (zero-wait read). If an address in the memory mapped access area that is not continuous to the previous read address is read, the HREADY signal is pulled down to low immediately and the FIFO read level is cleared to 0 (empty status). The #QSPISS signal is negated once for the period set in the QSPIMMACFG1.TCSH[3:0] bits and then asserted again. After that a new address cycle, dummy cycle, and data cycle are executed.
The beginning and the end of each address, dummy, or data cycle take a couple of HCLK clocks for handshaking.

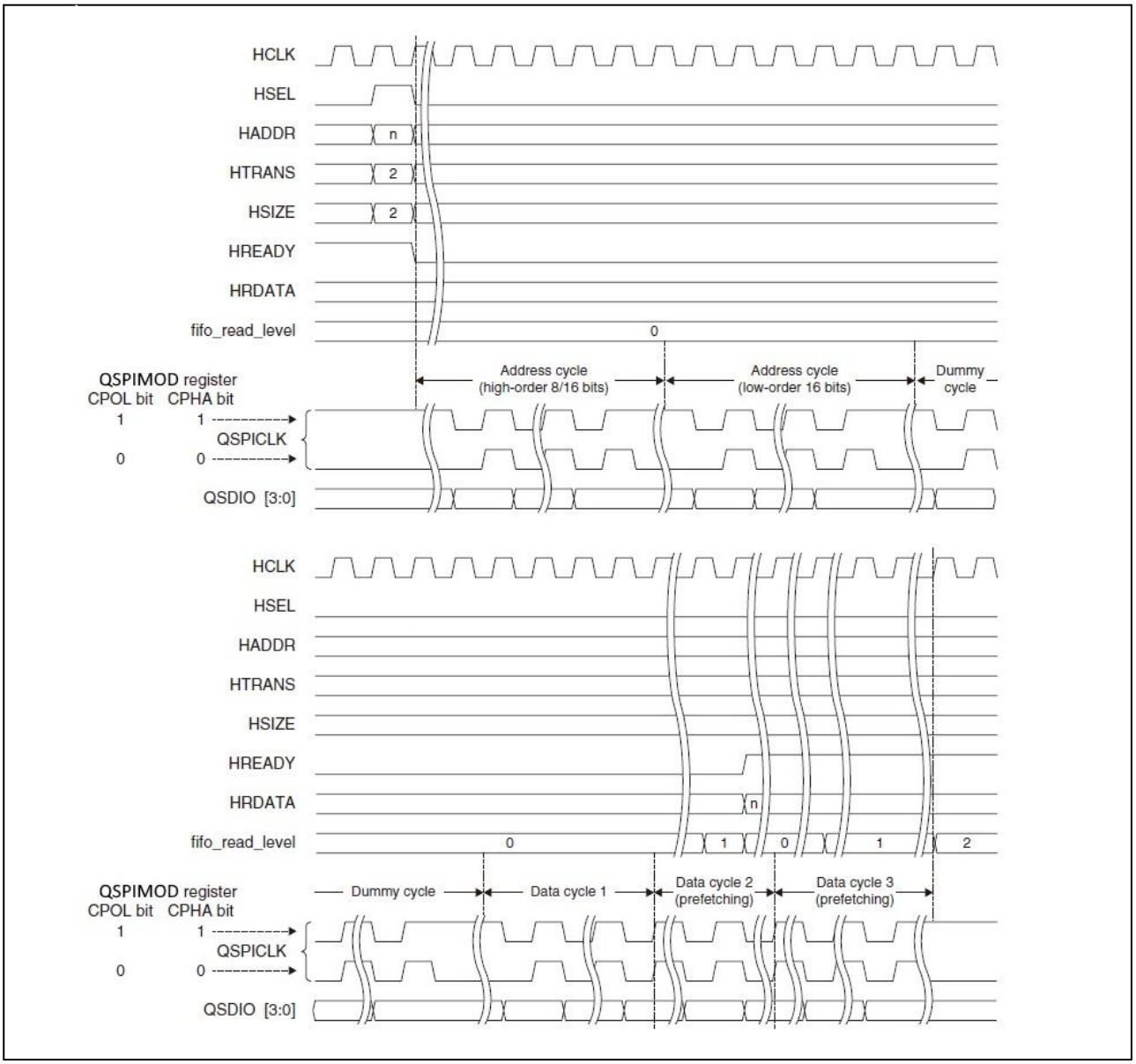


Figure 17.19 Data Receiving Operation in Memory Mapped Access Mode - First 32-bit Read

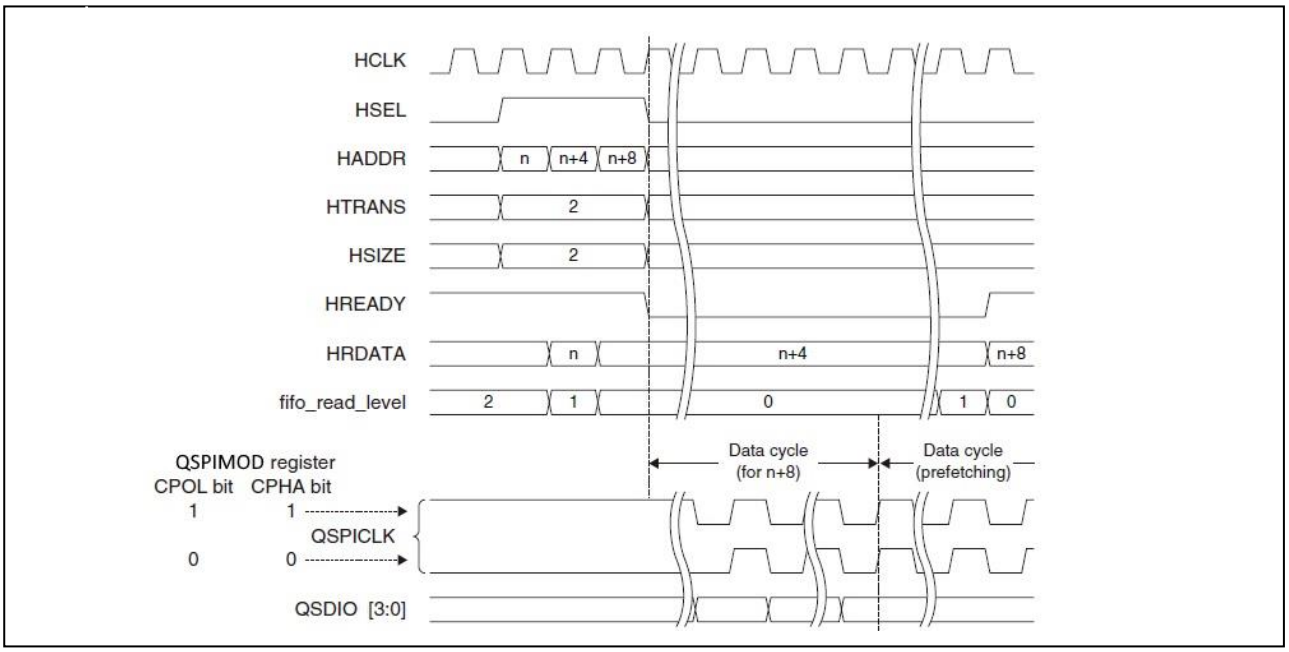


Figure 17.20 Data Receiving Operation in Memory Mapped Access Mode - 32-bit Sequential Read

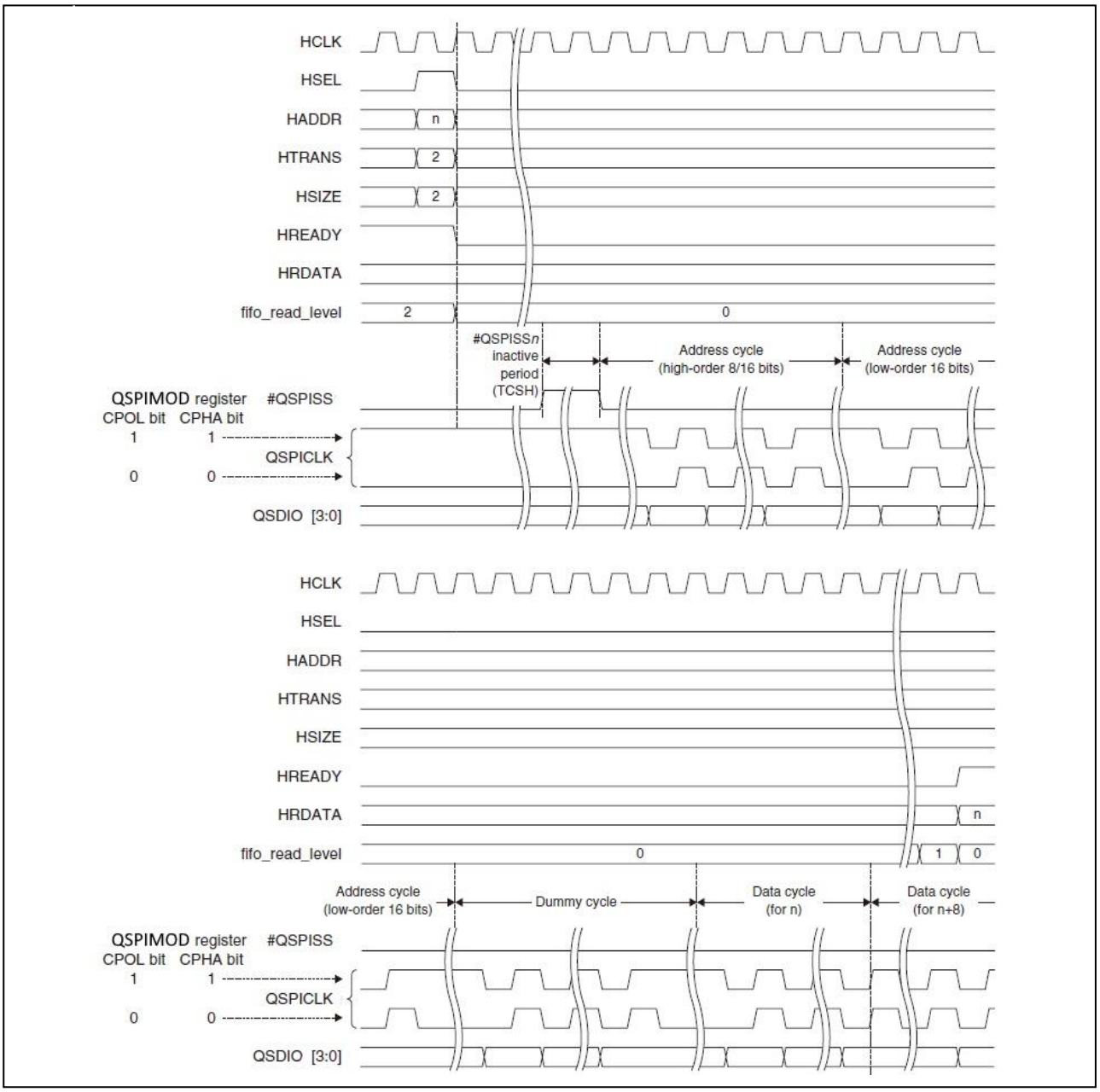


Figure 17.21 Data Receiving Operation in Memory Mapped Access Mode - 32-bit Non-Sequential Read

Data receiving operations (8/16-bit read)

The 8 and 16-bit read operations are the same as the 32-bit read operation except that data are not prefetched into the FIFO.

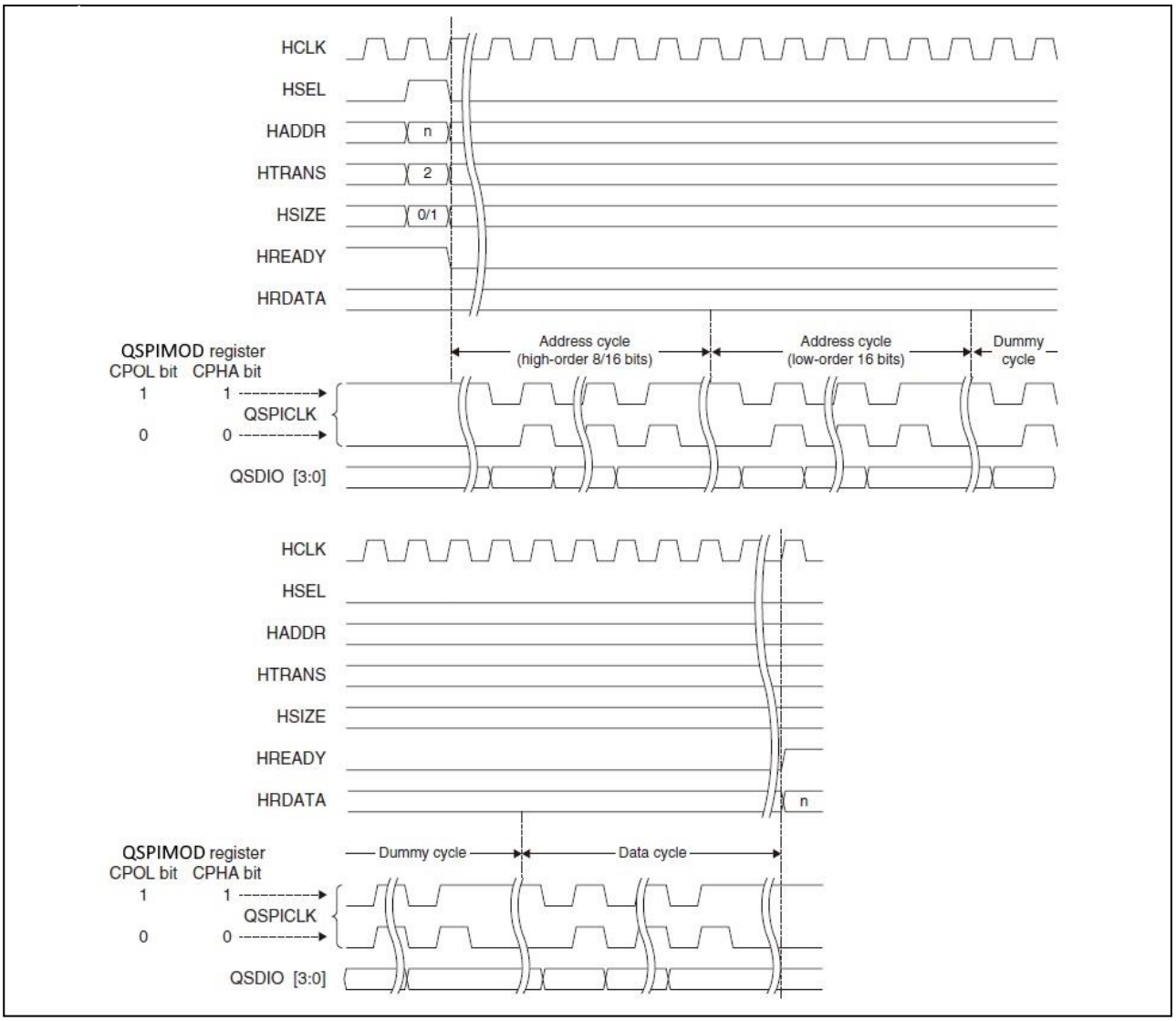


Figure 17.22 Data Receiving Operation in Memory Mapped Access Mode - First 8/16-bit Read

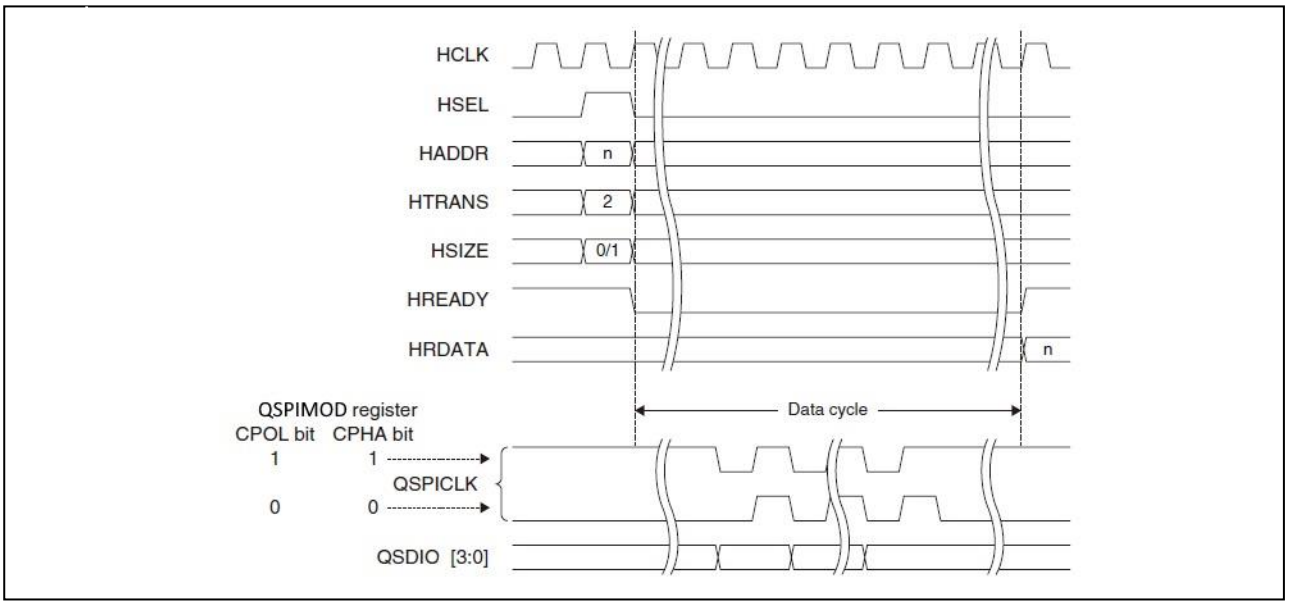


Figure 17.23 Data Receiving Operation in Memory Mapped Access Mode - 8/16-bit Sequential Read

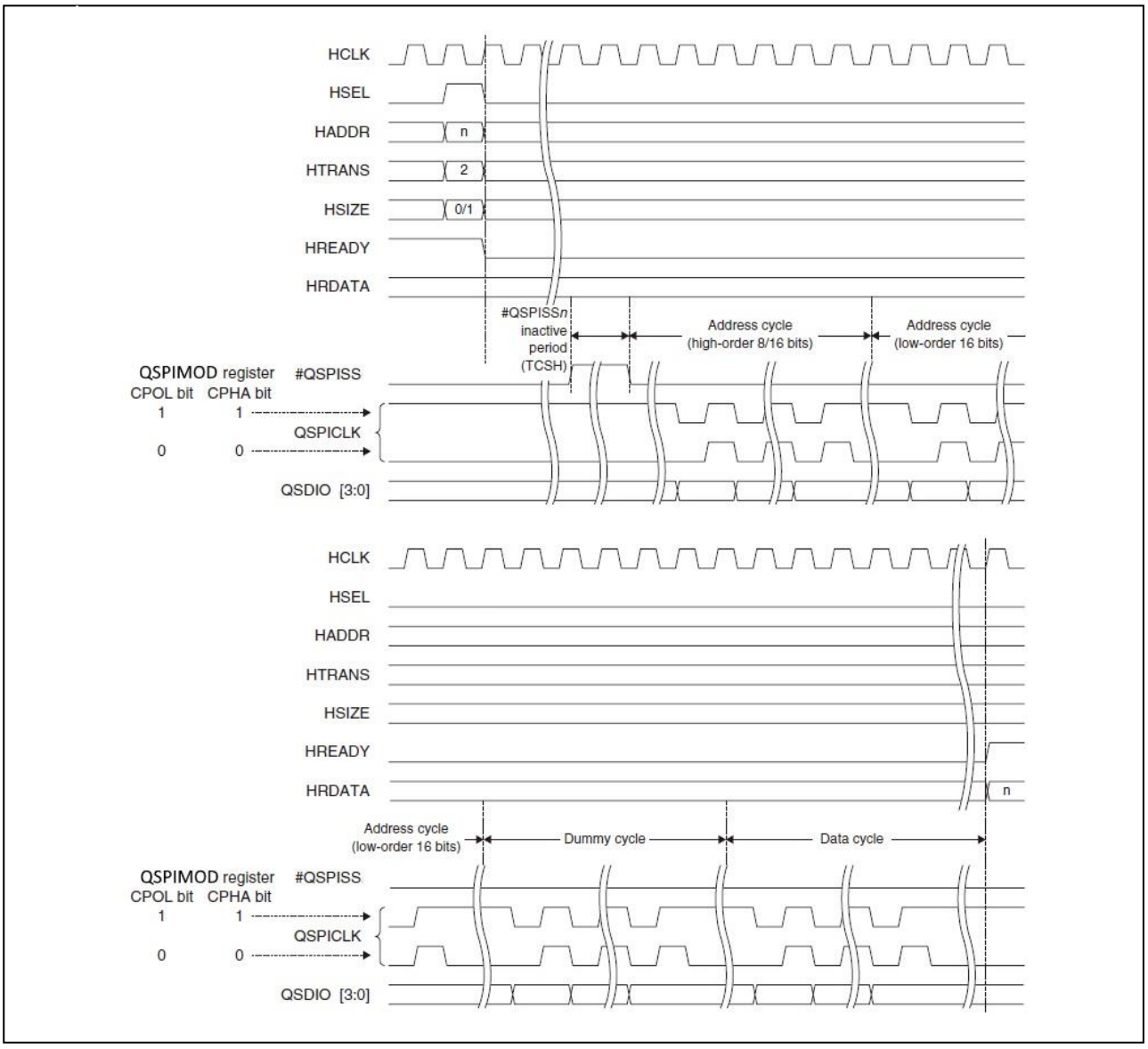


Figure 17.24 Data Receiving Operation in Memory Mapped Access Mode - 8/16-bit Non-Sequential Read

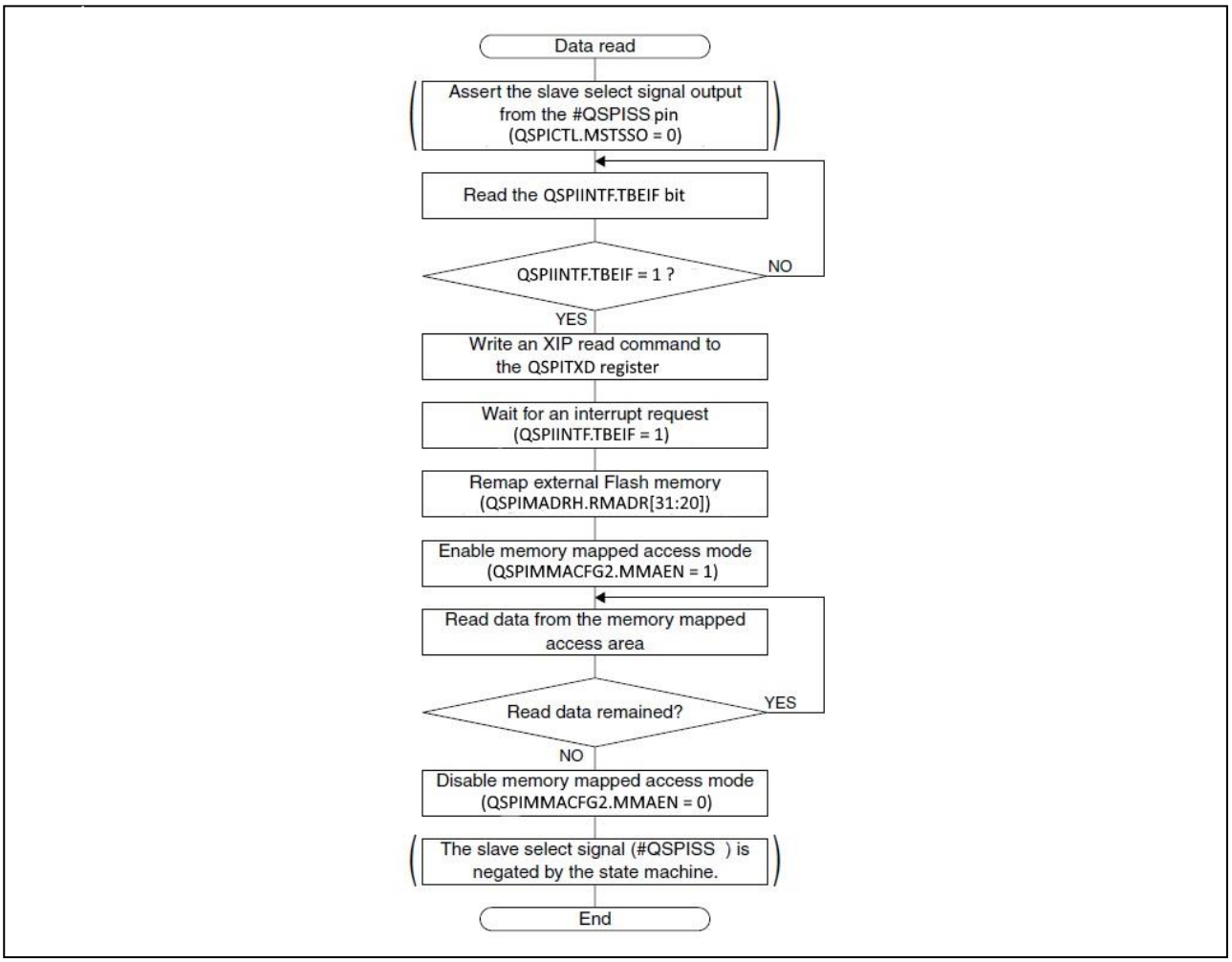


Figure 17.25 Data Reception Flowchart in Memory Mapped Access Mode

Data reception using DMA

In memory mapped access mode, DMA transfer from the external Flash memory to the internal memory is allowed only for the 32-bit sequential read using the internal FIFO. A non-sequential read and 8/16-bit reads cannot issue a DMA transfer request as they cannot use the FIFO.

By setting the QSPIFRLDMAEN.FRLDMAENx bit to 1 (DMA transfer request enabled), a DMA transfer request is sent to the DMA controller and the external Flash memory data is transferred to the specified internal memory via DMA Ch.x when the FIFO read level is incremented (FIFO data ready flag is set). This function allows high-speed data block transfer as it does not need to execute read commands and uses the data pre-fetched into the FIFO.

Note, however, that the first data read must be performed via software or a software triggered DMA.

The transfer source/destination and control data must be set for the DMA controller and the relevant DMA channel must be enabled to start a DMA transfer in advance. For more information on DMA, refer to the "DMA Controller" chapter.

Table 17.7 DMA Data Structure Configuration Example
(for 32-bit Sequential Read in Memory Mapped Access Mode)

Item		Setting example
End pointer	Transfer source	External Flash memory transfer start address
	Transfer destination	Memory area start address from which the read data are stored
Control data	dst_inc	0x2 (+4)
	dst_size	0x2 (word)
	src_inc	0x2 (+4)
	src_size	0x2 (word)
	R_power	0x0 (arbitrated for every transfer)
	n_minus_1	Number of receive data
	cycle_ctrl	0x1 (basic transfer)

The following shows an example of the control procedure including the DMA controller operations:
1. Configure the primary data structure for the DMA channel (Ch.x) as shown in Table 17.7.
2. Enable the DMA channel using the DMA controller register.
3. Clear the channel request mask for the DMA channel using the DMA controller register.
4. Clear the DMA transfer completion interrupt flag using the DMA controller register.
5. Enable the DMA transfer completion interrupt of the DMA channel using the DMA controller register.
6. Clear pending DMA interrupts in the host MCU.
7. Enable pending DMA interrupts in the host MCU.
8. Enable the QSPI to issue DMA transfer requests to the DMA channel using the QSPIFRLDMAEN.FRLDMAENx bit.
9. Issue a software DMA transfer request to the DMA channel by setting the DMA controller register. This operation is required to kickstart the first data fetching.
10. Wait for a DMA interrupt.
11. Disable DMA requests to be sent to the DMA channel using the QSPIFRLDMAEN.FRLDMAENx bit.
12. Set the channel request masks for the DMA channel using the DMA controller register.
13. Disable the DMA channels using the DMA controller register.

17.5.7 Terminating Memory Mapped Access Operations

A procedure to terminate memory mapped access operations is shown below.
1. Write 0 to the QSPIMMACFG2.MMAEN bit. (Disable memory mapped access mode)
   The slave select signal is negated. Note that the slave signal control via software is disabled by the state machine in memory mapped access mode.
2. Wait until the QSPIINTF.MMABSY bit is set to 0 (memory mapped access operation not busy).

17.5.8 Terminating Data Transfer in Master Mode

A procedure to terminate data transfer in master mode is shown below.
1. Wait for an end-of-transmission interrupt (QSPIINTF.TENDIF bit = 1).
2. Set the QSPICTL.MODEN bit to 0 to disable the QSPI operations.
3. Stop the 16-bit timer to disable the clock supply to QSPI.

17.5.9 Data Transfer in Slave Mode

A data sending/receiving procedure and operations in slave mode are shown below. Figure 17.26 and Figure 17.27 show a timing chart and flowcharts, respectively.

Data sending procedure

1. Check to see if the QSPIINTF.TBEIF bit is set to 1 (transmit buffer empty).
2. Write transmit data to the QSPITXD register.
3. Wait for a transmit buffer empty interrupt (QSPIINTF.TBEIF bit = 1).
4. Repeat Steps 2 and 3 until the end of transmit data.

Note: Transmit data must be written to the QSPInTXD register after the QSPIINTF.TBEIF bit is set to 1 by the time the sending QSPITXD register data written is completed. If no transmit data is written during this period, the data bits input from the QSDIO pins are shifted and output from the QSDIO pins without being modified.

Data receiving procedure

1. Wait for a receive buffer full interrupt (QSPIINTF.RBFIF bit = 1).
2. Read the received data from the QSPIRXD register.
3. Repeat Steps 1 and 2 until the end of data reception.

Data transfer operations

The following shows the slave mode operations different from master mode:

- Slave mode operates with the QSPI clock supplied from the external QSPI master to the QSPICLK pin. The data transfer rate is determined by the QSPICLK frequency. It is not necessary to control the 16-bit timer.
- QSPI can operate as a slave device only when the slave select signal input from the external QSPI master to the #QSPISS pin is set to the active (low) level. If #QSPISS = high, the software transfer control, the QSPICLK pin input, and the QSDIO pins input are all ineffective. If the #QSPISS signal goes high during data transfer, the transfer bit counter is cleared and data in the shift register is discarded.
- Slave mode starts data transfer when QSPICLK is input from the external QSPI master after the #QSPISS signal is asserted. Writing transmit data is not a trigger to start data transfer. Therefore, it is not necessary to write dummy data to the transmit data buffer when performing data reception only.

Other operations are the same as master mode.

NOTES:

- If data of the number of cycles specified by the QSPIMOD.CHLN[3:0] bits is received when the QSPIINTF.RBFIF bit is set to 1, the QSPI_nRXD register is overwritten with the newly received data and the previously received data is lost. In this case, the QSPIINTF.OEIF bit is set.
- When the clock for the first bit is input from the QSPICLK pin, QSPI starts sending the data currently stored in the shift register even if the QSPIINTF.TBEIF bit is set to 1.

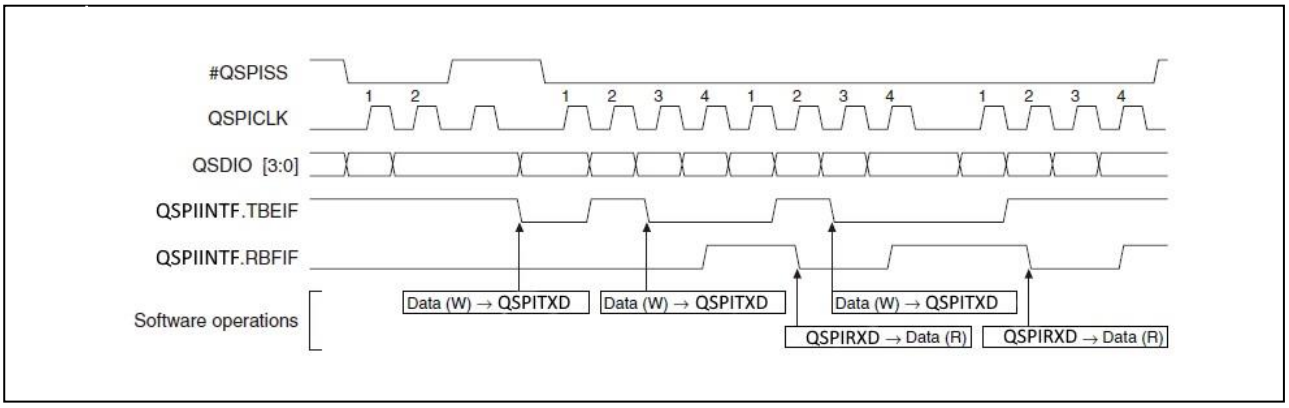


Figure 17.26 Example of Data Transfer Operations in Slave Mode
(QSPIMOD.CHDL[3:0] bits = QSPIMOD.CHLN[3:0] bits = 0x3)

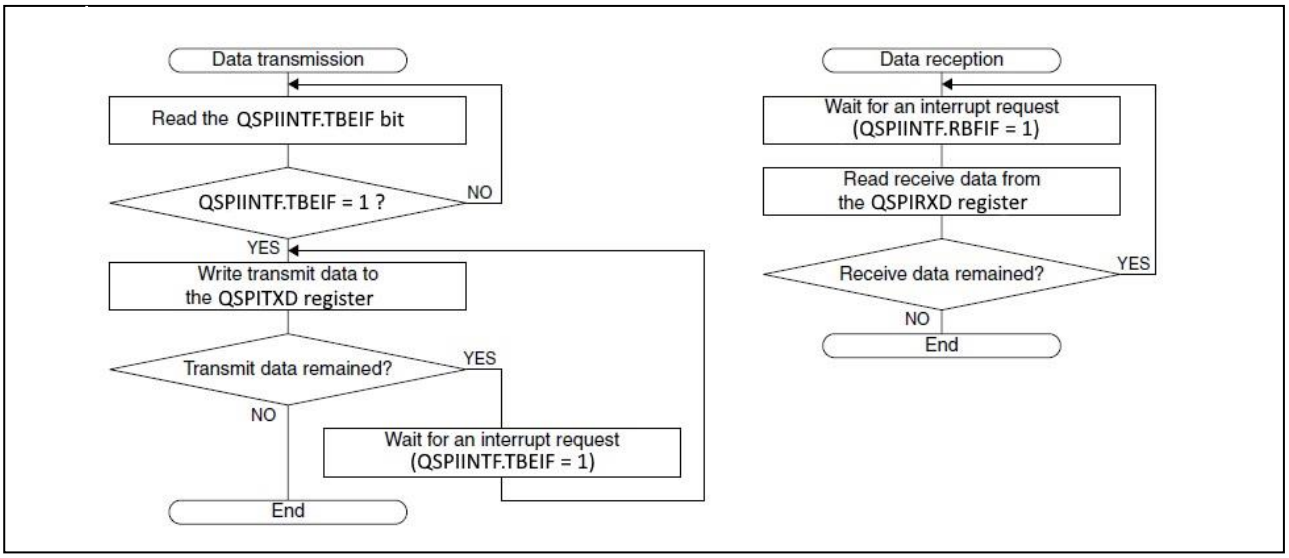


Figure 17.27 Data Transfer Flowcharts in Slave Mode

17.5.10 Terminating Data Transfer in Slave Mode

A procedure to terminate data transfer in slave mode is shown below.

1. Wait for an end-of-transmission interrupt (QSPIINTF.TENDIF bit = 1).
   Or determine end of transfer via the received data.
2. Set the QSPICTL.MODEN bit to 0 to disable the QSPI operations.

17.6 Interrupts

QSPI has a function to generate the interrupts shown in Table 17.8.

Table 17.8 QSPI Interrupt Function

Interrupt	Interrupt flag	Set condition	Clear condition
End of transmission	QSPIINTF.TENDIF	When the QSPIINTF.TBEIF bit = 1 after data of the specified bit length (defined by the QSPIMOD.CHLN[3:0] bits) has been sent	Writing 1
Receive buffer full	QSPIINTF.RBFIF	When data of the specified bit length is received and the received data is transferred from the shift register to the received data buffer	Reading the QSPIRXD register
Transmit buffer empty	QSPIINTF.TBEIF	When transmit data written to the transmit data buffer is transferred to the shift register	Writing to the QSPITXD register
Overrun error	QSPIINTF.OEIF	When the receive data buffer is full (when the received data has not been read) at the point that receiving data to the shift register has completed	Writing 1

The QSPI provides interrupt enable bits corresponding to each interrupt flag. An interrupt request is sent to the host MCU only when the interrupt flag, of which interrupt has been enabled by the interrupt enable bit, is set. The QSPIINTF register also contains the BSY and MMABSY bits that indicate the QSPI operating status in register access and memory mapped access modes, respectively. Figure 17.29 shows the QSPIINTF.BSY, QSPIINTF.MMABSY and QSPIINTF.TENDIF bit set timings.

Figure 17.28 shows a diagram of the QSPIINT interrupt signal. The QSPIINT signal is logical ORed with interrupt signals from other peripherals to generate the HIFIRQ interrupt signal to the host MCU.
See Section 22 (“Interrupts”) for more details on the HIFIRQ output.

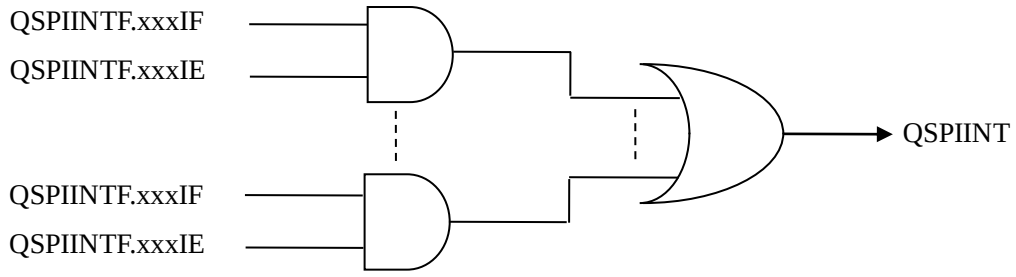


Figure 17.28 QSPIINT Interrupt Circuit

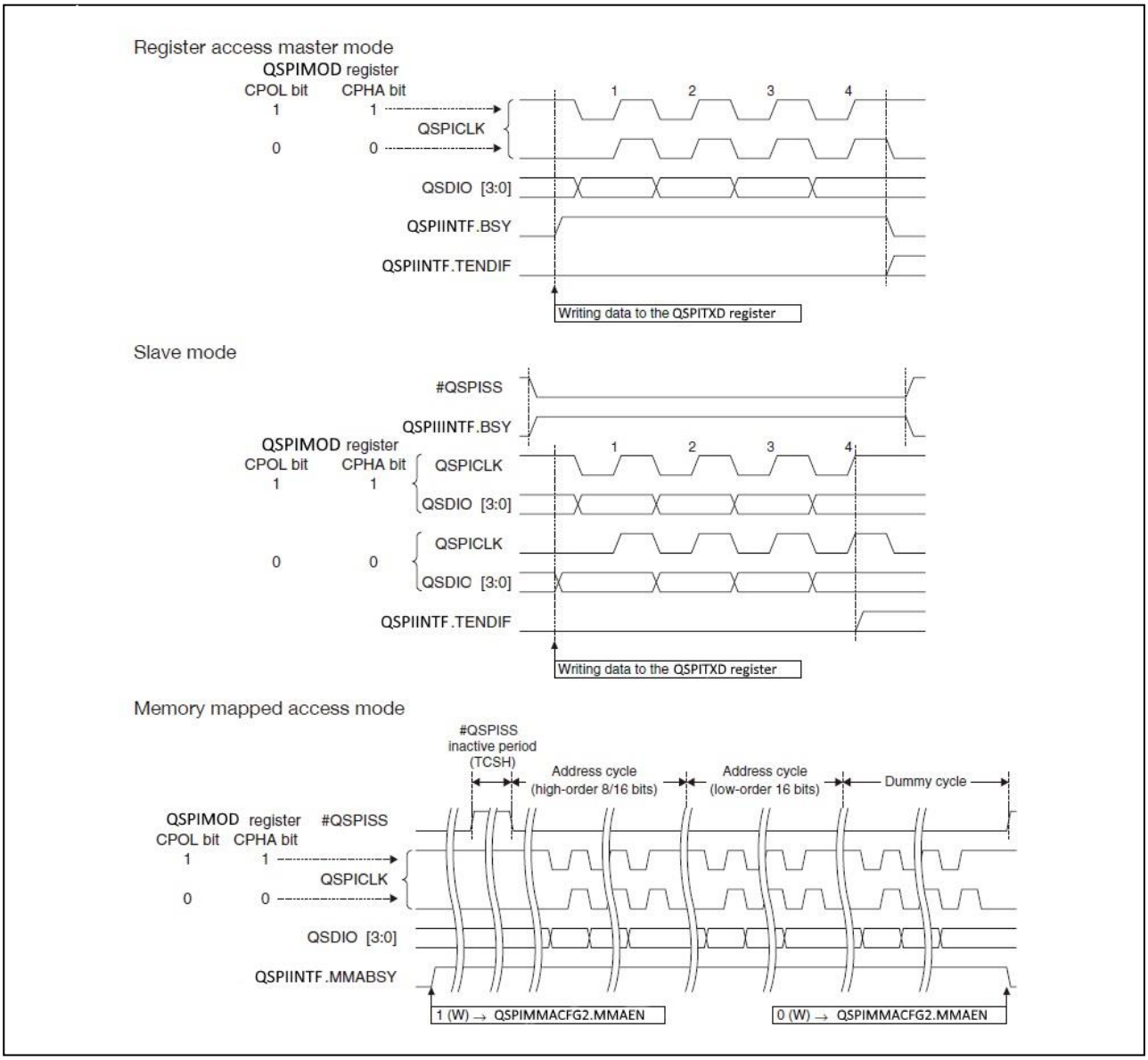


Figure 17.29 QSPIINTF.BSY, QSPIINTF.MMABSY, and QSPIINTF.TENDIF Bit Set Timings (when QSPIMOD.CHDL[3:0] bits = QSPIMOD.CHLN[3:0] bits = 0x3)

17.7 DMA Transfer Requests

The QSPI has a function to generate DMA transfer requests from the causes shown in Table 17.9.

Table 17.9 DMA Transfer Request Causes of QSPI

Cause to request DMA transfer	DMA transfer request flag	Set condition	Clear condition
Receive buffer full	Receive buffer full flag (QSPIINTF.RBFIF)	When data of the specified bit length is received and the received data is transferred from the shift register to the received data buffer	Reading the QSPIRXD register
Transmit buffer empty	Transmit buffer empty flag (QSPIINTF.TBEIF)	When transmit data written to the transmit data buffer is transferred to the shift register	Writing to the QSPITXD register
Memory mapped access FIFO data ready	Memory mapped access FIFO data ready flag (internal signal)	When a 32-bit is prefetched into the FIFO in memory mapped access mode	When the FIFO read level is cleared to 0

The QSPI provides DMA transfer request enable bits corresponding to each DMA transfer request flag shown above for the number of DMA channels. A DMA transfer request is sent to the pertinent channel of the DMA controller only when the DMA transfer request flag, of which DMA transfer has been enabled by the DMA transfer request enable bit, is set. The receive buffer full and transmit buffer empty DMA transfer request flags also serve as interrupt flags, therefore, both the DMA transfer request and the interrupt cannot be enabled at the same time. After a DMA transfer has completed, disable the DMA transfer to prevent unintended DMA transfer requests from being issued. For more information on the DMA control, refer to the "DMA Controller" chapter.

17.8 Control Registers

See Section 10.7 for descriptions of the control registers for QSPI.

18. Sound Generator (SND)

18.1 Overview

SND is a sound generator that generates melodies and buzzer signals. The features of the SND are listed below.

- Sound output mode is selectable from three types.
  1. Normal buzzer mode (for normal buzzer output of which the output duration is controlled via software)
     - Output frequency: Can be set within the range of 512 Hz to 16,384 Hz.
     - Duty ratio: Can be set within the range of 0 % to 100 %.
  2. One-shot buzzer mode (for short buzzer output such as a clicking sound)
     - Output frequency: Can be set within the range of 512 Hz to 16,384 Hz.
     - Duty ratio: Can be set within the range of 0 % to 100 %.
     - One-shot output duration: Can be set within the range of 15.6 ms to 250 ms. (16 types)
  3. Melody mode (for playing single note melody)
     - Pitch: Can be set within the range of 128 Hz to 16,384 Hz. (Scale: 3 octave from C3 to C6 with reference to A4 = 443 Hz)
     - Duration: Can be set within the range of half note/rest to thirty-second note/rest. (7 types)
     - Tempo: Can be set within the range of 30 to 480. (16 types)
     - Other: Tie and slur can be specified.
- A piezoelectric buzzer can be driven with the inverted and non-inverted output pins.
- Can control the non-inverted output pin status while sound stops.

Figure 18.1 shows the SND configuration.

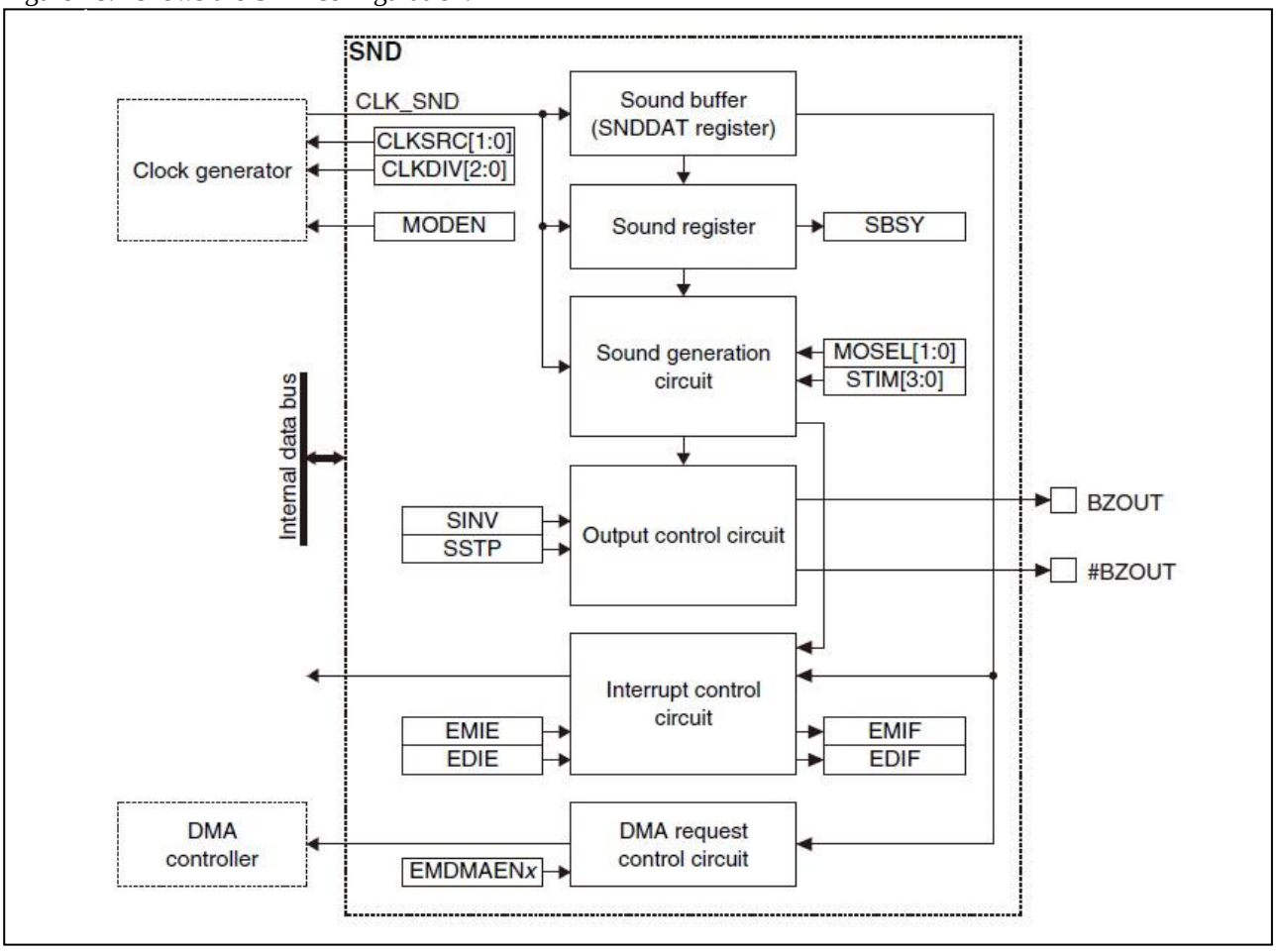


Figure 18.1 SND Configuration

18.2 Input/Output Pins and External Connections

18.2.1 List of Input/Output Pins

Table 18.1 lists the SND pins.

Table 18.1 List of SND Pins

Pin name	I/O*	Initial status*	Function
BZOUT	O	O (Low)	Non-inverted buzzer output pin
#BZOUT	O	O (Low)	Inverted buzzer output pin

* Indicates the status when the pin is configured for the SND.

If the port is shared with the SND pin and other functions, the SND output function must be assigned to the port before activating the SND. For more information, refer to the "GPIO Ports" chapter.

18.2.2 Output Pin Drive Mode

The drive mode of the BZOUT and #BZOUT pins can be set to one of the two types shown below using the SNDSEL.SINV bit.

Direct drive mode (SNDSEL.SINV bit = 0)

This mode drives both the BZOUT and #BZOUT pins to low while the buzzer signal output is off to prevent the piezoelectric buzzer from applying unnecessary bias.

Normal drive mode (SNDSEL.SINV bit = 1)

In this mode, the #BZOUT pin always outputs the inverted signal of the BZOUT pin even when the buzzer output is off.

18.2.3 External Connections

Figure 18.2 and Figure 18.3 show connection diagrams between SND and a piezoelectric buzzer.

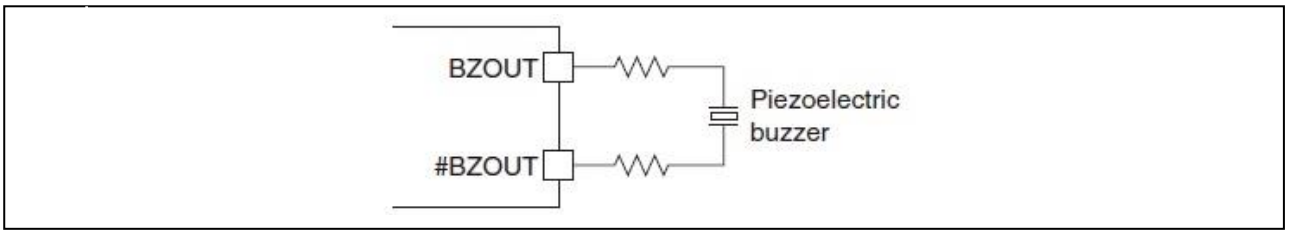


Figure 18.2 Connection between SND and Piezoelectric Buzzer (Direct Drive)

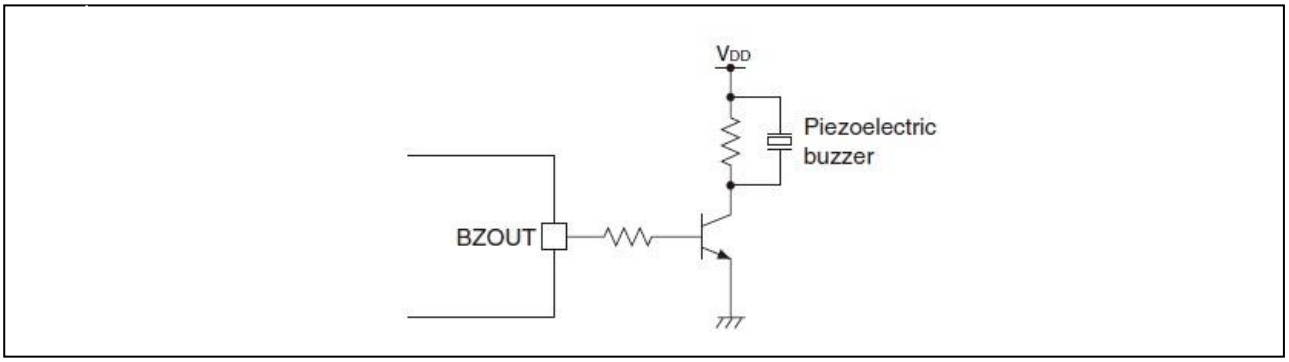


Figure 18.3 Connection between SND and Piezoelectric Buzzer (Single Pin Drive)

18.3 Clock Settings

18.3.1 SND Operating Clock

When using SND, the SND operating clock CLK_SND must be supplied to SND from the clock generator. The CLK_SND supply should be controlled as in the procedure shown below.

1. Enable the clock source in the clock generator if it is stopped.
2. Set the following SNDCLK register bits:
   - SNDCLK.CLKSRC[1:0] bits (Clock source selection)
   - SNDCLK.CLKDIV[2:0] bits (Clock division ratio selection = Clock frequency setting)

The CLK_SND frequency should be set to around 32,768 Hz.

18.4 Operations

18.4.1 Initialization

The SND should be initialized with the procedure shown below.

1. Assign the SND output function to the ports. (Refer to the "GPIO Ports" chapter.)
2. Configure the SND operating clock.
3. Set the SNDCTL.MODEN bit to 1. (Enable SND operations)
4. Set the following SNDSEL register bits:
   - Set the SNDSEL.SINV bit (Set output pin drive mode)
   - Set the SNDSEL.MOSEL[1:0] bits (Set sound output mode)
5. Set the following bits when using the interrupt:
   - Write 1 to the interrupt flags in the SNDINTF register. (Clear interrupt flags)
   - Set the interrupt enable bits in the SNDINTE register to 1. (Enable interrupts)
6. Configure the DMA controller and set the following SNDA control bits when using DMA transfer:
   - Write 1 to the DMA transfer request enable bits in the SNDEMDMAEN register. (Enable DMA transfer requests)

18.4.2 Buzzer Output in Normal Buzzer Mode

Normal buzzer mode generates a buzzer signal with the software specified frequency and duty ratio, and outputs the generated signal to outside the IC. The buzzer output duration can also be controlled via software. An output start/stop procedure and the SND operations are shown below.

Normal buzzer output start/stop procedure

1. Set the SNDSEL.MOSEL[1:0] bits to 0x0. (Set normal buzzer mode)
2. Write data to the following sound buffer (SNDDAT register) bits. (Start buzzer output)
   - SNDDAT.SLEN[5:0] bits (Set buzzer output signal duty ratio)
   - SNDDAT.SFRQ[7:0] bits (Set buzzer output signal frequency)
3. Write 1 to the SNDCTL.SSTP bit after the output period has elapsed. (Stop buzzer output)

Normal buzzer output operations

When data is written to the sound buffer (SNDDAT register), SND clears the SNDINTF.EMIF bit (sound buffer empty interrupt flag) to 0 and starts buzzer output operations.
The data written to the sound buffer is loaded into the sound register in sync with the CLK_SND clock. At the same time, the SNDINTF.EMIF bit and SNDINTF.SBSY bit are both set to 1. The output pin outputs the buzzer signal with the frequency/duty ratio specified.
Writing 1 to the SNDCTL.SSTP bit stops buzzer output and sets the SNDINTF.EDIF bit (sound output completion interrupt flag) to 1. The SNDINTF.SBSY bit is cleared to 0.
Figure 18.4 shows a buzzer output timing chart in normal buzzer mode.

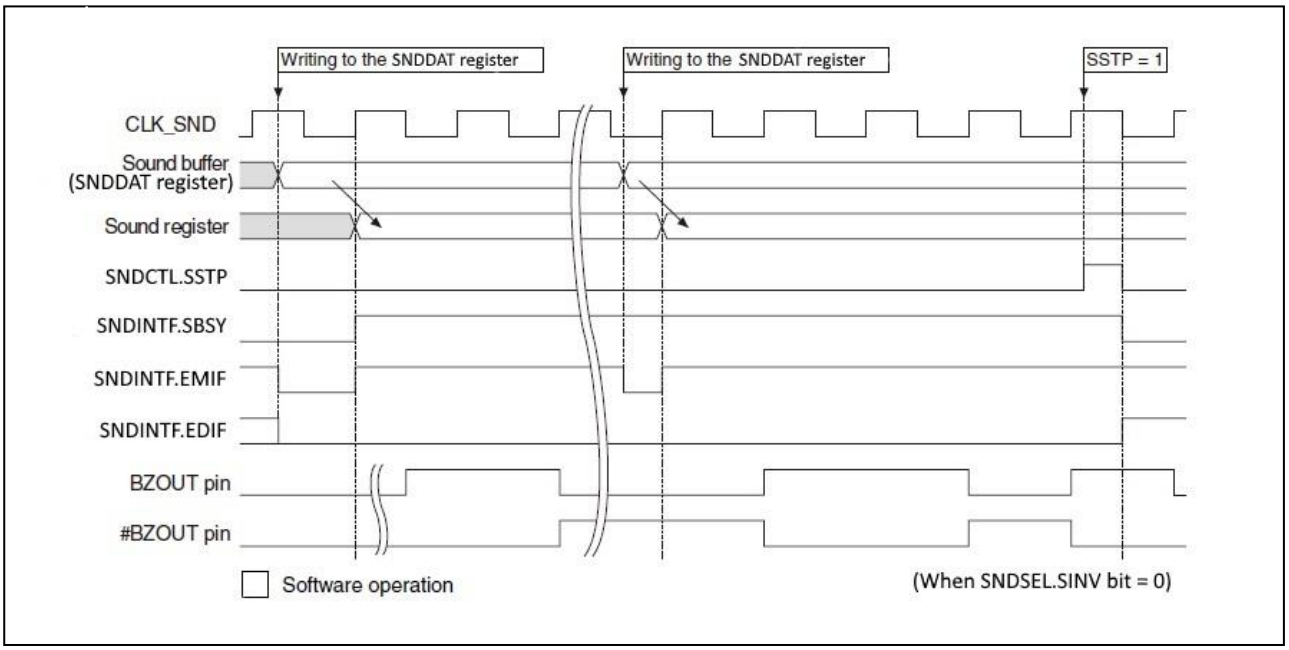


Figure 18.4 Buzzer Output Timing Chart in Normal Buzzer Mode

Buzzer output waveform configuration (normal buzzer mode/one-shot buzzer mode)

Set the buzzer signal frequency and duty ratio (high period/cycle) using the SNDDAT.SFRQ[7:0] and SNDDAT.SLEN[5:0] bits, respectively. Use the following equations to calculate these setting values.

SNDDAT.SFRQ[7:0] bits = (f_{CLK_SND} / f_{BZOUT}) – 1
SNDDAT.SLEN[5:0] bits = [(f_{CLK_SND} / f_{BZOUT}) x (DUTY / 100)] – 1

Where:

f_{CLK_SND}:	CLK_SND frequency [Hz]
f_{BZOUT}:	Buzzer signal frequency [Hz]
DUTY:	Buzzer signal duty ratio [%]

However, the following settings are prohibited:

- Settings as SNDDAT.SFRQ[7:0] bits ≤ SNDDAT.SLEN[5:0] bits
- Settings as SNDDAT.SFRQ[7:0] bits = 0x00

Table 18.2 Buzzer Frequency Settings (when f_{CLK_SND} = 32,768 Hz)

SNDDAT. SFRQ[7:0]	Frequency (Hz)	SNDDAT. SFRQ[7:0]	Frequency (Hz)	SNDDAT. SFRQ[7:0]	Frequency (Hz)	SNDDAT. SFRQ[7:0]	Frequency (Hz)
0x3F	512.0	0x2F	682.7	0x1F	1,024.0	0x0F	2,048.0
0x3E	520.1	0x2E	697.2	0x1E	1,057.0	0x0E	2,184.5
0x3D	528.5	0x2D	712.3	0x1D	1,092.3	0x0D	2,340.6
0x3C	537.2	0x2C	728.2	0x1C	1,129.9	0x0C	2,520.6
0x3B	546.1	0x2B	744.7	0x1B	1,170.3	0x0B	2,730.7
0x3A	555.4	0x2A	762.0	0x1A	1,213.6	0x0A	2,978,9
0x39	565.0	0x29	780.2	0x19	1,260,3	0x09	3,276.8
0x38	574.9	0x28	799.2	0x18	1,310.7	0x08	3,640.9
0x37	585.1	0x27	819.2	0x17	1,365.3	0x07	4,096.0
0x36	595.8	0x26	840.2	0x16	1,424.7	0x06	4,681.1
0x35	606.8	0x25	862.3	0x15	1,489.5	0x05	5,461.3
0x34	618.3	0x24	885.6	0x14	1,560.4	0x04	6,553.6
0x33	630.2	0x23	910.2	0x13	1,638.4	0x03	8,192.0
0x32	642.5	0x22	936.2	0x12	1,724.6	0x02	10,922.7
0x31	655.4	0x21	963.8	0x11	1,820.4	0x01	16,384.0
0x30	668.7	0x20	993.0	0x10	1,927.5	0x00	Cannot be set

Table 18.3 Buzzer Duty Ratio Setting Examples (when f_{CLK_SND} = 32,768 Hz)

SNDDAT. SLEN[5:0]	Duty ratio by buzzer frequency					
	16,384 Hz	8,192 Hz	4,096 Hz	2,048 Hz	1,024 Hz	512 Hz
0x3F	-	-	-	-	-	-
0x3E	-	-	-	-	-	98.4
0x3D	-	-	-	-	-	96.9
0x3C	-	-	-	-	-	95.3
0x3B	-	-	-	-	-	93.8
0x3A	-	-	-	-	-	92.2
0x39	-	-	-	-	-	90.6
0x38	-	-	-	-	-	89.1
0x37	-	-	-	-	-	87.5
0x36	-	-	-	-	-	85.9
0x35	-	-	-	-	-	84.4
0x34	-	-	-	-	-	82.8
0x33	-	-	-	-	-	81.3
0x32	-	-	-	-	-	79.7
0x31	-	-	-	-	-	78.1
0x30	-	-	-	-	-	76.6
0x2F	-	-	-	-	-	75.0
0x2E	-	-	-	-	-	73.4
0x2D	-	-	-	-	-	71.9
0x2C	-	-	-	-	-	70.3
0x2B	-	-	-	-	-	68.8
0x2A	-	-	-	-	-	67.2

0x29	-	-	-	-	-	65.6
0x28	-	-	-	-	-	64.1
0x27	-	-	-	-	-	62.5
0x26	-	-	-	-	-	60.9
0x25	-	-	-	-	-	59.4
0x24	-	-	-	-	-	57.8
0x23	-	-	-	-	-	56.3
0x22	-	-	-	-	-	54.7
0x21	-	-	-	-	-	53.1
0x20	-	-	-	-	-	51.6
0x1F	-	-	-	-	-	50.0
0x1E	-	-	-	-	96.9	48.4
0x1D	-	-	-	-	93.8	46.9
0x1C	-	-	-	-	90.6	45.3
0x1B	-	-	-	-	87.5	43.8
0x1A	-	-	-	-	84.4	42.2
0x19	-	-	-	-	81.3	40.6
0x18	-	-	-	-	78.1	39.1
0x17	-	-	-	-	75.0	37.5
0x16	-	-	-	-	71.9	35.9
0x15	-	-	-	-	68.8	34.4
0x14	-	-	-	-	65.6	32.8
0x13	-	-	-	-	62.5	31.3
0x12	-	-	-	-	59.4	29.7
0x11	-	-	-	-	56.3	28.1
0x10	-	-	-	-	53.1	26.6
0x0F	-	-	-	-	50.0	25.0
0x0E	-	-	-	93.8	46.9	23.4
0x0D	-	-	-	87.5	43.8	21.9
0x0C	-	-	-	81.3	40.6	20.3
0x0B	-	-	-	75.0	37.5	18.8
0x0A	-	-	-	68.8	34.4	17.2
0x09	-	-	-	62.5	31.3	15.6
0x08	-	-	-	56.3	28.1	14.1
0x07	-	-	-	50.0	25.0	12.5
0x06	-	-	87.5	43.8	21.9	10.9
0x05	-	-	75.0	37.5	18.8	9.4
0x04	-	-	62.5	31.3	15.6	7.8
0x03	-	-	50.0	25.0	12.5	6.3
0x02	-	75.0	37.5	18.8	9.4	4.7
0x01	-	50.0	25.0	12.5	6.3	3.1
0x00	50.0	25.0	12.5	6.3	3.1	1.6

18.4.3 Buzzer Output in One-shot Buzzer Mode

One-shot buzzer mode is provided for clicking sound and short-duration buzzer output. This mode generates a buzzer signal with the software specified frequency and duty ratio, and outputs the generated signal for the short duration specified.
An output start procedure and the SND operations are shown below. For the buzzer output waveform, refer to "Buzzer Output in Normal Buzzer Mode."

One-shot buzzer output start procedure

1. Set the following SNDSEL register bits:
   - Set the SNDSEL.MOSEL[1:0] bits to 0x1. (Set one-shot buzzer mode)
   - SNDSEL.STIM[3:0] bits (Set output duration)
2. Write data to the following sound buffer (SNDDAT register) bits. (Start buzzer output)
   - SNDDAT.SLEN[5:0] bits (Set buzzer output signal duty ratio)
   - SNDDAT.SFRQ[7:0] bits (Set buzzer output signal frequency)

One-shot buzzer output operations

When data is written to the sound buffer (SNDDAT register), SND clears the SNDINTF.EMIF bit (sound buffer empty interrupt flag) to 0 and starts buzzer output operations.
The data written to the sound buffer is loaded into the sound register in sync with the CLK_SND clock. At the same time, the SNDINTF.EMIF bit and SNDINTF.SBSY bit are both set to 1. The output pin outputs the buzzer signal with the frequency/duty ratio specified.
The buzzer output automatically stops when the duration specified by the SNDSEL.STIM[3:0] bits has elapsed. At the same time, the SNDINTF.EDIF bit (sound output completion interrupt flag) is set to 1 and the SNDINTF.SBSY bit is cleared to 0.
Figure 18.5 shows a buzzer output timing chart in one-shot buzzer mode.

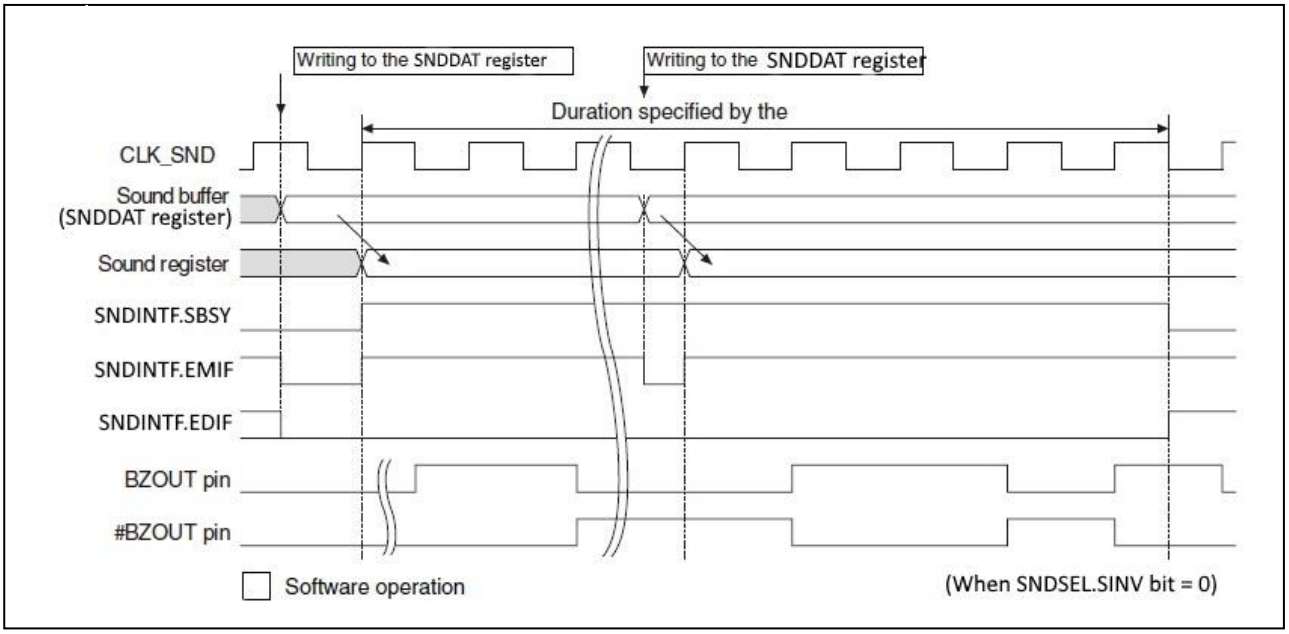


Figure 18.5 Buzzer Output Timing Chart in One-shot Buzzer Mode

18.4.4 Output in Melody Mode

Melody mode generates the buzzer signal with a melody according to the data written to the sound buffer (SNDDAT register) successively, and outputs the generated signal to outside the IC. An output start procedure and the SND operations are shown below.

Melody output start procedure

1. Set the following SNDSEL register bits:
   - Set the SNDSEL.MOSEL[1:0] bits to 0x2. (Set melody mode)
   - SNDSEL.STIM[3:0] bits (Set tempo)
2. Write data to the following sound buffer (SNDDAT register) bits. (Start sound output)
   - SNDDAT.MDTI bit (Set tie/slur)
   - SNDDAT.MDRS bit (Set note/rest)
   - SNDDAT.SLEN[5:0] bits (Set duration)
   - SNDDAT.SFRQ[7:0] bits (Set scale)
3. Check to see if the SNDINTF.EMIF bit is set to 1 (an interrupt can be used).
4. Repeat Steps 2 and 3 until the end of the melody.

Melody output operations

When data is written to the sound buffer (SNDDAT register), SND clears the SNDINTF.EMIF bit (sound buffer empty interrupt flag) to 0 and starts sound output operations.
The data written to the sound buffer is loaded into the sound register by the internal trigger signal. At the same time, the SNDINTF.EMIF bit and SNDINTF.SBSY bit are both set to 1. The output pin outputs the sound specified.
The sound output stops if data is not written to the sound buffer (SNDDAT register) until the next trigger is issued. At the same time, the SNDINTF.EDIF bit (sound output completion interrupt flag) is set to 1 and the SNDINTF.SBSY bit is cleared to 0.
Figure 18.6 shows a melody mode operation timing chart.

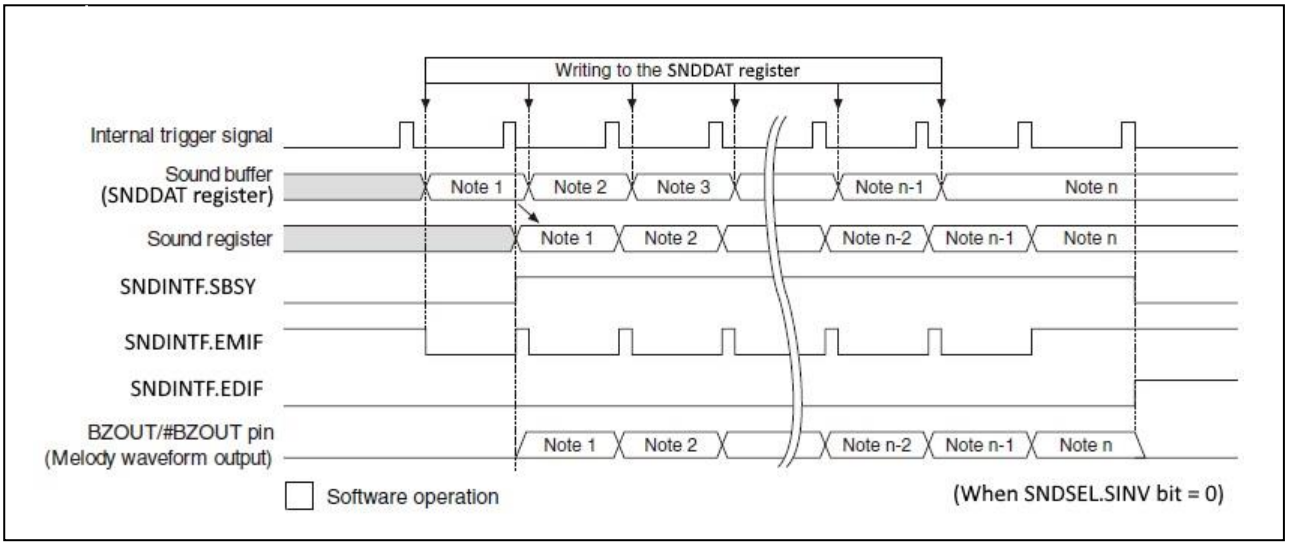


Figure 18.6 Melody Mode Operation Timing Chart

Melody output using DMA

By setting the SNDEMDMAEN.EMDMAENx bit to 1 (DMA transfer request enabled), a DMA transfer request is sent to the DMA controller and melody data is transferred from the specified memory to the sound buffer (SNDDAT register) via DMA Ch.x when the SNDINTF.EMIF bit is set to 1 (sound buffer empty). This automates the melody output procedure from Steps 2 to 4 described above.
The transfer source/destination and control data must be set for the DMA controller and the relevant DMA channel must be enabled to start a DMA transfer in advance so that transmit data will be transferred to the sound buffer (SNDDAT register). For more information on DMA, refer to the "DMA Controller" chapter.

Table 18.4 DMA Data Structure Configuration Example (for Melody Output)

Item		Setting example
End pointer	Transfer source	Memory address in which the last melody data is stored
	Transfer destination	SNDDAT register address
Control data	dst_inc	0x3 (no increment)
	dst_size	0x1 (halfword)
	src_inc	0x1 (+2)
	src_size	0x1 (halfword)
	R_power	0x0 (arbitrated for every transfer)
	n_minus_1	Number of transfer data
	cycle_ctrl	0x1 (basic transfer)

Melody output waveform configuration

Note/rest (duration) specification

Notes and rests can be specified using the SNDDAT.MDRS and SNDDAT.SLEN[5:0] bits.

Table 18.5 Note/Rest Specification (when f_{CLK_SND} = 32,768 Hz)

SNDDAT.SLEN[5:0] bits	SNDDAT.MDRS bit	
	0: Note	1: Rest
0x0F	Half note	Half rest
0x0B	Dotted quarter note	Dotted quarter rest
0x07	Quarter note	Quarter rest
0x05	Dotted eighth note	Dotted eighth rest
0x03	Eighth note	Eighth rest
0x01	Sixteenth note	Sixteenth rest
0x00	Thirty-second note	Thirty-second rest
Other	Setting prohibited	

Tie/slur specification

A tie or slur takes effect by setting the SNDDAT.MDTI bit to 1 and the previous note and the current note are played continuously.

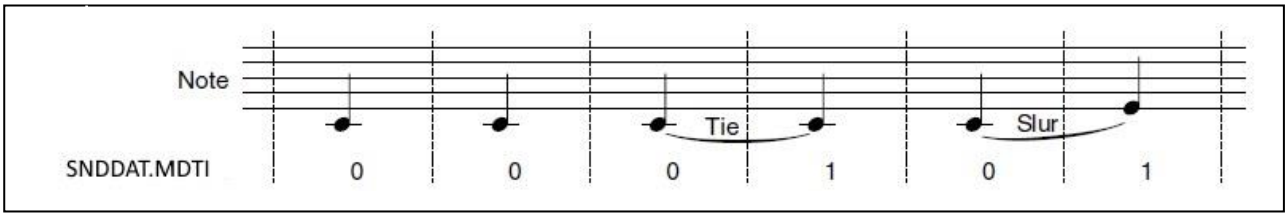


Figure 18.7 Tie and Slur

Scale specification

Scales can be specified using the SNDDAT.SFRQ[7:0] bits.

Table 18.6 Scale Specification (when f_{CLK_SND} = 32,768 Hz)

SNDDAT.SFRQ[7:0]	Scale	Frequency (Hz)
0xF8	C3	131.60
0xEA	C#3	139.44
0xDD	D3	147.60
0xD1	D#3	156.04
0xC5	E3	165.49
0xBA	F3	175.23
0xAF	F#3	186.18
0xA5	G3	197.40
0x9C	G#3	208.71
0x93	A3	221.41
0x8B	A#3	234.06
0x83	B3	248.24
0x7C	C4	262.14
0x75	C#4	277.69
0x6E	D4	295.21
0x68	D#4	312.08
0x62	E4	330.99
0x5C	F4	352.34
0x57	F#4	372.36
0x52	G4	394.80
0x4E	G#4	414.78
0x49	A4	442.81
0x45	A#4	468.11
0x41	B4	496.48
0x3D	C5	528.52
0x3A	C#5	555.39
0x37	D5	585.14
0x33	D#5	630.15
0x30	E5	668.73
0x2E	F5	697.19
0x2B	F#5	744.73
0x29	G5	780.19
0x26	G#5	840.21
0x24	A5	885.62
0x22	A#5	936.23
0x20	B5	992.97
0x1E	C6	1057.03

18.5 Interrupts

SND has a function to generate the interrupts shown in Table 18.7.

Table 18.7 SND Interrupt Function

Interrupt	Interrupt flag	Set condition	Clear condition
Sound buffer empty	SNDINTF.EMIF	When data in the sound buffer (SNDDAT register) is transferred to the sound register or 1 is written to the SNDCTL.SSTP bit	Writing to the SNDDAT register
Sound output completion	SNDINTF.EDIF	When a sound output has completed	Writing 1 or writing to the SNDDAT register

SND provides interrupt enable bits corresponding to each interrupt flag. An interrupt request is sent to the host MCU only when the interrupt flag, of which interrupt has been enabled by the interrupt enable bit, is set.

Figure 18.8 shows a diagram of the SNDINT interrupt signal. The SNDINT signal is logical ORed with interrupt signals from other peripherals to generate the HIFIRQ interrupt signal to the host MCU.
See Section 22 (“Interrupts”) for more details on the HIFIRQ output.

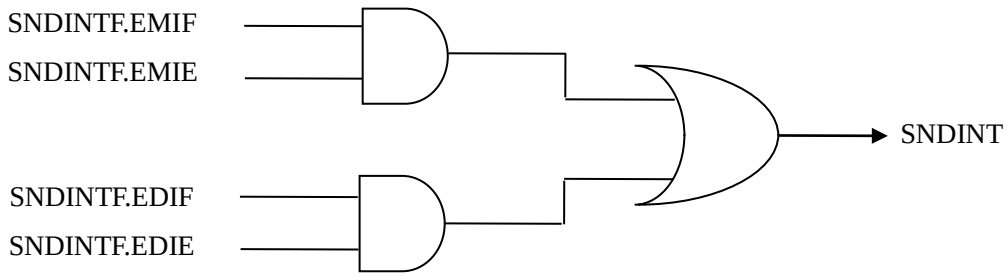


Figure 18.8 SNDINT Interrupt Circuit

18.6 DMA Transfer Requests

The SND has a function to generate DMA transfer requests from the causes shown in Table 18.8.

Table 18.8 DMA Transfer Request Causes of SND

Cause to request DMA transfer	DMA transfer request flag	Set condition	Clear condition
Sound buffer empty	Sound buffer empty flag (SNDINTF.EMIF)	When data in the sound buffer (SNDDAT register) is transferred to the sound register or 1 is written to the SNDCTL.SSTP bit	Writing to the SNDDAT register

The SND provides DMA transfer request enable bits corresponding to each DMA transfer request flag shown above for the number of DMA channels. A DMA transfer request is sent to the pertinent channel of the DMA controller only when the DMA transfer request flag, of which DMA transfer has been enabled by the DMA transfer request enable bit, is set. The DMA transfer request flag also serves as an interrupt flag, therefore, both the DMA transfer request and the interrupt cannot be enabled at the same time. After a DMA transfer has completed, disable the DMA transfer to prevent unintended DMA transfer requests from being issued. For more information on the DMA control, refer to the “DMA Controller” chapter.

18.7 Control Registers

See Section 10.8 for descriptions of the control registers for SND.

19. IR Remote Control Transmitter (REMC)

19.1 Overview

The REMC circuit generates infrared remote control output signals. This circuit can also be applicable to an EL lamp drive circuit by adding a simple external circuit.
The features of the REMC are listed below.

- Outputs an infrared remote control signal.
- Includes a carrier generator.
- Flexible carrier signal generation and data pulse width modulation.
- Automatic data setting function for continuous data transmission.
- Output signal inverting function supporting various formats.
- EL lamp drive waveform can be generated for an application example.

Figure 19.1 shows the REMC configuration.

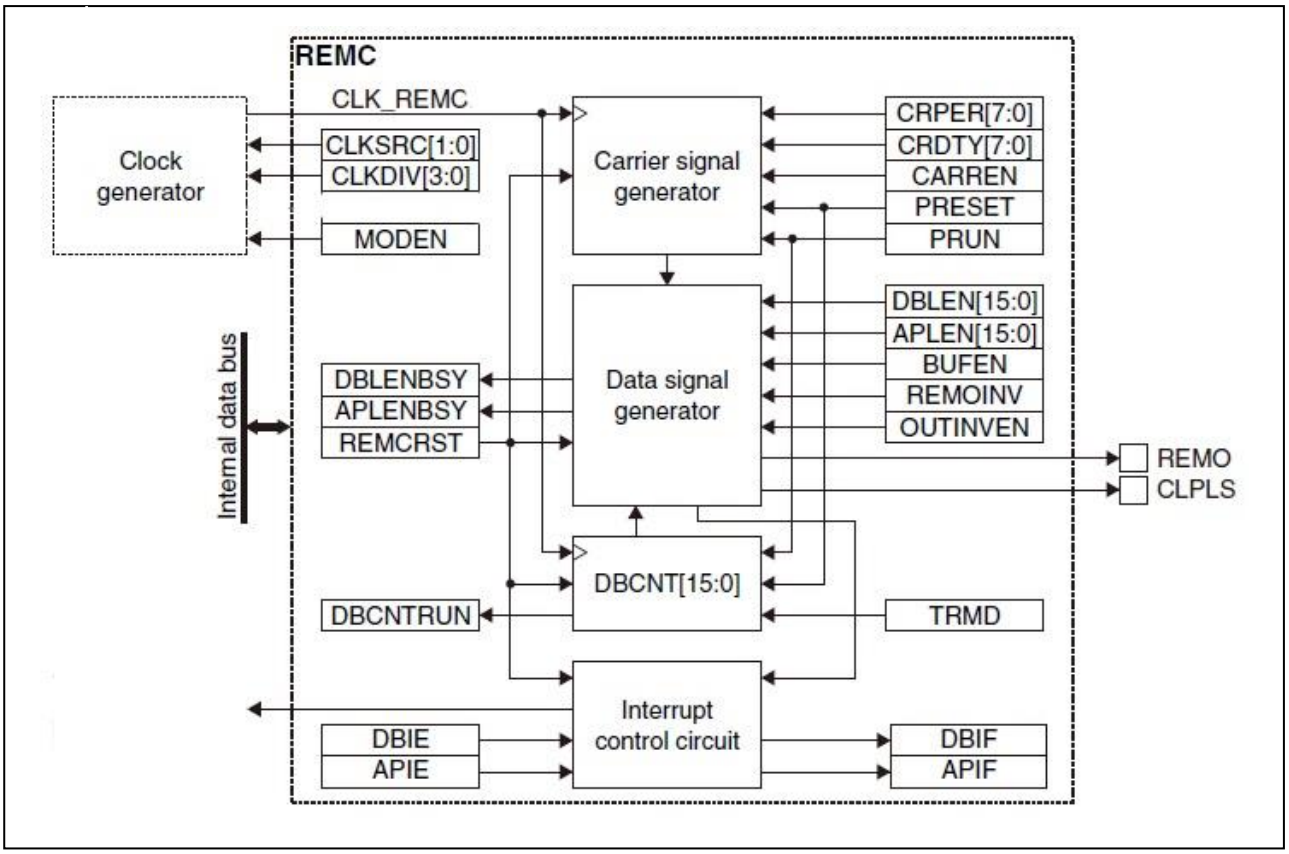


Figure 19.1 REMC Configuration

19.2 Input/Output Pins and External Connections

19.2.1 List of Input/Output Pins

Table 19.1 lists the REMC pins.

Table 19.1 List of REMC Pins

Pin name	I/O*	Initial status*	Function
REMO	O	O (Low)	IR remote controller transmit data output
CLPLS	O	O (Low)	IR remote controller clear pulse output

* Indicates the status when the pin is configured for the REMC.

If the port is shared with the REMC pin and other functions, the REMC output function must be assigned to the port before activating the REMC. For more information, refer to the "GPIO Ports" chapter.

19.2.2 External Connections

Figure 19.2 shows a connection example between the REMC and an external infrared module.

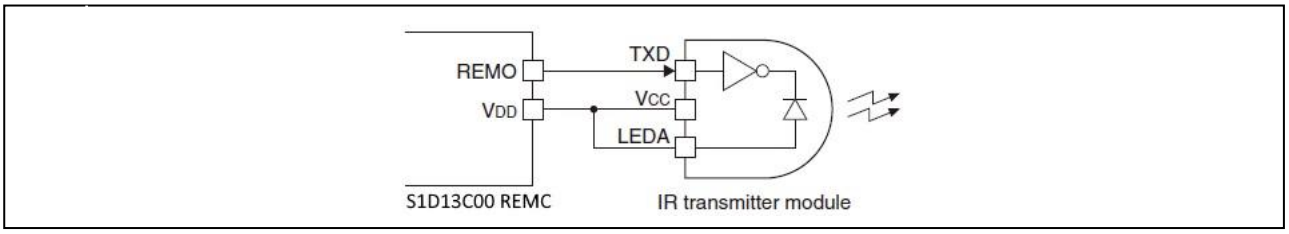


Figure 19.2 Connection Example Between REMC and External Infrared Module

19.3 Clock Settings

19.3.1 REMC Operating Clock

When using the REMC, the REMC operating clock CLK_REMC must be supplied to the REMC from the clock generator. The CLK_REMC supply should be controlled as in the procedure shown below.

1. Enable the clock source in the clock generator if it is stopped.
2. Set the following REMCCLK register bits:
   - REMCCLK.CLKSRC[1:0] bits (Clock source selection)
   - REMCCLK.CLKDIV[3:0] bits (Clock division ratio selection = Clock frequency setting)

19.4 Operations

19.4.1 Initialization

The REMC should be initialized with the procedure shown below.

1. Write 1 to the REMCDBCTL.REMCRST bit. (Reset REMC)
2. Configure the REMCCLK.CLKSRC[1:0] and REMCCLK.CLKDIV[3:0] bits. (Configure operating clock)
3. Assign the REMC output function to the port. (Refer to the "GPIO Ports" chapter.)
4. Configure the following REMCDBCTL register bits:
   - Set the REMCDBCTL.MODEN bit to 1. (Enable count operation clock)
   - REMCDBCTL.TRMD bit (Select repeat mode/one-shot mode)
   - Set the REMCDBCTL.BUFEN bit to 1. (Enable compare buffer)
   - REMCDBCTL.REMOINV bit (Configure inverse logic output signal)

5. Configure the following REMCCARR register bits:
   - REMCCARR.CRPER[7:0] bit (Set carrier signal cycle)
   - REMCCARR.CRDTY[7:0] bit (Set carrier signal duty)
6. Configure the following REMCCCTL register bits:
   - REMCCCTL.CARREN bit (Enable/disable carrier modulation)
   - REMCCCTL.OUTINVEN bit (Configure output signal polarity)
7. Set the following bits when using the interrupt:
   - Write 1 to the interrupt flags in the REMCINTF register. (Clear interrupt flags)
   - Set the interrupt enable bits in the REMCINTE register to 1. (Enable interrupts)

19.4.2 Data Transmission Procedures

Starting data transmission

The following shows a procedure to start data transmission.

1. Set the REMCAPLEN.APLEN[15:0] bits. (Set data signal duty)
2. Set the REMCDBLEN.DBLEN[15:0] bits. (Set data signal cycle)
3. Set the following REMC3DBCTL register bits:
   - Set the REMCDBCTL.PRESET bit to 1. (Reset internal counters)
   - Set the REMCDBCTL.PRUN bit to 1. (Start counting)

Continuous data transmission control

The following shows a procedure to send data continuously after starting data transmission (after Step 3 above).

1. Set the duty and cycle for the subsequent data to the REMCAPLEN.APLEN[15:0] and REMCDBLEN.DBLEN[15:0] bits, respectively, before a compare DB interrupt (REMCINTF.DBIF bit = 1) occurs. (It is not necessary to rewrite settings when sending the same data with the current settings.)
2. Wait for a compare DB interrupt (REMCINTF.DBIF bit = 1).
3. Repeat Steps 1 and 2 until the end of data.

Terminating data transmission

The following shows a procedure to terminate data transmission.

1. Wait for a compare DB interrupt. (REMCINTF.DBIF bit = 1).
2. Set the REMCDBCTL.PRUN bit to 0. (Stop counting)
3. Set the REMCDBCTL.MODEN bit to 0. (Disable count operation clock)

19.4.3 REMO Output Waveform

Carrier refers to infrared frequency in infrared remote control communication. Note, however, that carrier in this manual refers to sub-carrier used in infrared remote control communication, as REMC does not control infrared rays directly.

The REMC outputs the logical AND between the carrier signal output from the carrier generator and the data signal output from the data signal generator. Figure 19.3 shows an example of the output waveform.

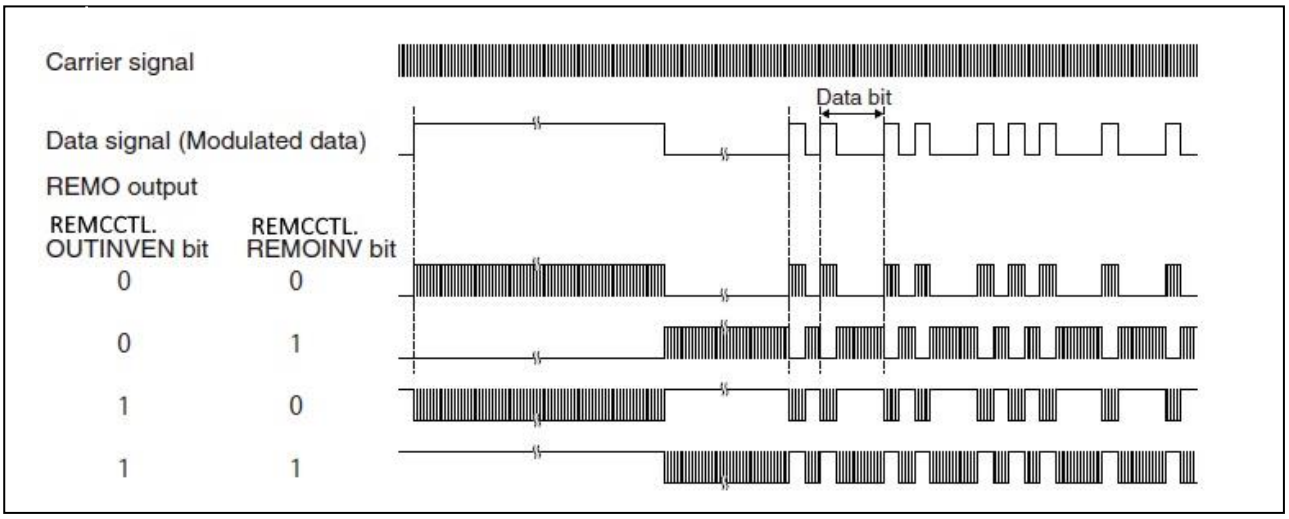


Figure 19.3 REMO Output Waveform Example

Carrier signal

The carrier signal is generated by comparing the values of the 8-bit counter for carrier generation that runs with CLK_REMC and the setting values of the REMCCARR.CRDTY[7:0] and REMCCARR.CRPER[7:0] bits. Figure 19.4 shows an example of the carrier signal generated.

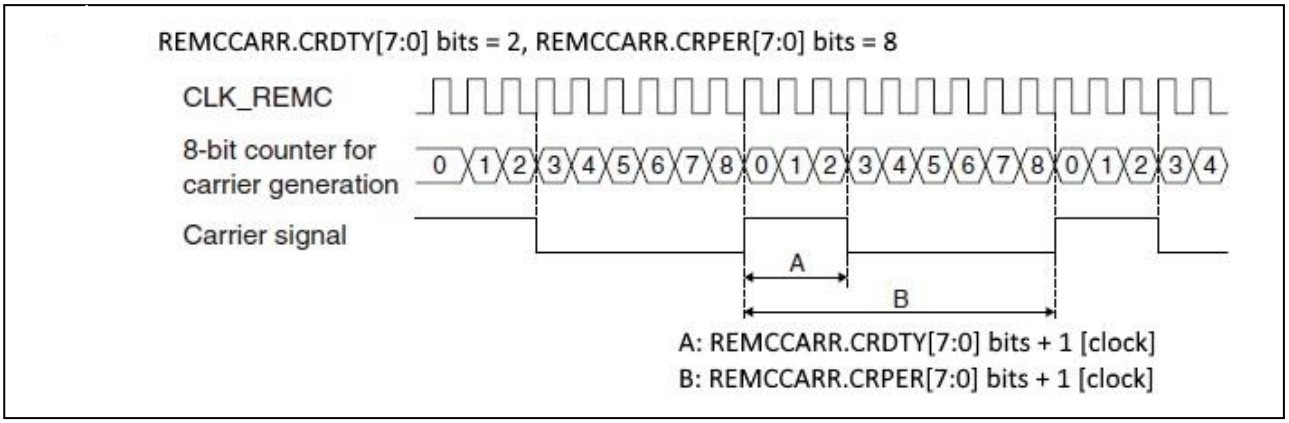


Figure 19.4 Example of Carrier Signal Generated

The carrier signal frequency and duty ratio can be calculated by the equations shown below.

Carrier frequency = f_{CLK_REMC} / (CRPER + 1)
Duty ratio = (CRDTY + 1) / (CRPER + 1)

Where,
f_{CLK_REMC}: CLK_REMC frequency [Hz]
CRPER: REMCCARR.CRPER[7:0] bit-setting value (1-255)
CRDTY: REMCCARR.CRDTY[7:0] bit-setting value (0-254)
* REMCCARR.CRDTY[7:0] bits < REMCCARR.CRPER[7:0] bits

The 8-bit counter for carrier generation is reset by the REMCDBCTL.PRESET bit and is started/stopped by the REMCDBCTL.PRUN bit in conjunction with the 16-bit counter for data signal generation. When the counter value is matched with the REMCCARR.CRDTY[7:0] bits, the carrier signal waveform is inverted. When the counter value is matched with the REMCCARR.CRPER[7:0] bits, the carrier signal waveform is inverted and the counter is reset to 0x00.

Data signal

The data signal is generated by comparing the values of the 16-bit counter for data signal generation (REMCDBCNT.DBCNT[15:0] bits) that runs with CLK_REMC and the setting values of the REMCAPLEN.APLEN[15:0] and REMCDBLEN.DBLEN[15:0] bits. Figure 19.5 shows an example of the data signal generated.

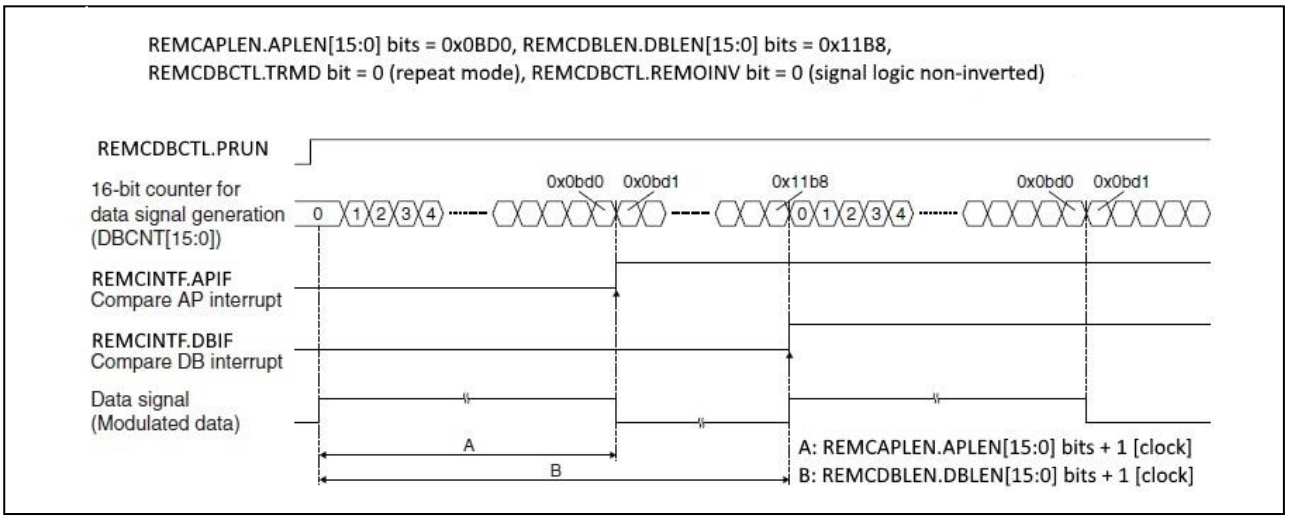


Figure 19.5 Example of Data Signal Generated

The data length and duty ratio of the pulse-width-modulated data signal can be calculated with the equations shown below.

Data length = (DBLEN + 1) / f_{CLK_REMC}
Duty ratio = (APLEN + 1) / (DBLEN + 1)

Where,
f_{CLK_REMC}: CLK_REMC frequency [Hz]
DBLEN: REMCDBLEN.DBLEN[15:0] bit-setting value (1-65,535)
APLEN: REMCAPLEN.APLEN[15:0] bit-setting value (0-65,534)
* REMCAPLEN.APLEN[15:0] bits < REMCDBLEN.DBLEN[15:0] bits

The 16-bit counter for data signal generation is reset by the REMCDBCTL.PRESET bit and is started/stopped by the REMCDBCTL.PRUN bit. When the counter value is matched with the REMCAPLEN.APLEN[15:0] bits (compare AP), the data signal waveform is inverted. When the counter value is matched with the REMCDBLEN.DBLEN[15:0] bits (compare DB), the data signal waveform is inverted and the counter is reset to 0x0000.
A different interrupt can be generated when the counter value is matched with the REMCDBLEN.DBLEN[15:0] and REMCAPLEN.APLEN[15:0] bits, respectively.

Repeat mode and one-shot mode

When the 16-bit counter for data signal generation is set to repeat mode (REMCDBCTL.TRMD bit = 0), the counter keeps operating until it is stopped using the REMC3DBCTL.PRUN bit. When the counter is set to one-shot mode (REMCDBCTL.TRMD bit = 1), the counter stops automatically when the counter value is matched with the REMCDBLEN.DBLEN[15:0] bit-setting value.

19.4.4 Continuous Data Transmission and Compare Buffers

Figure 19.6 shows an operation example of continuous data transmission with the compare buffer enabled.

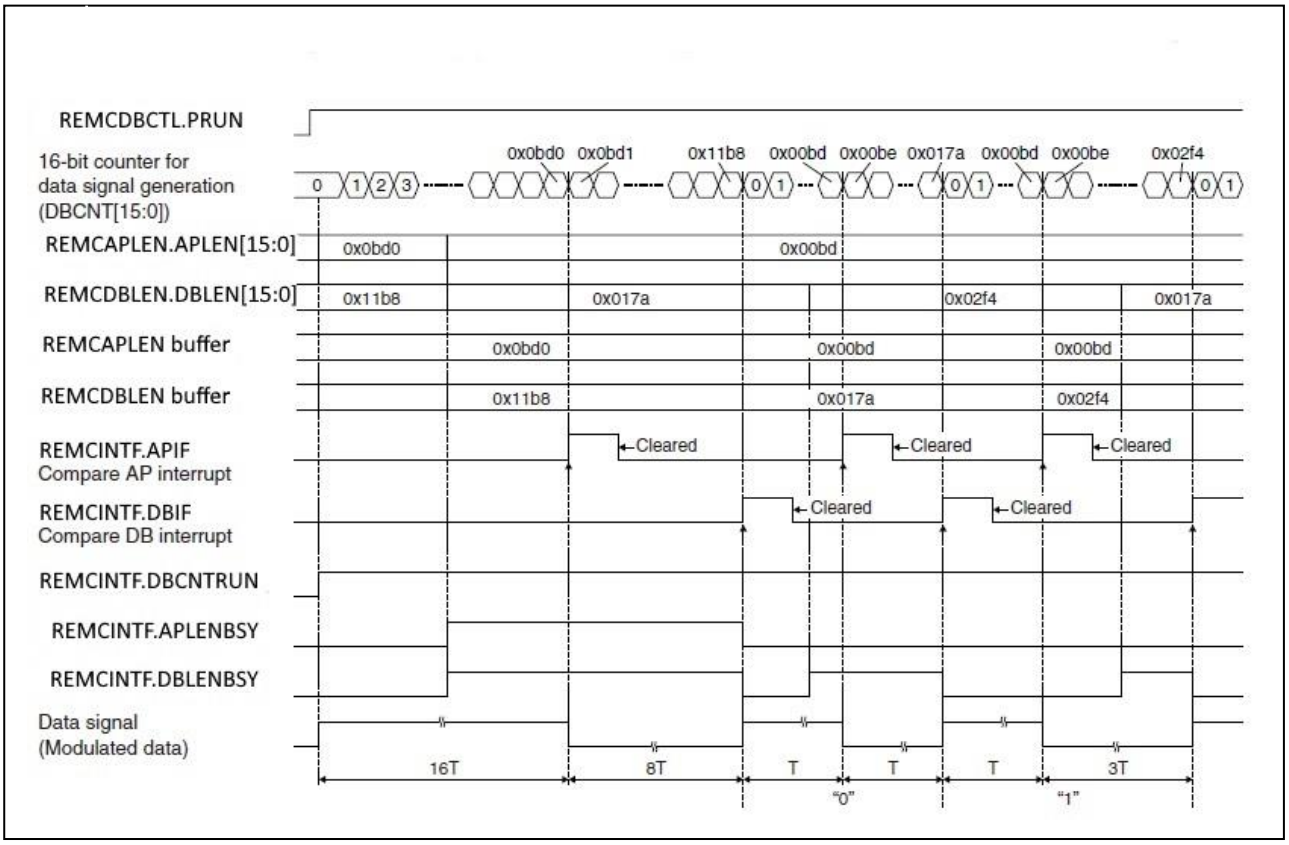


Figure 19.6 Continuous Data Transmission Example

When the compare buffer is disabled (REMCDBCTL.BUFEN bit = 0), the 16-bit counter value is directly compared with the REMCAPLEN.APLEN[15:0] and REMCDBLEN.DBLEN[15:0] bit values. The comparison value is altered immediately after the REMCAPLEN.APLEN[15:0] or REMCDBLEN.DBLEN[15:0] bits are rewritten.
When the compare buffer is enabled (REMCDBCTL.BUFEN bit = 1), the REMCAPLEN.APLEN[15:0] and REMCDBLEN.DBLEN[15:0] bit values are loaded into the compare buffers provided respectively (REMCAPLEN buffer and REMCDBLEN buffer) and the 16-bit counter value is compared with the compare buffers. The comparison values are loaded into the compare buffers when the 16-bit counter is matched with the REMCDBLEN buffer (when the count for the data length has completed). Therefore, the next transmit data can be set during the current data transmission. When the compare buffers are enabled, the buffer status flags (REMCINTF.APLENBSY bit and REMCINTF.DBLENBSY bit) become effective. The flag is set to 1 when the setting value is written to the register and cleared to 0 when the written value is transferred to the buffer.

19.5 Interrupts

The REMC has a function to generate the interrupts shown in Table 19.2.

Table 19.2 REMC Interrupt Function

Interrupt	Interrupt flag	Set condition	Clear condition
Compare AP	REMCINTF.APIF	When the REMCAPLEN register (or REMCAPLEN buffer) value and the 16-bit counter for data signal generation are matched	Writing 1 to the interrupt flag or the REMCDBCTL.REMCRST bit
Compare DB	REMCINTF.DBIF	When the REMCDBLEN register (or REMCDBLEN buffer) value and the 16-bit counter for data signal generation are matched	Writing 1 to the interrupt flag or the REMCDBCTL.REMCRST bit

The REMC provides interrupt enable bits corresponding to each interrupt flag. An interrupt request is sent to the host MCU only when the interrupt flag, of which interrupt has been enabled by the interrupt enable bit, is set. Figure 19.7 shows a diagram of the REMCINT interrupt signal. The REMCINT signal is logical ORed with interrupt signals from other peripherals to generate the HIFIRQ interrupt signal to the host MCU.
See Section 22 ("Interrupts") for more details on the HIFIRQ output.

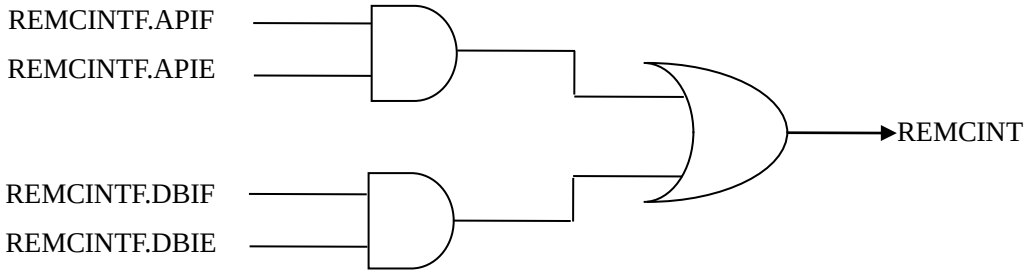


Figure 19.7 REMCINT Interrupt Circuit

19.6 Application Example: Driving EL Lamp

The REMC can be used to simply drive an EL lamp as an application example. Figure 19.8 and Figure 19.9 show an example of an EL lamp drive circuit and an example of the drive waveform generated, respectively.

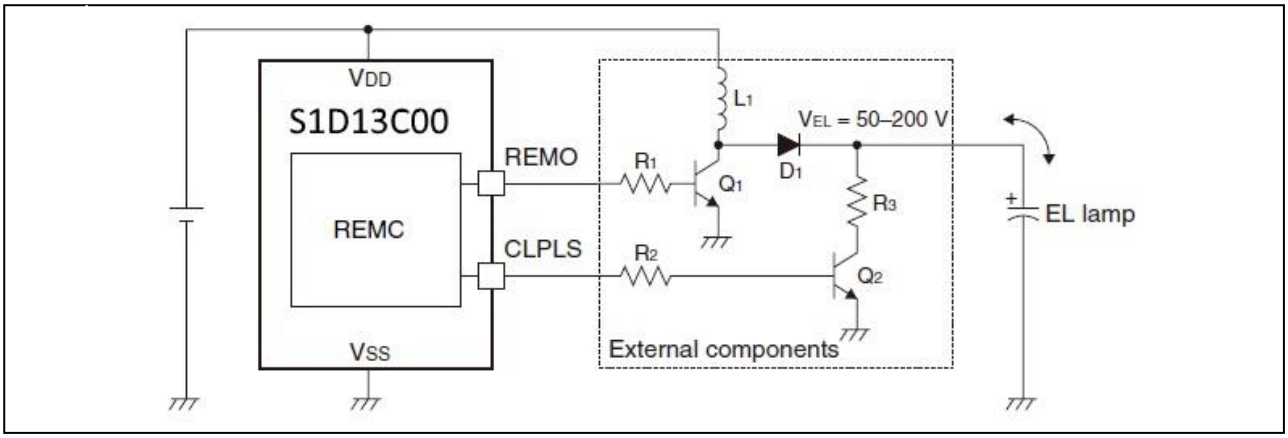


Figure 19.8 Example of EL Lamp Drive Circuit

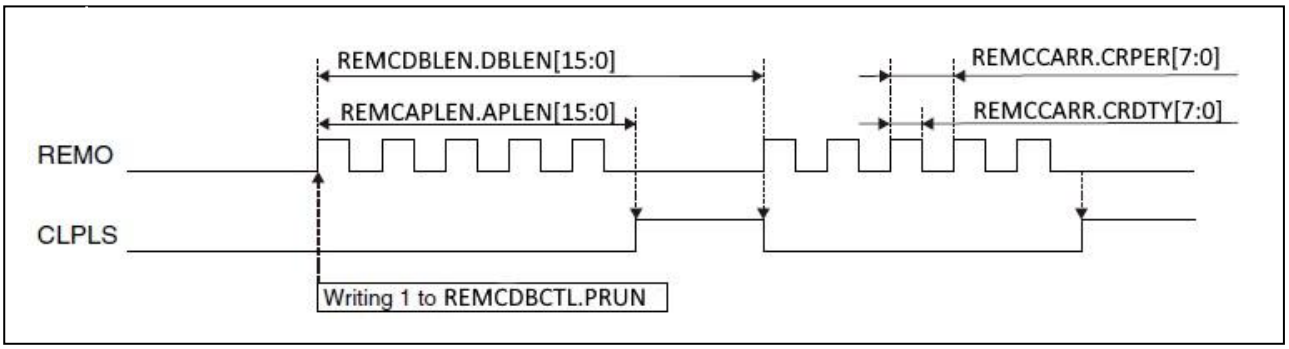


Figure 19.9 Example of Generated Drive Waveform

The REMO and CLPLS signals are output from the respective pins while the REMCDBCTL.PRUN bit = 1. The difference between the setting values of the REMCDBLEN.DBLEN[15:0] bits and REMCAPLEN.APLEN[15:0] bits becomes the CLPLS pulse width (high period).

19.7 Control Registers

See Section 10.9 for descriptions of the control registers for REMC.

20. DMA Controller (DMAC)

20.1 Overview

The main features of the DMAC are outlined below.

- Supports byte, halfword, and word transfers.
- Each DMAC channel can be configured to different transfer conditions independently.
- Supports memory-to-memory, memory-to-peripheral circuit, and peripheral circuit-to-memory transfers.
- Supports hardware DMA requests from peripheral circuits and software DMA requests.
- Priority level for each channel is selectable from two levels.

Figure 20.1 shows the DMAC configuration.

Table 20.1 DMAC Channel Configuration of S1D13C00

Item	S1D13C00
Number of channels	4 channels (Ch.0 to Ch.3)
Transfer source memories	External Serial Flash and Internal RAM
Transfer destination memories	Internal RAM
Transfer source peripheral circuits	SPI, QSPI, I2C
Transfer destination peripheral circuits	SPI, QSPI, I2C, SND

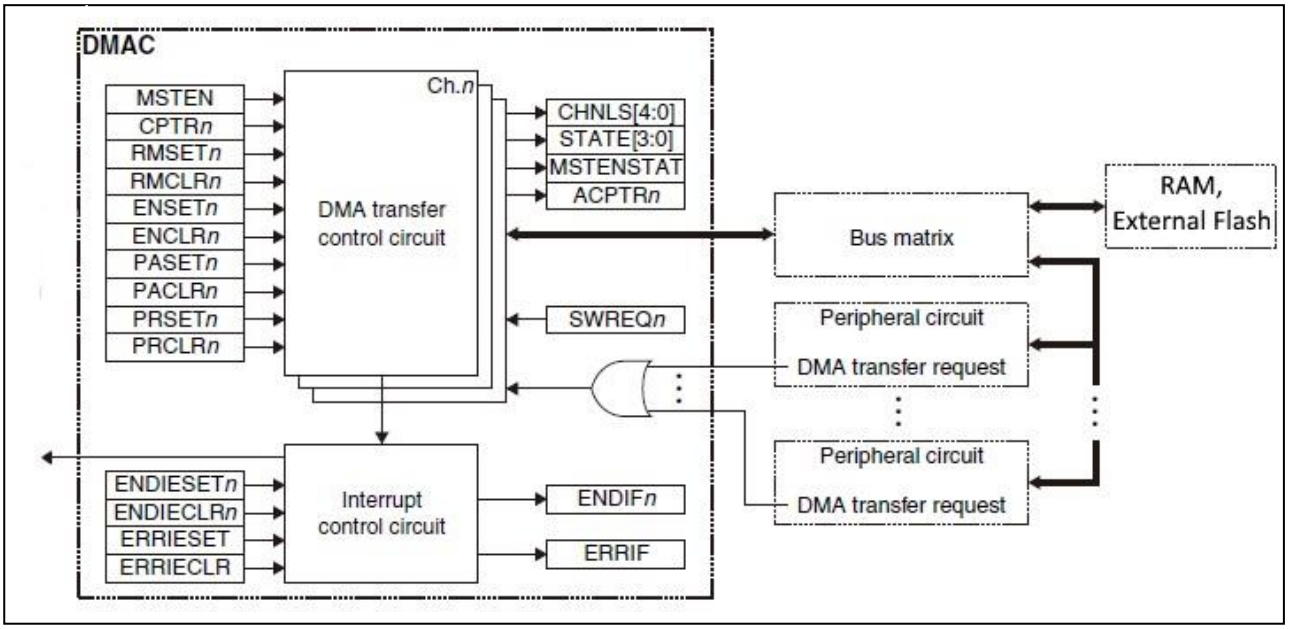


Figure 20.1 DMAC Configuration

20.2 Operations

20.2.1 Initialization

The DMAC should be initialized with the procedure shown below.
1. Set the data structure base address to the DMACCPTR register.
2. Configure the data structure for the channels to be used.
   - Set the control data.
   - Set the transfer source end pointer.
   - Set the transfer destination end pointer.
3. Set the DMACCFG.MSTEN bit to 1. (Enable DMAC)
4. Configure the DMACRMSET and DMACRMCLR registers.
   (Configure masks for DMA transfer requests from peripheral circuits)
5. Configure the DMACENSET and DMACENCLR registers. (Enable channels used)
6. Configure the DMACPASET and DMACPACLR registers. (Select data structure used)
7. Configure the DMACPRSET and DMACPRCLR registers. (Set priorities)
8. Set the following registers when using the interrupt:
   - Write 1 to the interrupt flags in the DMACENDIF and DMACERRIF registers. (Clear interrupt flags)
   - Configures the DMACENDIESET/DMACENDIECLR and DMACERRIESET/DMACERRIECLR registers. (Enable/disable interrupts)
9. Set the DMA request enable bits of the peripheral circuits that use DMA transfer to 1.
10. To issue a software DMA request to Ch.n, write 1 to the DMACSWREQ.SWREQn bit.

20.3 Priority

If DMA requests are issued to two or more channels, the DMA transfers are performed in order from the highest priority channel. The channel of which the priority level is set to 1 by the DMACPRSET.PRSETn bit has the highest priority. If two or more channels have been set to the same priority level, the smaller channel number takes precedence.

20.4 Data Structure

To perform DMA transfers, a data structure that contains basic transfer control information must be provided. The data structure consists of two blocks, primary data structure and alternate data structure, and one of them is used according to the DMA transfer mode.
The data structure can be located at an arbitrary address in the RAM area by setting the base address to the DMACCPTR.CPTR[31:0] bits.
The data structure for each channel consists of a transfer source end pointer, a transfer destination end pointer, and control data. An area of 16 bytes × 2 is allocated in the RAM for each channel.
The whole size of the data structure and the alternate data structure base address depend on the number of channels implemented.

Table 20.2 Data Structure Size According to Number of Channels Implemented

Number of channels Implemented	Data structure Size	Primary data structure Base address	Alternate data structure Base address
1	32	DMACCPTR.CPTR[31:0] (CPTR[4:0] = 0x00)	DMACCPTR.CPTR[31:0] + 0x10
2	64	DMACCPTR.CPTR[31:0] (CPTR[5:0] = 0x00)	DMACCPTR.CPTR[31:0] + 0x20
3 to 4	**128**	**DMACCPTR.CPTR[31:0] (CPTR[6:0] = 0x00)**	**DMACCPTR.CPTR[31:0] + 0x40**
5 to 8	256	DMACCPTR.CPTR[31:0] (CPTR[7:0] = 0x00)	DMACCPTR.CPTR[31:0] + 0x80
9 to 16	512	DMACCPTR.CPTR[31:0] (CPTR[8:0] = 0x000)	DMACCPTR.CPTR[31:0] + 0x100
17 to 32	1,024	DMACCPTR.CPTR[31:0] (CPTR[9:0] = 0x000)	DMACCPTR.CPTR[31:0] + 0x200

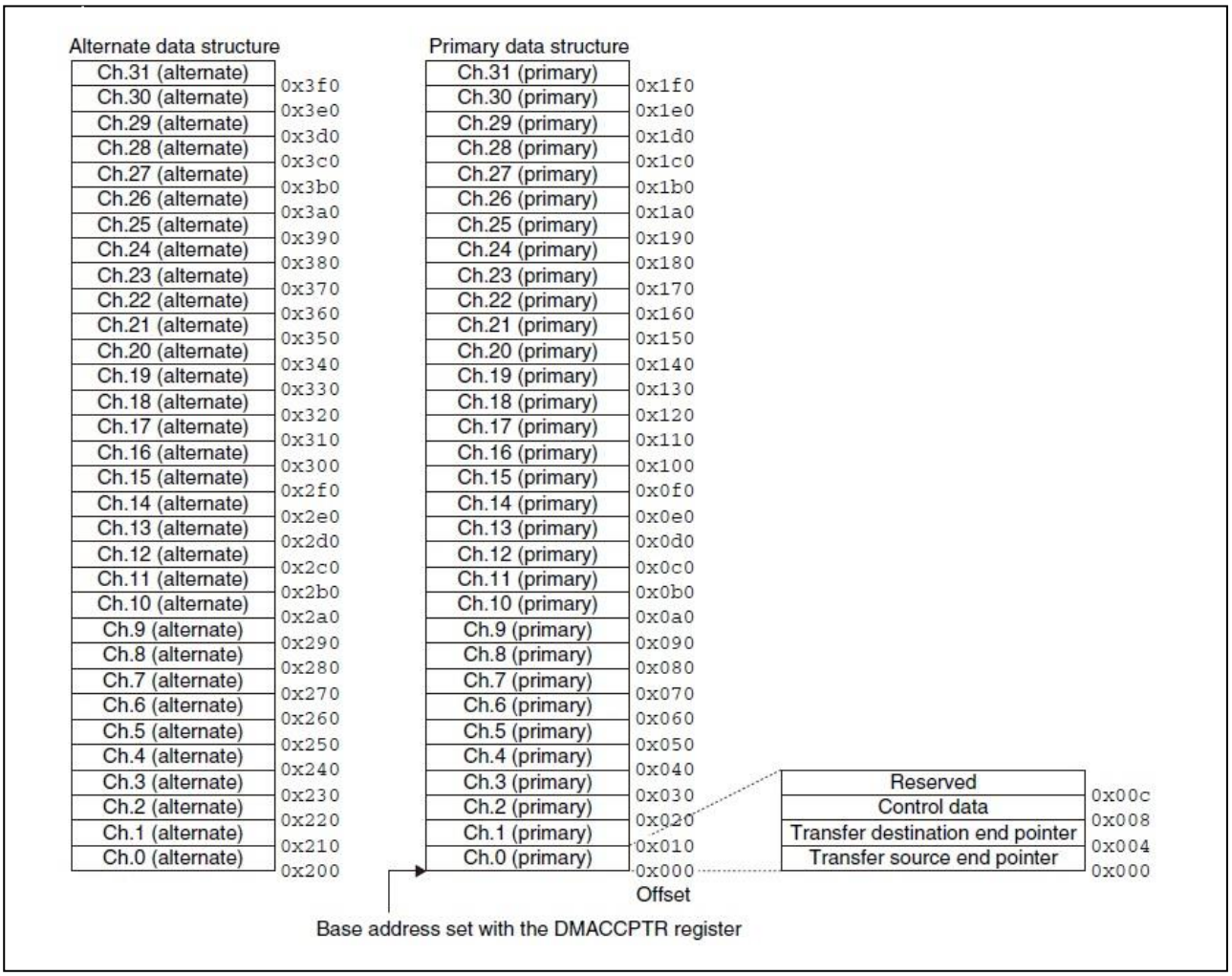


Figure 20.2 Data Structure Address Map (when 32 channels are implemented)

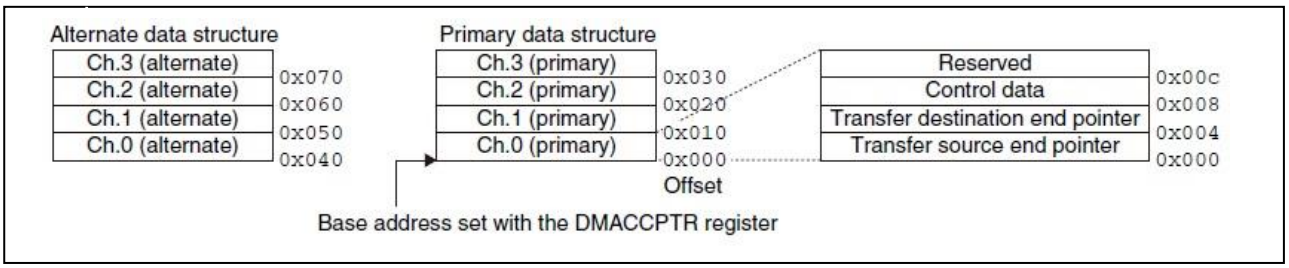


Figure 20.3 Data Structure Address Map (when 4 channels are implemented)

The alternate data structure base address can be determined from the DMACACPTR.ACPTR[31:0] bits.

20.4.1 Transfer Source End Pointer

Set the source data end address. The address of data to be transferred should be set as it is if the transfer source address is not incremented.

20.4.2 Transfer Destination End Pointer

Set the address to which the last transfer data is written. The address for writing transfer data should be set as it is if the transfer destination address is not incremented.

20.4.3 Control Data

Set the DMA transfer information. Figure 20.4 shows the constituent elements of the control data.

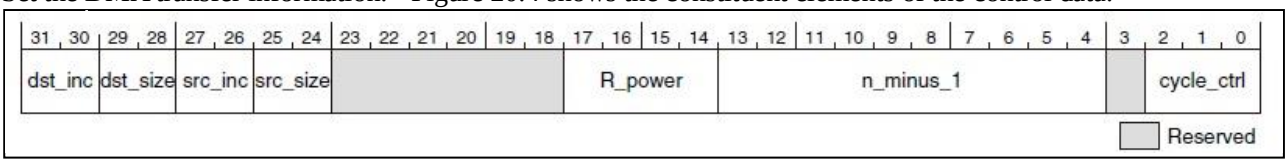


Figure 20.4 Constituent Elements of Control Data

dst_inc

Set the increment value of the transfer destination address. The setting value must be equal to or larger than the transfer data size when the address is incremented.

Table 20.3 Increment Value of Transfer Destination Address

dst_inc	Increment value
0x3	No increment
0x2	+4
0x1	+2
0x0	+1

dst_size

Set the size of the data to be written to the transfer destination. It should be the same value as the src_size.

Table 20.4 Size of Data Written to Transfer Destination

dst_size	Data size
0x3	Reserved
0x2	Word
0x1	Halfword
0x0	Byte

src_inc

Set the increment value of the transfer source address. The setting value must be equal to or larger than the transfer data size when the address is incremented.

Table 20.5 Increment Value of Transfer Source Address

dst_inc	Increment value
0x3	No increment
0x2	+4
0x1	+2
0x0	+1

src_size

Set the size of the data to be read from the transfer source. It should be the same value as the dst_size.

Table 20.6 Size of Data Read from Transfer Source

src_size	Data size
0x3	Reserved
0x2	Word
0x1	Halfword
0x0	Byte

R_power

Set the arbitration cycle during successive data transfer.

Arbitration cycle (2^R) = 2^{R_power}

When the DMAC is performing a successive transfer, it suspends the data transfer at the cycle set with R_power. If DMA requests have been issued at that point, the DMAC re-arbitrates them according to their priorities and then performs a DMA transfer for the channel with the highest priority.
If the arbitration cycle setting value is larger than the number of successive data transfers, successive data transfers will not be suspended.

n_minus_1

Set the number of DMA transfers to be executed successively.

Number of successive transfers (N) = n_minus_1 + 1

When the set number of successive transfers has completed, a transfer completion interrupt occurs.

cycle_ctrl

Set the DMA transfer mode.

Table 20.7 DMA Transfer Mode

cycle_ctrl	DMA transfer mode
0x7	Peripheral scatter-gather transfer (for alternate data structure)
0x6	Peripheral scatter-gather transfer (for primary data structure)
0x5	Memory scatter-gather transfer (for alternate data structure)
0x4	Memory scatter-gather transfer (for primary data structure)
0x3	Ping-pong transfer
0x2	Auto-request transfer
0x1	Basic transfer
0x0	Stop

20.5 DMA Transfer Mode

20.5.1 Basic Transfer

This is the basic DMA transfer mode. In this mode, DMA transfer starts when a DMA transfer request from a peripheral circuit or a software DMA request is issued, and it continues until it is completed for the set number of successive transfers or it is suspended at the arbitration cycle. To resume the DMA transfer suspended at the arbitration cycle, a DMA transfer request must be reissued.
When the set number of successive transfers has completed, a transfer completion interrupt occurs.

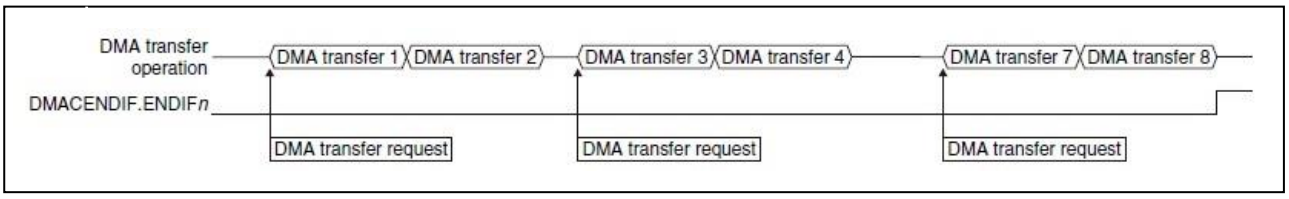


Figure 20.5 Basic Transfer Operation Example (N = 8, 2^R = 2)

20.5.2 Auto-Request Transfer

Similar to the basic transfer, DMA transfer starts when a DMA transfer request from a peripheral circuit or a software DMA request is issued, and it continues until it is completed for the set number of successive transfers or it is suspended at the arbitration cycle. The DMAC resumes the DMA transfer suspended at the arbitration cycle without a DMA transfer request being reissued.
When the set number of successive transfers has completed, a transfer completion interrupt occurs.

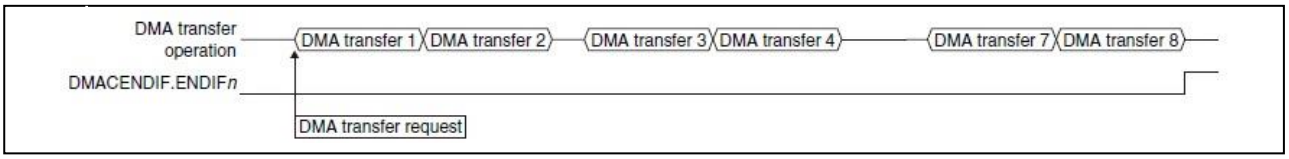


Figure 20.6 Auto-Request Transfer Operation Example (N = 8, 2^R = 2)

20.5.3 Ping-Pong Transfer

In ping-pong transfer mode, the DMAC performs basic transfers repeatedly while switching between the primary data structure and alternate data structure. The data structures are referred alternately, and DMA transfer is terminated when the control data with cycle_ctrl set to 0x0 is referred. A transfer completion interrupt occurs each time a transfer using a data structure is completed.

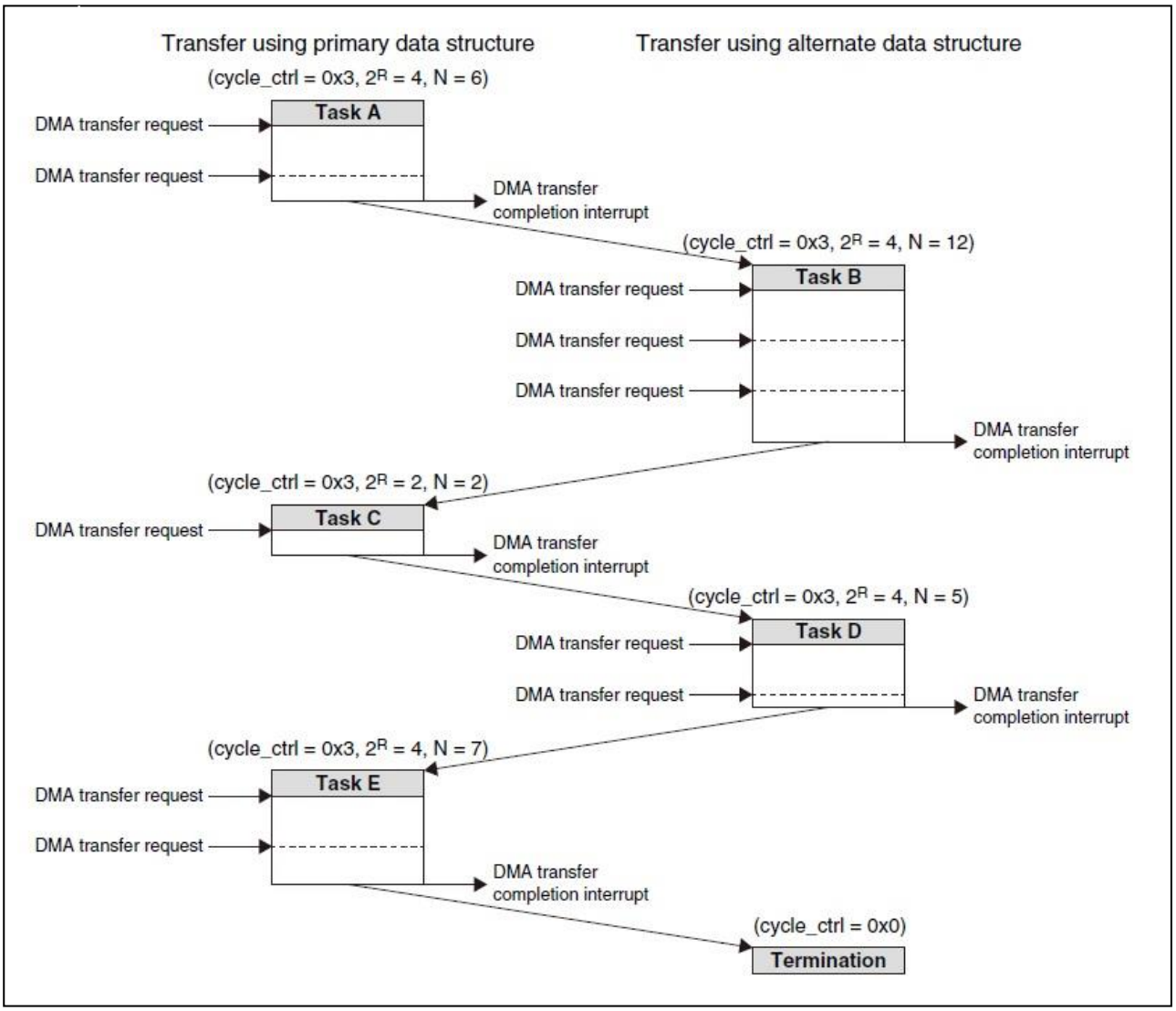


Figure 20.7 Ping-Pong Transfer Operation Example

DMA transfer procedure

1. Start data transfer by following the procedure shown in Section 20.2.1, "Initialization." In Step 2 of the initialization procedure, set Task A and Task B to the primary data structure and the alternate data structure, respectively.
2. Set Task C to the primary data structure after a DMA transfer completion interrupt has occurred by Task A.
3. Set Task D to the alternate data structure when a DMA transfer completion interrupt has occurred by Task B.
4. Repeat Steps 2 and 3.
5. Set cycle_ctrl to 0x0 after a DMA transfer completion interrupt has occurred by the next to last task.
6. The DMA transfer is completed when a DMA transfer completion interrupt occurs by the last task.

20.5.4 Memory Scatter-Gather Transfer

In scatter-gather transfer mode, first the DMAC, using the primary data structure, copies a data structure from the data structure table, which has been prepared with multiple data structures included in advance, to the alternate data structure, and then it performs DMA transfer using the alternate data structure. The DMAC performs this operation repeatedly. By programming the transfer mode of the data structure located at the end of the table as a basic transfer, the DMA transfer can be terminated with a transfer completion interrupt. This mode requires a

DMA transfer request only for starting the first data transfer. Subsequent data transfers are performed by auto-requests.

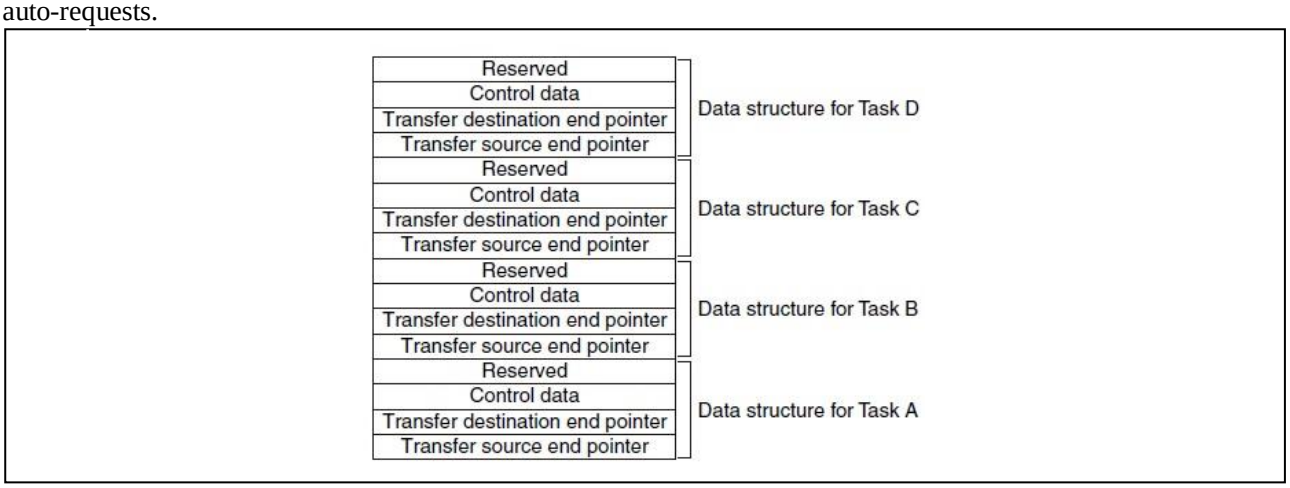


Figure 20.8 Example of Data Structure Table for Scatter-Gather Transfer

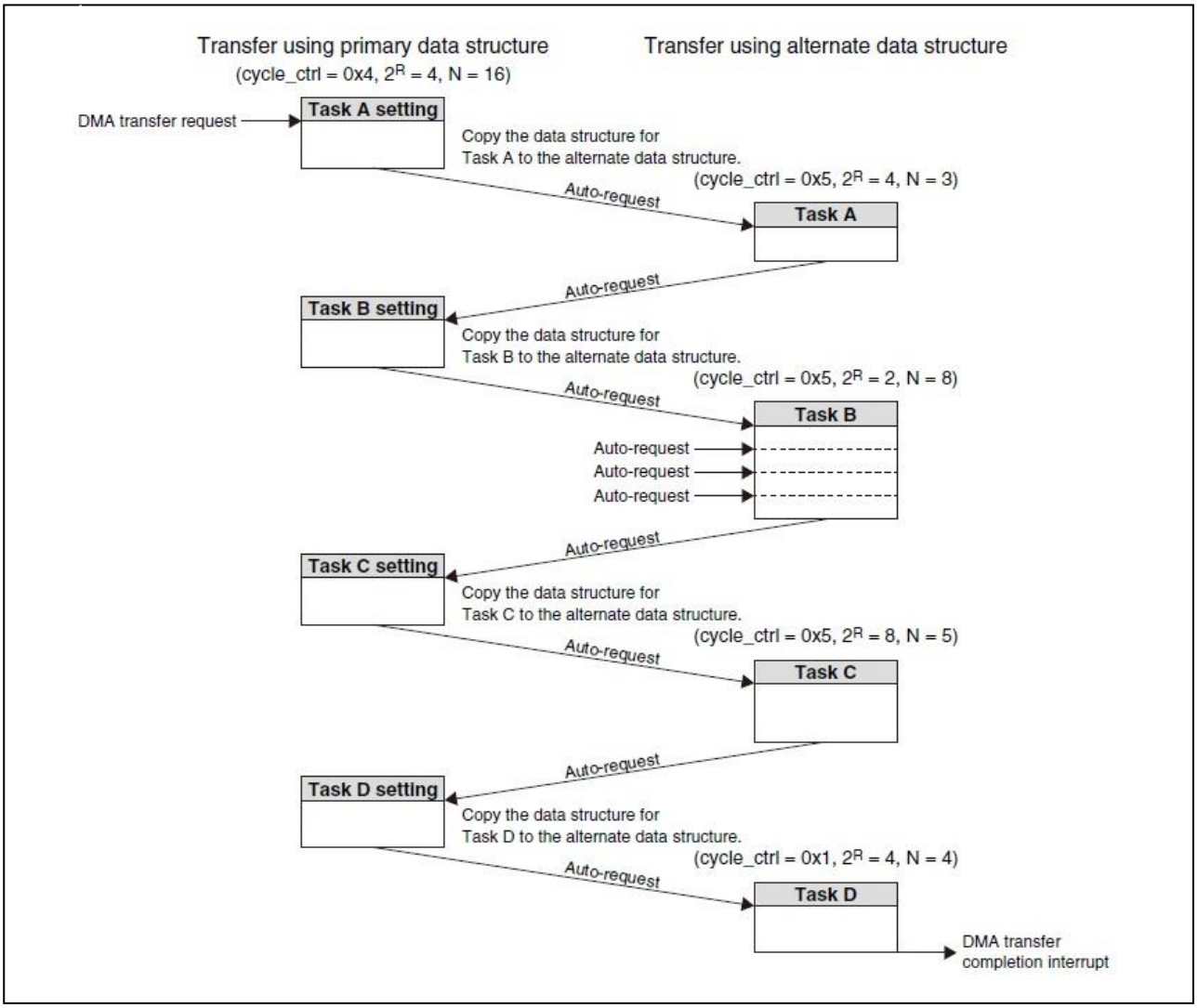


Figure 20.9 Memory Scatter-Gather Transfer Operation Example

DMA transfer procedure

1. Configure the data structure table for scatter-gather transfer.
   Set the cycle_ctrl for the last task to 0x1 and those for other tasks to 0x5.
2. Start data transfer by following the procedure shown in Section 20.2.1, "Initialization." In Step 2 of the initialization procedure, configure the primary data structure with the control data shown below.

   Transfer source end pointer = Data structure table end address
   Transfer destination end pointer = Alternate data structure end address
   dst_inc = 0x2
   dst_size = 0x2
   src_inc = 0x2
   src_size = 0x2
   R_power = 0x2
   n_minus_1 = Number of tasks × 4 – 1
   cycle_ctrl = 0x4

3. The DMA transfer is completed when a DMA transfer completion interrupt occurs.

20.5.5 Peripheral Scatter-Gather Transfer

In memory scatter-gather transfer mode, the second and subsequent DMA transfers are performed by auto-requests. On the other hand, in peripheral scatter-gather transfer mode, all DMA transfers are performed by a DMA transfer request issued by a peripheral circuit or a software DMA request.

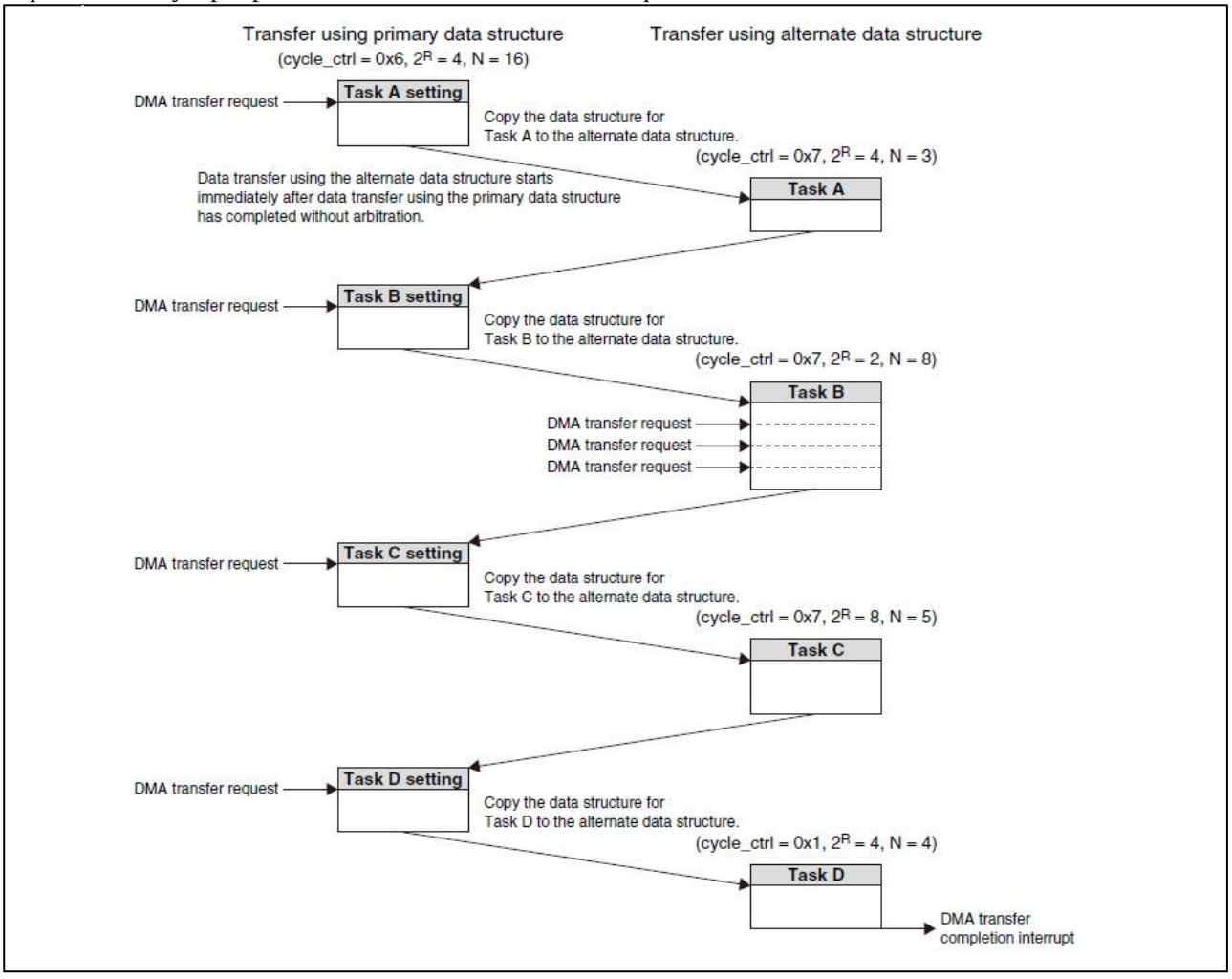


Figure 20.10 Peripheral Scatter-Gather Transfer Operation Example

DMA transfer procedure

1. Configure the data structure table for scatter-gather transfer.
   Set the cycle_ctrl for the last task to 0x1 and those for other tasks to 0x7.
2. Start data transfer by following the procedure shown in Section 20.2.1, "Initialization." In Step 2 of the initialization procedure, configure the primary data structure with the control data shown below.

   Transfer source end pointer = Data structure table end address
   Transfer destination end pointer = Alternate data structure end address
   dst_inc = 0x2
   dst_size = 0x2
   src_inc = 0x2
   src_size = 0x2
   R_power = 0x2
   n_minus_1 = Number of tasks × 4 – 1
   cycle_ctrl = 0x6

3. Issue a DMA transfer request in each task using a peripheral circuit or via software.
4. The DMA transfer is completed when a DMA transfer completion interrupt occurs.

20.6 DMA Transfer Cycle

A DMA transfer requires several clock cycles to execute. Figure 20.11 shows sequence of the internal bus accesses of a DMA transfer cycle. Note that a bus access may take more than one clock cycle depending on the access type (read or write) and memory type (Flash, RAM, memory-mapped serial flash, registers).

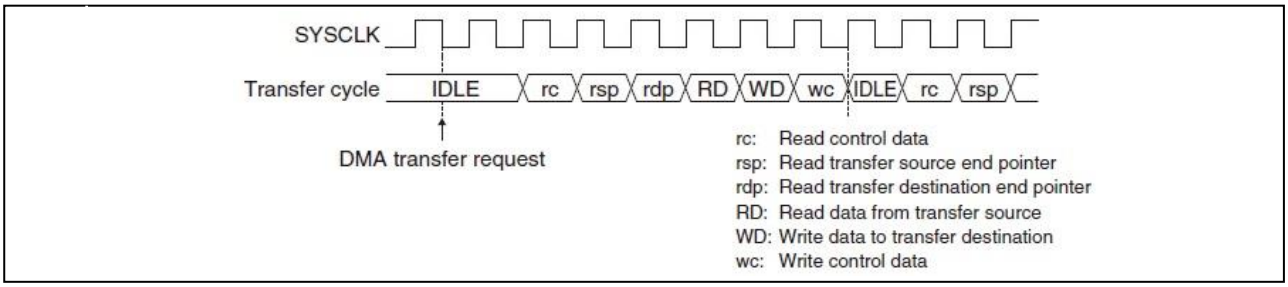


Figure 20.11 DMA Transfer Cycle

20.7 Interrupts

The DMAC has a function to generate the interrupts shown in Table 20.8.

Table 20.8 DMAC Interrupt Function

Interrupt	Interrupt flag	Set condition	Clear condition
DMA transfer completion	DMACENDIF.ENDIFn	When DMA transfers for a set number of successive transfers have completed	Writing 1
DMA transfer error	DMACERRIF.ERRIF	When an internal bus error has occurred	Writing 1

The DMAC provides interrupt enable bits corresponding to each interrupt flag. An interrupt request is sent to the Host MCU only when the interrupt flag, of which interrupt has been enabled by the interrupt enable bit, is set. Figure 20.12 shows a diagram of the DMAINT interrupt signal. The DMAINT signal is logical ORed with interrupt signals from other peripherals to generate the HIFIRQ interrupt signal to the host MCU.
See Section 22 ("Interrupts") for more details on the HIFIRQ output.

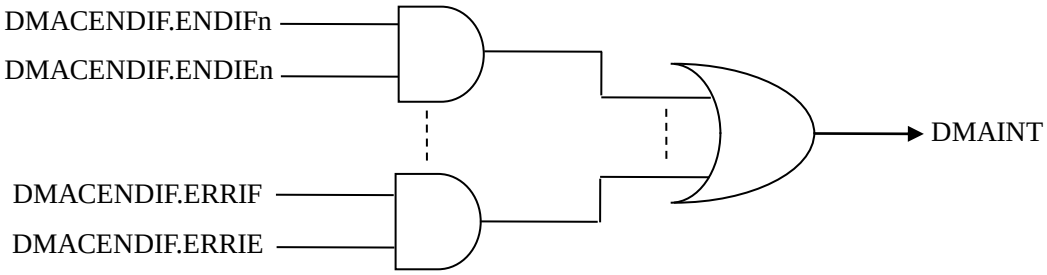


Figure 20.12 DMAINT Interrupt Circuit

20.8 Control Registers

See Section 10.10 for descriptions of the control registers for DMAC.

21. Memory Display Controller (MDC)

21.1 Overview

The MDC is a multi-function peripheral which provides hardware support for interfacing to memory displays. It also provides graphics hardware acceleration functions. The features of the MDC are listed below.

- Panel interfaces:
  - Parallel 6-bit color interface
  - SPI 1-bit black-and-white interface + COM
  - SPI 3-bit color interface + COM
- Output voltage supplies for display panels:
  - Programmable VMDH and VMDL step-up voltage supplies
- Rotation of frame buffer image to panel: 0, 90, 180, and 270 degrees
- Color formats: 6-bit RRGGBB, 3-bit RGB color, 1-bit black-and-white
- Bitmap formats: 1 bit, 2 bits (with alpha-blending value)
- Image/bitmap copy functions:
  - Image/bitmap copy with scaling and rotation from source to destination memory location with alpha-blending of background and foreground pixels for 6-bit color.
  - Image/bitmap copy with horizontal and vertical shearing from source to destination memory location with alpha-blending of background and foreground pixels for 6-bit color.
- Drawing functions:
  - Line drawing with programmable thickness.
  - Rectangle drawing, filled and unfilled, with programmable horizontal and vertical line thickness for unfilled rectangle drawing.
  - Ellipse drawing, filled and unfilled, with programmable thickness at X- and Y-axis crossings for unfilled ellipse drawing
- External Host Interface:
  - Configurable indirect 8-bit parallel, SPI, or QSPI interface.
  - Allows an external host MCU to access the internal bus.
- Event Processor with simple instruction set

Figure 21.1 shows the MDC block diagram.

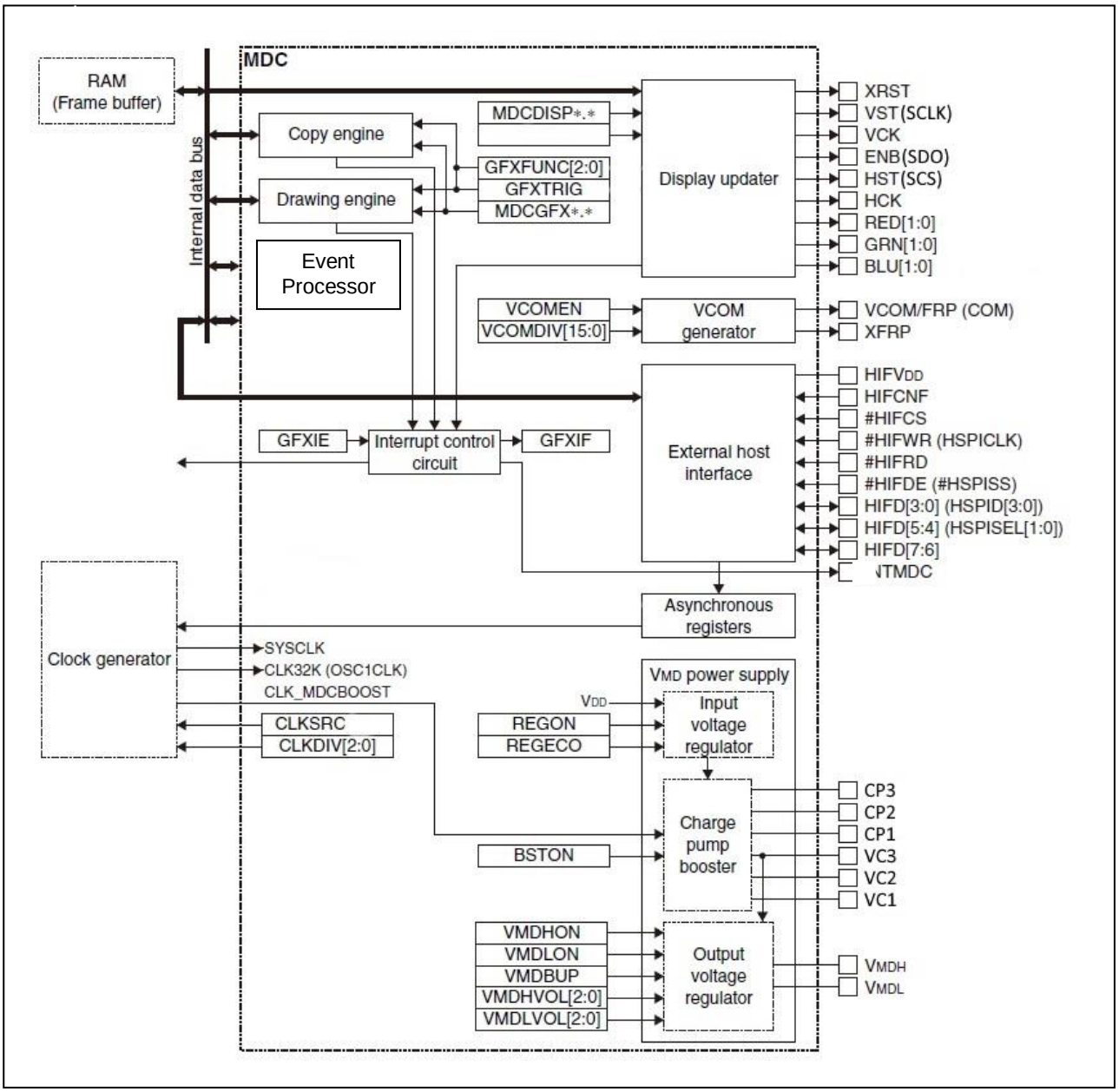


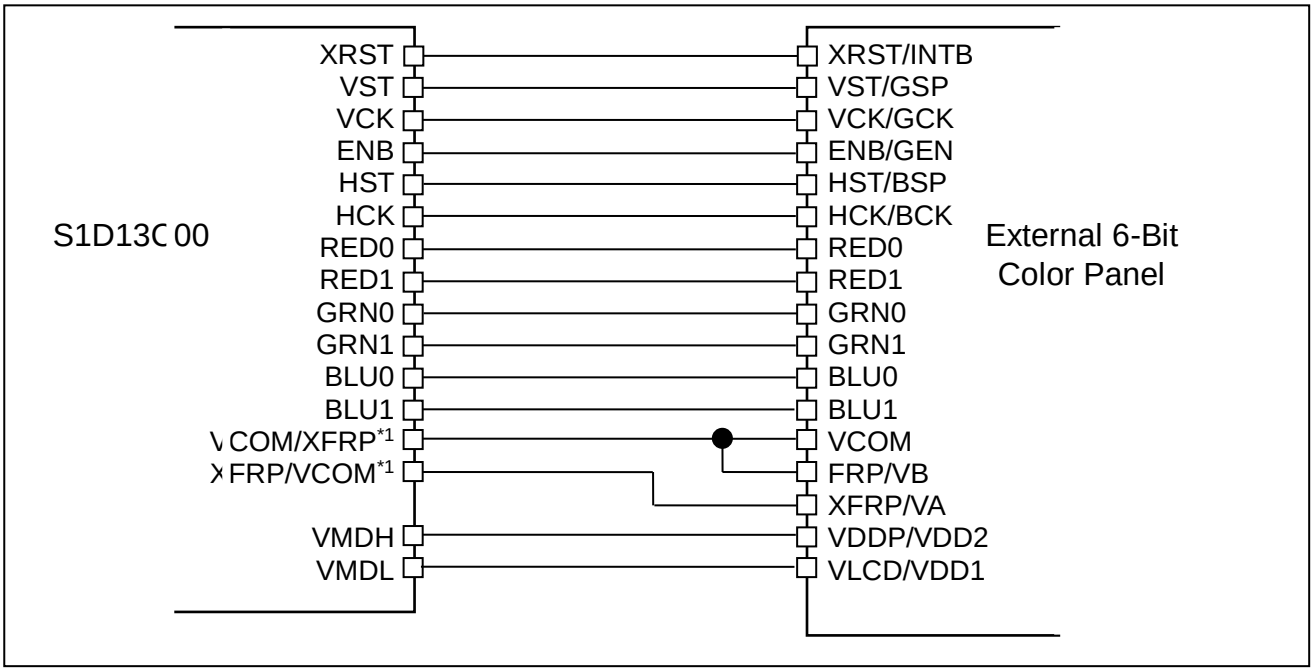
Figure 21.1 MDC Block Diagram

21.2 Input/Output Pins and External Connections

21.2.1 Display Panel External Connections

Section 4.2.2 describes the pins for the panel interface. Figure 21.2 to Figure 21.8 show the connections for the different panel interfaces supported. Sections in 21.6.2 describe the initialization procedure for each panel.

21.2.1.1 6-Bit Color Panel Connections



*1 VCOM/XFRP pins should be connected according to the panel specification

Figure 21.2 Connection between S1D13C00 and 6-Bit Color Panel

21.2.1.2 SPI Panel Connections

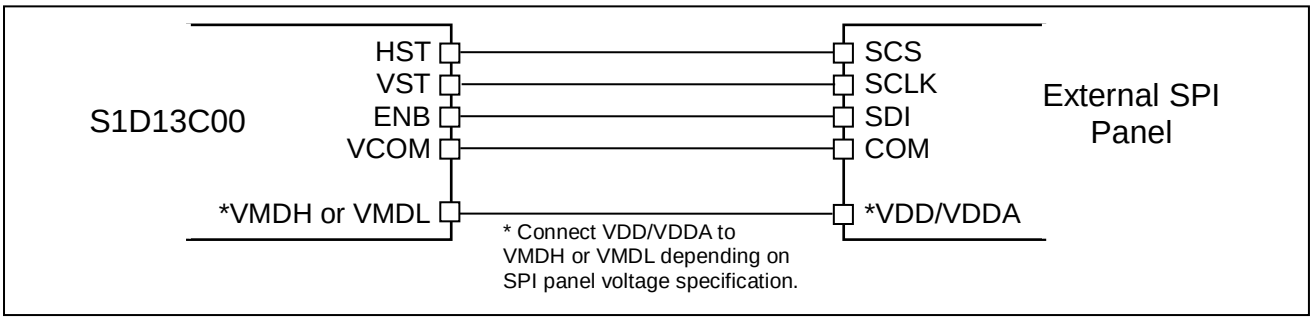


Figure 21.3 Connection between S1D13C00 and 1-/3-Bit SPI Panel

21.2.1.3 Grayscale Panels Connections

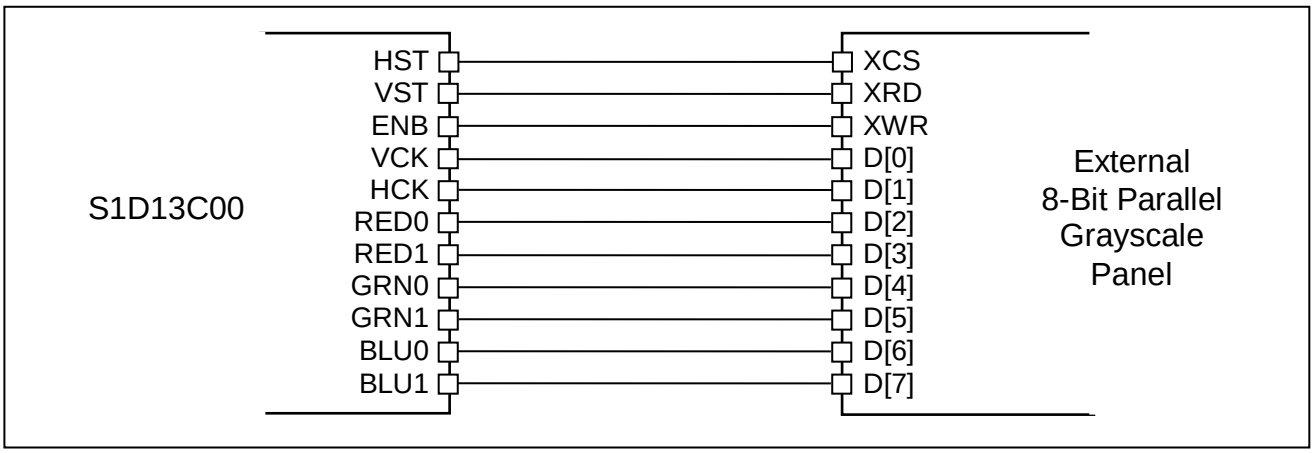


Figure 21.4 Connection between S1D13C00 and 8-Bit Parallel Grayscale Panel

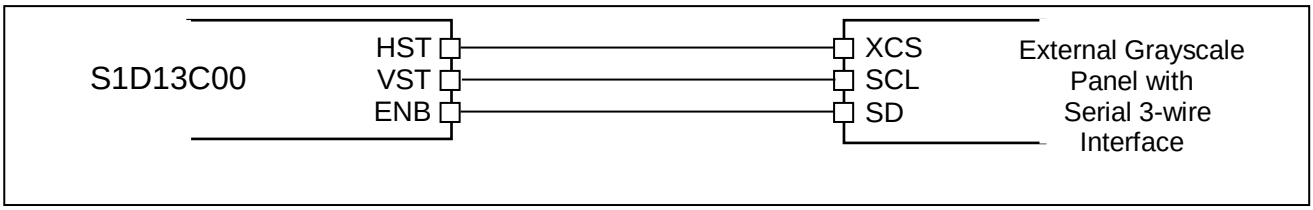


Figure 21.5 Connection between S1D13C00 and 3-Wire Serial Grayscale Panel

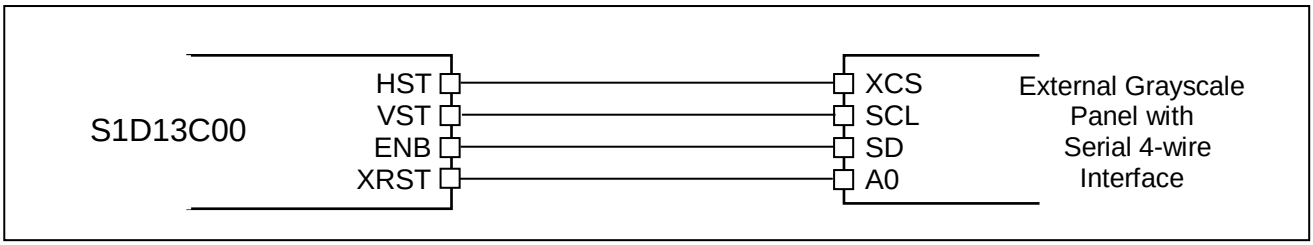


Figure 21.6 Connection between S1D13C00 and 4-Wire Serial Grayscale Panel

21.2.1.4 EPD Panel Connections

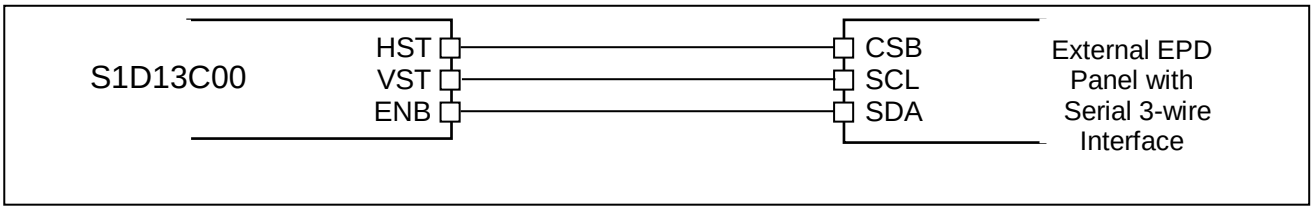


Figure 21.7 Connection between S1D13C00 and 3-Wire Serial EPD Panel

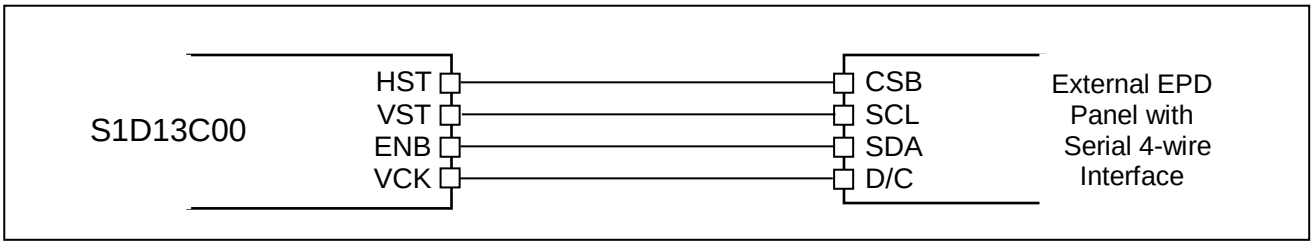


Figure 21.8 Connection between S1D13C00 and 4-Wire Serial EPD Panel

21.2.2 External Host Interface Connections

Section 4.2.1 describes the pins for the host MCU interface and Section 4.3 provides the connection information (including unused pins termination) for the types of host interface connections. Figure 21.9 to Figure 21.12 show connection examples to a Host MCU with the 4 different interface types.

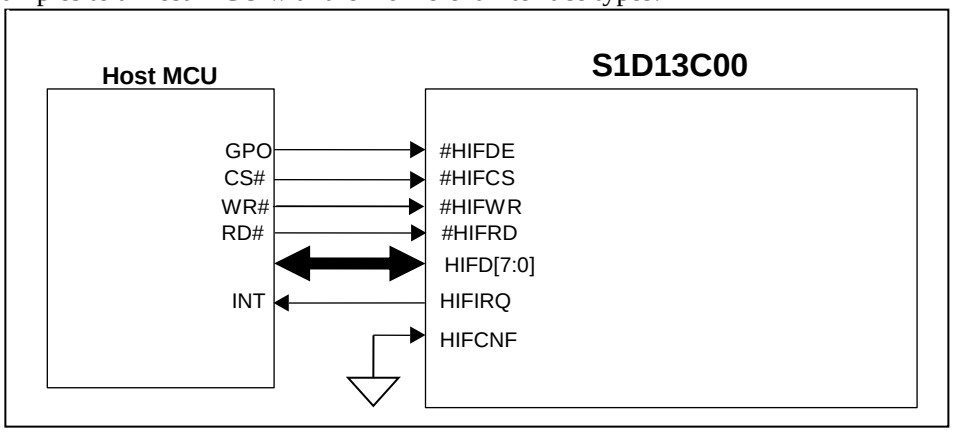


Figure 21.9 Connection between Host MCU and S1D13C00 with Indirect 8-Bit Interface

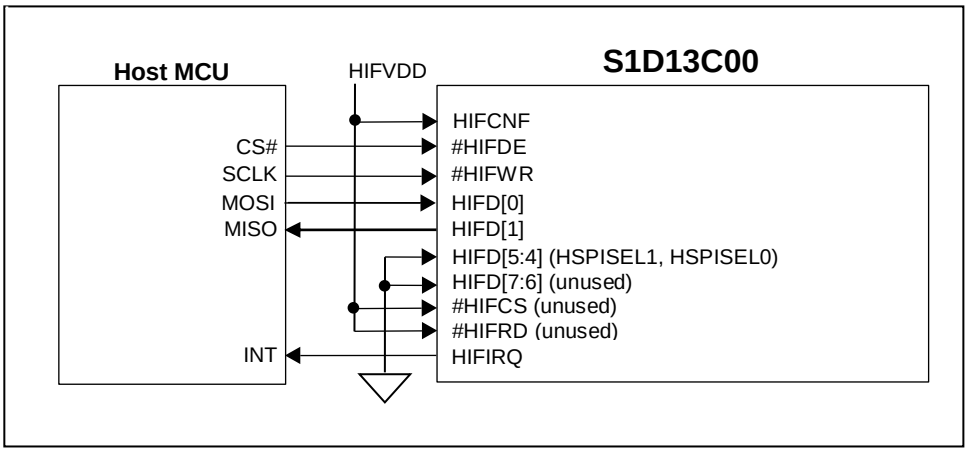


Figure 21.10 Connection between Host MCU and S1D13C00 with Normal SPI Interface

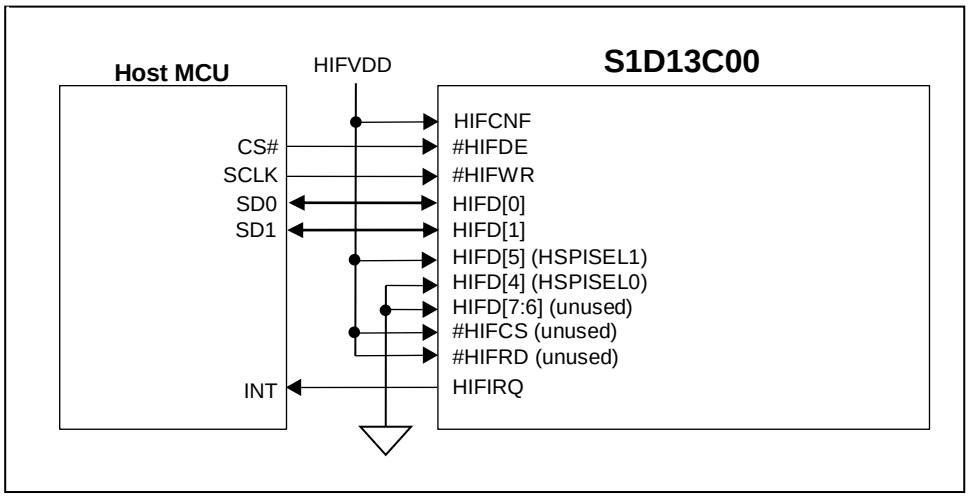


Figure 21.11 Connection between Host MCU and S1D13C00 with Dual-Data SPI Interface

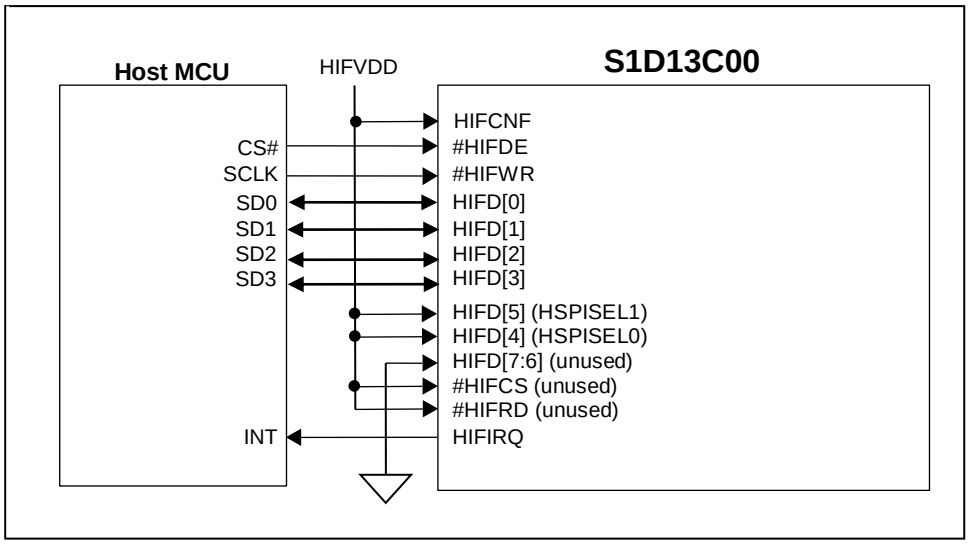


Figure 21.12 Connection between Host MCU and S1D13C00 with Quad-Data SPI Interface

Unused pins for SPI interfaces should be properly terminated (see Section 4.3). Bidirectional data I/O pins (HIFD[7:0]) should be terminated with pull-up or pull-down resistors to reduce current when the interface is idle.

21.3 Voltage Booster

The display drive voltages VMDH and VMDL can be generated by the internal VMD power supply circuit.

21.3.1 Overview

The MDC has a built-in voltage booster circuit and linear regulators which provide two voltage supplies VMDH and VMDL. These two voltage supplies each have a 3-bit voltage selection control to support supply requirements of different memory display panels. The following table shows the nominal voltage selections for VMDH and VMDL:

Table 21.1 VMDH and VMDL Nominal Voltage Selections

MDCBSTVMD.VMDHVOL[2:0]	VMDH
0	4.3V
1	4.4V
2	4.5V
3	4.6V
4	4.7V
5	4.8V
6	4.9V
7	5.0V

MDCBSTVMD.VMDLVOL[2:0]	VMDL
0	2.3V
1	2.7V
2	3.0V
3	3.1V
4	3.2V
5	3.3V
6	3.4V
7	3.6V

The minimum input supply voltage for the voltage booster circuit is VDD=2.0V.

21.3.2 Configuration of VMD Power Supply Circuit

Figure 21.13 shows a block diagram of the voltage booster circuit.

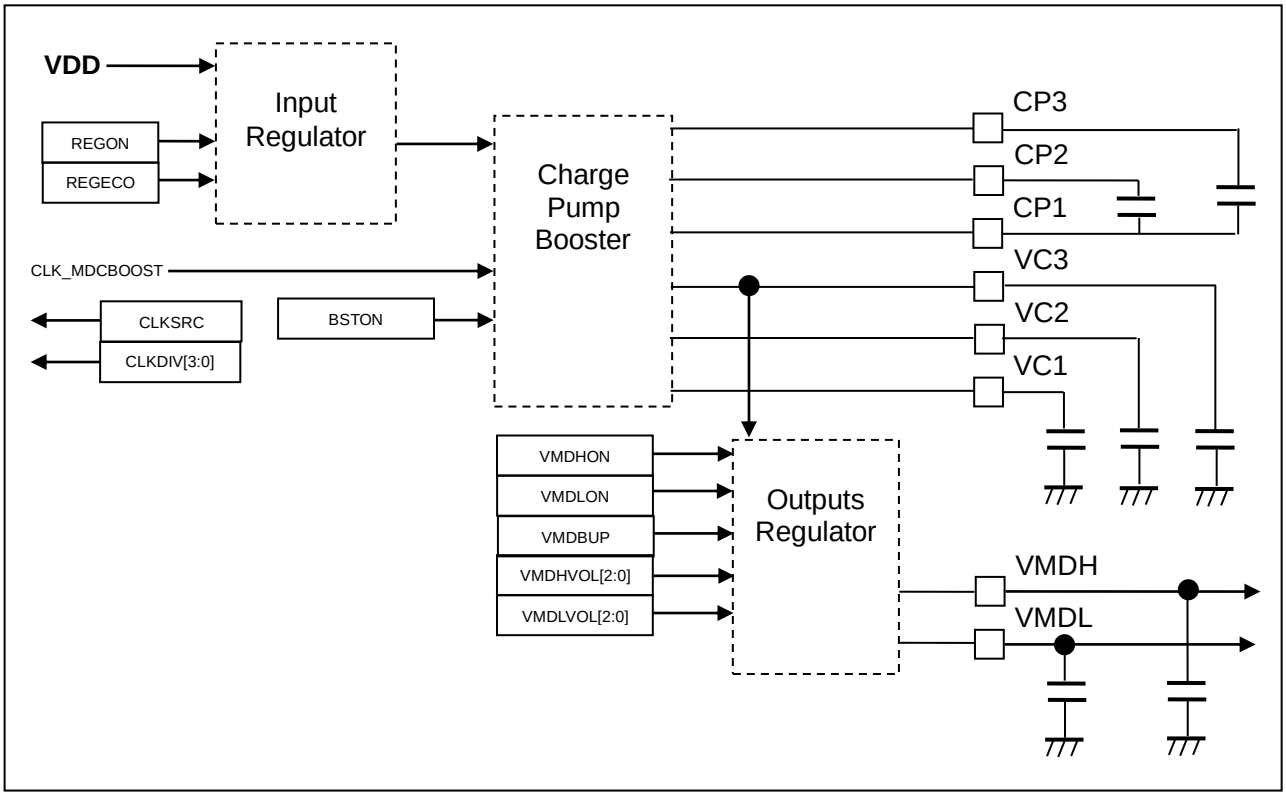


Figure 21.13 Voltage Booster Block Diagram

The internal VMD power supply circuit consists of an input voltage regulator, a charge pump booster, and an output voltage regulator. The input voltage regulator generates a regulated constant voltage for driving the charge pump booster from the power supply voltage VDD. The charge pump booster generates a boosted voltage VC3. The output voltage regulator generates VMDH and VMDL from VC3. The VMDH and VMDL output voltage levels can be adjusted via software.

If the display panel does not use the VMDH and VMDL voltages, leave the VMDH and VMDL pins open. Do not alter the VMD power supply control bit initial values.

21.3.3 Controlling VMD Power Supply Circuit

The VMD power supply circuit should be controlled as in the procedure shown below.

1. Configure the MDCBSTCLK.CLKSRC and MDCBSTCLK.CLKDIV[2:0] bits. (Configure voltage booster clock)
2. Configure the following MDCBSTPWR register bits:
   - Write 0 to the MDCBSTPWR.REGECO bit. (Set voltage regulator to normal mode) *1
   - Write 1 to the MDCBSTPWR.REGON bit. (Activate voltage regulator)
   - Write 1 to the MDCBSTPWR.BSTON bit. (Activate voltage booster)

   *1 This bit can be set to 1 for “economy mode” if the display is static (no display update). Before performing a display update, it should be cleared back to 0 for “normal mode”.
3. Wait for the VC3 boosted voltage output to stabilize.
   For the boosted voltage output stabilization time, refer to “DC Characteristics” chapter.

4. Configure the following MDCBSTVMD register bits:
   - MDCBSTVMD.VMDHVOL[2:0] bits (Adjust VMDH voltage level)
   - MDCBSTVMD.VMDLVOL[2:0] bits (Adjust VMDL voltage level)
   - Write 1 to the MDCBSTVMD.VMDHON bit. (Enable VMDH output)
   - Write 1 to the MDCBSTVMD.VMDLON bit. (Enable VMDL output)
5. Wait for the VMDH and VMDL voltage outputs to stabilize.
   For the VMD voltage outputs stabilization time, refer to "DC Characteristics" chapter.

21.3.4 External VC3 Input Supply

If higher output current is required for VMDH and VMDL, or VDD supply voltage is required to be the same as in booster off, the internal voltage booster can be turned off and the VMD power supply can be powered by an external source through the VC3 pin. Note, the power supply for VC3 needs to cover the output current for VMDH and VMDL.

When applying an external power source through the VC3 pin, set the MDCBSTPWR.REGON and MDCBSTPWR.BSTON bits to 0.

For the external supply voltage applied to VC3, refer to "DC Characteristics" chapter.

See Section 4.5 for details about the connection diagram for applying an external VC3 supply.

21.4 VCOM/XFRP Output

The VCOM and XFRP output clock signals are both off and logic 0 on power-up or reset. The pixels on the panel are not visible (black display) when VCOM/XFRP is off, and VCOM/XFRP must be turned on in order for the pixels to become visible. The VCOM/XFRP outputs require a proper turn-on and turn-off timing sequence which is handled by the VCOM/XFRP clock generation circuit. The following descriptions assume that the system clock is already turned on (SYSCTRL.IOSCEN=1) in order to access the synchronous registers in the S1D13C00.

Turn-On Sequence

1. Make sure the OSC1 clock is turned on first. See Section 7.1.2 for details.
2. Set the desired VCOM frequency by programming the MDCDISPVCOMDIV register. See Section 10.11.4 for details on the relationship between the VCOM frequency and the MDCDISPVCOMDIV register value.
3. Set MDCDISPCTL.VCOMEN to 1 to turn on VCOM.

Turn-Off Sequence

1. Clear the MDCDISPCTL.VCOMEN bit to 0 to turn off VCOM/XFRP.
2. If OSC1 will be turned off, wait at least one VCOM/XFRP clock cycle before turning off OSC1.
3. Turn off OSC1 (if required by application).

Dynamic Switching of VCOM/XFRP Frequency

When the VCOM/XFRP is on (running), its frequency can be dynamically switched by the following steps:

1. Set the MDCVCOMCLKCTL.VCOMCLKDIS bit to 1 to disable the clock for the VCOM/XFRP clock generation circuit.
2. Set the MDCVCOMCLKCTL.VCOMCNTRST bit to 1 to reset the internal 16-bit counter for VCOM/XFRP clock generation to 0x0000.
3. Clear the MDCVCOMCLKCTL.VCOMCNTRST bit to 0 to release the internal 16-bit counter for VCOM/XFRP from reset.
4. Write the new MDCDISPVCOMDIV register value which will determine the new frequency.
5. Clear the MDCVCOMCLKCTL.VCOMCLKDIS bit to 0 to enable the clock for the VCOM/XFRP clock generation circuit.

21.5 Image Data Formats

21.5.1 Pixel Data Formats

The MDC supports several pixel data formats depending on the type of panel selected. The panel type and pixel data format are controlled by the following register bits:

- MDCDISPCTL.DISPEPD – selects EPD panels
- MDCDISPCTL.DISPGS – selects grayscale panel or other panels
- MDCDISPCTL.DISPSPI – selects 1-bit/3-bit SPI panel or 6-bit color panel
- MDCDISPCTL.SPITYPE – selects SPI panel type between 1-bit black-and-white or 3-bit color
- MDCTRIGCTL.DISPBPP[1:0] – selects grayscale panel bits-per-pixel (BPP) format
- MDCDISPCTL2.GSALPHA – enables/disables alpha channel for grayscale formats

The following table shows the control register bits and panel types selected:

Table 21.2 Register Control Bits for Panel Types

Panel Type	MDCDISPCTL. DISPEPD	MDCDISPCTL. DISPGS	MDCDISPCTL. DISPSPI	MDCDISPCTL. SPITYPE	MDCTRIGCTL. DISPBPP[1:0]	MDCDISPCTL2. GSALPHA
6-Bit Color	0	0	0	-	-	-
3-Bit Color SPI	0	0	1	1	-	-
*1-Bit B&W SPI	0	0	1	0	-	0 = no alpha channel 1 = 1-bit alpha channel
*1BPP EPD/Grayscale	-	1	-	-	00b	0 = no alpha channel 1 = 1-bit alpha channel
2BPP EPD/Grayscale	-	1	-	-	01b	0 = no alpha channel 1 = 2-bit alpha channel
4BPP EPD/Grayscale	-	1	-	-	10b	0 = no alpha channel 1 = 4-bit alpha channel
8BPP EPD/Grayscale	-	1	-	-	11b	0 = no alpha channel 1 = 8-bit alpha channel

* The 1-Bit black-and-white SPI panel and 1BPP Grayscale panel pixel formats are the same (1BPP with or without 1-bit alpha channel).

21.5.1.1 6-Bit Color

The 6-bit color format supports a 6-bit RRGGBB data (2 bits per color) and each pixel is stored as a byte in memory. The format of each pixel byte is as follows:

Table 21.3 6-Bit Color Byte Format

7	6	5	4	3	2	1	0
A1	A0	R1	R0	G1	G0	B1	B0

The upper two bits store the alpha-channel value (0 to 3). The alpha-channel value is only used during drawing or pixel copy operations and is ignored by the display updater which transfers pixels to the panel. The alpha value specifies the alpha-blending level between the source pixel and the background/destination pixel. Alpha value of 00b means the pixel is transparent (invisible) and alpha value of 11b means the pixel is 100% visible. Alpha value of 01b is 33% visible and alpha value of 10b is 67% visible.

The memory organization of the frame buffer has the top left pixel at the base address of the frame buffer. The address increases from left to right pixel of each line of the image and continues increasing onto the subsequent lines until the bottom right of the image, as shown below:

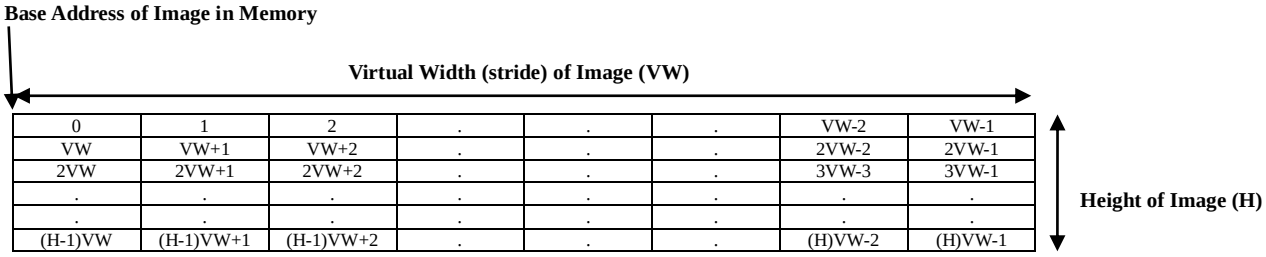


Figure 21.14 Frame Buffer for 6-bit Color Image

Figure 21.15 shows an example of a 10x2 image pattern of pixels:

Assuming the alpha channel is 100% (0b11) and color component values are 100% (0b11) for all the pixels in this example, the bytes in memory would be as follows:

Address	Value (binary)	Value (hex)	Color	Address	Value (binary)	Value (hex)	Color
0	0b11110000	0xF0	0	10	0b11111111	0xFF	10
1	0b11001100	0xCC	1	11	0b11000011	0xC3	11
2	0b11000011	0xC3	2	12	0b11111111	0xFF	12
3	0b11111111	0xFF	3	13	0b11001100	0xCC	13
4	0b11000000	0xC0	4	14	0b11110000	0xF0	14
5	0b11111100	0xFC	5	15	0b11110000	0xF0	15
6	0b11110011	0xF3	6	16	0b11111111	0xFF	16
7	0b11001111	0xCF	7	17	0b11000000	0xC0	17
8	0b11001100	0xCC	8	18	0b11111100	0xFC	18
9	0b11001100	0xCC	9	19	0b11111100	0xFC	19

Figure 21.15 6-bit Color Image Data Example

21.5.1.2 3-Bit Color with 1-Bit Alpha Channel

Two SPI 3-bit color pixels are packed into a byte in memory. The format of each pixel byte is as follows:

Table 21.4 3-Bit Color Byte Format

7	6	5	4	3	2	1	0
A1	R1	G1	B1	A0	R0	G0	B0

The Rx/Gx/Bx bits are the pixel bits and the Ax bits are the alpha channel bits. The alpha-channel bit is only used during drawing and pixel copy operations and is ignored by the display updater which transfers pixels to the panel. Ax = 0 means the pixel is transparent (invisible) and Ax = 1 means the pixel is 100% visible. R0G0B0 is the left pixel on the display and R1G1B1 is the right pixel. The memory organization of the frame buffer has the top, two leftmost pixel data (first byte) at the base address of the frame buffer. The address increases from left to right groups of two pixels of each line of the image and continues increasing onto the subsequent lines until the bottom right of the image, as shown below:

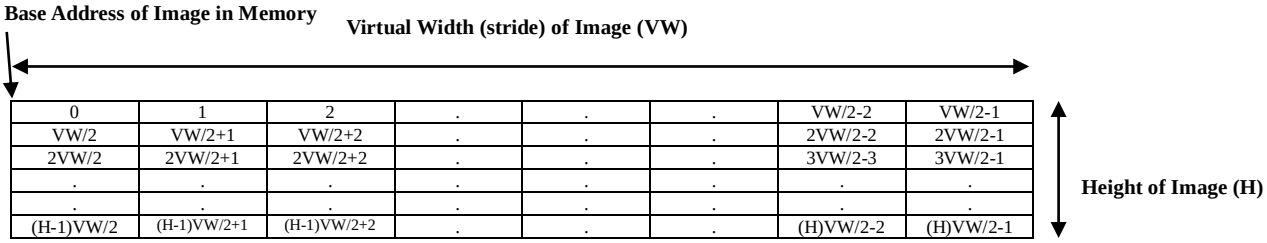


Figure 21.16 Frame Buffer for SPI 3-bit Color Image

Figure 21.17 shows an example of a 10x2 image pattern of pixels:

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19

Assuming the alpha channel (bit) is 100% (0b1) for all the pixels in this example, the bytes in memory would be as follows:

Address	Value (binary)	Value (hex)		
0	0b10101100	0xAC	1	0
1	0b11111001	0xF9	3	2
2	0b11101000	0xE8	5	4
3	0b10111101	0xBD	7	6
4	0b10101010	0xAA	9	8
5	0b10011111	0x9F	11	10
6	0b10101111	0xAF	13	12
7	0b11001100	0xCC	15	14
8	0b10001111	0x8F	17	16
9	0b11101110	0xEE	19	18

Figure 21.17 SPI 3-bit Color Image Data Example

21.5.1.3 1BPP without Alpha Channel (MDCDISPCTL2.GSALPHA=0)

Eight 1-bit pixels are packed into a byte in memory. The format of each pixel byte is as follows:

Table 21.5 Byte Format for 1BPP without Alpha

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

The Px bits are the 8 pixel bits. P0 is the leftmost pixel on the display and P7 is the rightmost pixel. The memory organization of the frame buffer has the top, 8 leftmost pixels (first byte) at the base address of the frame buffer. The address increases from left to right groups of 8 pixels of each line of the image and continues increasing onto the subsequent lines until the bottom right of the image, as shown below:

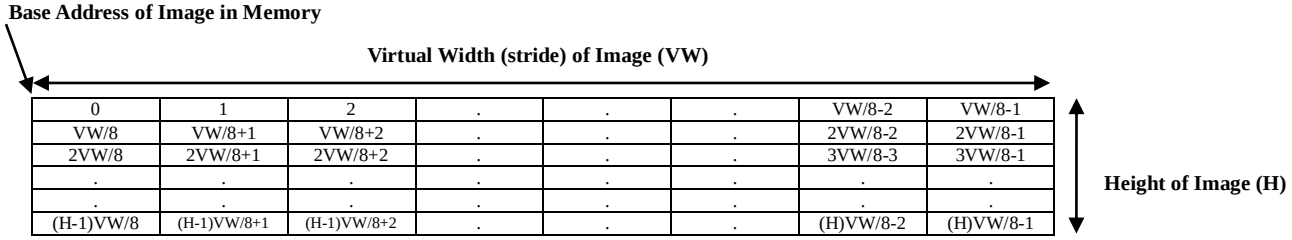


Figure 21.18 Frame Buffer for 1BPP without Alpha Image

Figure 21.19 shows an example of a 10x2 image pattern of black-and-white pixels:

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19

The bytes in memory would be as follows:

Address	Value (binary)	Value (hex)								
0	0b11001010	0xCA	7	6	5	4	3	2	1	0
1	*0x000000*10	0x02							9	8
2	0x00110101	0x35	17	16	15	14	13	12	11	10
3	*0x000000*01	0x01							19	18

Figure 21.19 1BPP without Alpha Image Data Example

There are 8 pixels per byte and each line occupies 2 bytes (roundup(10/8)). The upper 6 bits of the second byte of a line are not used.

21.5.1.4 1BPP with 1-bit Alpha Channel (MDCDISPCTL2.GSALPHA=1)

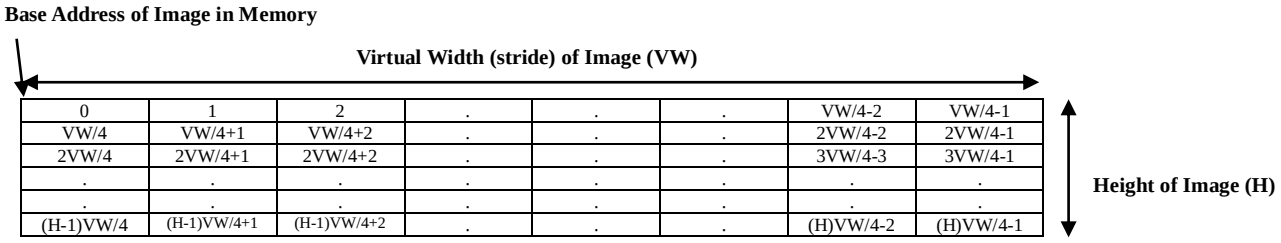
Four 1BPP with 1-bit alpha pixels are packed into a byte in memory. The format of each pixel byte is as follows:

Table 21.6 Byte Format for 1BPP with 1-bit Alpha

7	6	5	4	3	2	1	0
A3	**P3**	A2	**P2**	A1	**P1**	A0	**P0**

The Px bits are the 4 pixel bits and the Ax bits are the alpha channel bits. The alpha-channel bit is only used during drawing and pixel copy operations and is ignored by the Display Updater which transfers pixels to the panel. Ax = 0 means the pixel is transparent (invisible) and Ax = 1 means the pixel is 100% visible.

P0 is the leftmost pixel on the display and P3 is the rightmost pixel. The memory organization of the frame buffer has the top, 4 leftmost pixels (first byte) at the base address of the frame buffer. The address increases from left to right groups of 4 pixels of each line of the image and continues increasing onto the subsequent lines until the bottom right of the image, as shown below:



0	1	2	.	.	.	VW/4-2	VW/4-1
VW/4	VW/4+1	VW/4+2	.	.	.	2VW/4-2	2VW/4-1
2VW/4	2VW/4+1	2VW/4+2	.	.	.	3VW/4-3	3VW/4-1
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
(H-1)VW/4	(H-1)VW/4+1	(H-1)VW/4+2	.	.	.	(H)VW/4-2	(H)VW/4-1

Figure 21.20 Frame Buffer for 1BPP with 1-bit Alpha Image

Figure 21.21 shows an example of a 10x2 image pattern of black-and-white pixels:

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19

Assuming the alpha channel (bit) is 100% (1b) for all the pixels, the bytes in memory would be as follows:

Address	Value (binary)	Value (hex)				
0	1110 1110	EE	3	2	1	0
1	1111 0010	FA	7	6	5	4
2	*0000* 1110	0E			9	8
3	1011 1011	BB	13	12	11	10
4	1110 1011	EB	17	16	15	14
5	*0000* 1011	0B			19	18

Figure 21.21 1BPP with 1-bit Alpha Image Data Example

There are 4 pixels per byte and each line occupies 3 bytes (roundup(10/4)). The upper 4 bits of the third byte of a line are not used.

21.5.1.5 2BPP without Alpha Channel (MDCDISPCTL2.GSALPHA=0)

Four 2-bit pixels are packed into a byte in memory. The format of each pixel byte is as follows:

Table 21.7 Byte Format for 2BPP without Alpha

7	6	5	4	3	2	1	0
P3		P2		P1		P0	

P0 is the leftmost pixel on the display and P3 is the rightmost pixel. The memory organization of the frame buffer has the top, 4 leftmost pixels (first byte) at the base address of the frame buffer. The address increases from left to right groups of 4 pixels of each line of the image and continues increasing onto the subsequent lines until the bottom right of the image, as shown below:

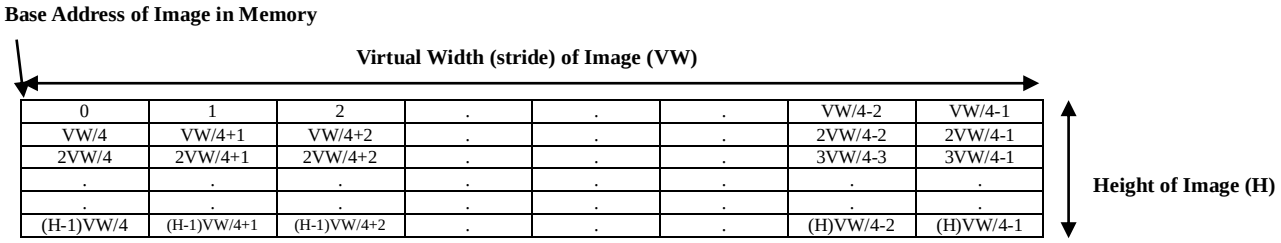


Figure 21.22 Frame Buffer for 2BPP without Alpha Image

Figure 21.23 shows an example of a 10x2 image pattern of 2BPP pixels:

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19

The bytes in memory would be as follows:

Address	Value (binary)	Value (hex)				
0	11101100	EC	3	2	1	0
1	11110100	F4	7	6	5	4
2	*0000*1100	0C			9	8
3	00110011	33	13	12	11	10
4	10000011	83	17	16	15	14
5	*0000*0011	03			19	18

Figure 21.23 2BPP without Alpha Image Data Example

There are 4 pixels per byte and each line occupies 3 bytes (roundup(10/4)). The upper 4 bits of the third byte of a line are not used.

21.5.1.6 2BPP with 2-bit Alpha Channel (MDCDISPCTL2.GSALPHA=1)

Two 2BPP with 2-bit alpha pixels are packed into a byte in memory. The format of each pixel byte is as follows:

Table 21.8 Byte Format for 2BPP with 2-bit Alpha

7	6	5	4	3	2	1	0
A1		**P1**		A0		**P0**	

The Px bits are the 2 pixel bits and the Ax bits are the alpha channel bits. The alpha-channel bits are only used during drawing and pixel copy operations and is ignored by the Display Updater which transfers pixels to the panel. The alpha value specifies the alpha-blending level between the source pixel and the background/destination pixel. Alpha value of 00b means the pixel is transparent (invisible) and alpha value of 11b means the pixel is 100% visible. Alpha value of 01b is 33% visible and alpha value of 10b is 67% visible.

P0 is the leftmost pixel on the display and P1 is the rightmost pixel. The memory organization of the frame buffer has the top, 2 leftmost pixels (first byte) at the base address of the frame buffer. The address increases from left to right groups of 2 pixels of each line of the image and continues increasing onto the subsequent lines until the bottom right of the image, as shown below:

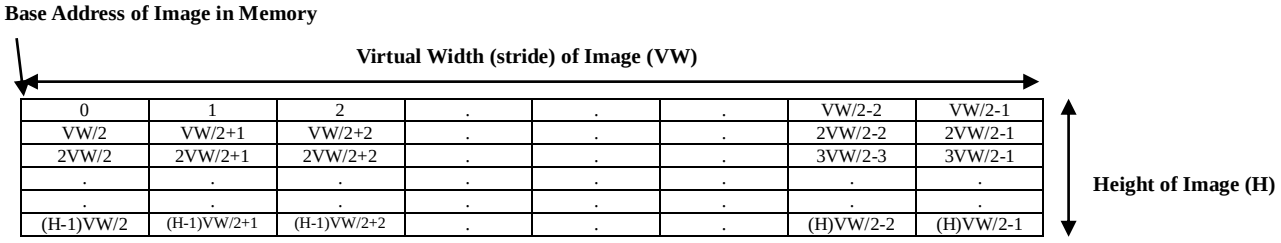


Figure 21.24 Frame Buffer for 2BPP with 2-bit Alpha Image

Figure 21.25 shows an example of a 9x2 image pattern of 2BPP pixels:

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17

Assuming the alpha channel is 100% (11b) for all the pixels, the bytes in memory would be as follows:

Address	Value (binary)	Value (hex)		
0	11111100	FC	1	0
1	11111110	FE	3	2
2	11011100	DC	5	4
3	11111111	FF	7	6
4	*0000*1100	0C		8
5	11001111	CF	10	9
6	11001111	CF	12	11
7	11111111	FF	14	13
8	11101100	EC	16	15
9	*0000*1111	0F		17

Figure 21.25 2BPP with 2-bit Alpha Image Data Example

There are 2 pixels per byte and each line occupies 5 bytes (roundup(9/2)). The upper 4 bits of the fifth byte of a line are not used.

21.5.1.7 4BPP without Alpha Channel (MDCDISPCTL2.GSALPHA=0)

Two 4BPP pixels are packed into a byte in memory. The format of each pixel byte is as follows:

Table 21.9 Byte Format for 4BPP without Alpha

7	6	5	4	3	2	1	0
P1				P0			

P0 is the leftmost pixel on the display and P1 is the rightmost pixel. The memory organization of the frame buffer has the top, 2 leftmost pixels (first byte) at the base address of the frame buffer. The address increases from left to right groups of 2 pixels of each line of the image and continues increasing onto the subsequent lines until the bottom right of the image, as shown below:

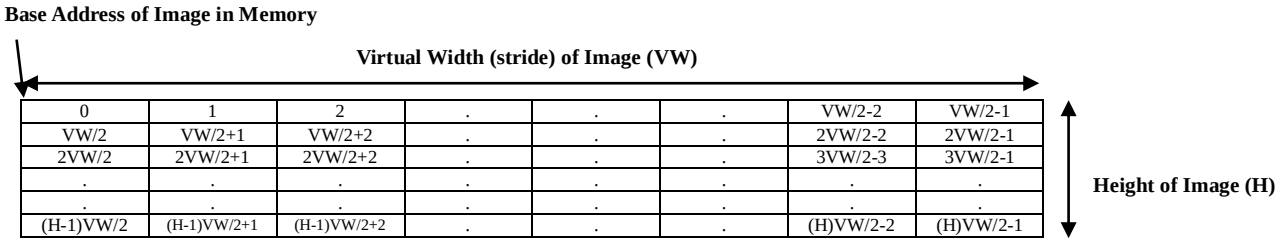


Figure 21.26 Frame Buffer for 4BPP without Alpha Image

Figure 21.27 shows an example of a 9x2 image pattern of 4BPP (black, white, 25% white, and 75% white) pixels:

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17

Assuming 25% white pixel value is 0100b and 75% white pixel value is 1100b, the bytes in memory would be as follows:

Address	Value (binary)	Value (hex)		
0	11110000	F0	1	0
1	11111100	FC	3	2
2	01000000	40	5	4
3	11111111	FF	7	6
4	*0000*0000	00		8
5	00001111	0F	10	9
6	00001111	0F	12	11
7	11111111	FF	14	13
8	11000000	C0	16	15
9	*0000*1111	0F		17

Figure 21.27 4BPP without Alpha Image Data Example

There are 2 pixels per byte and each line occupies 5 bytes (roundup(9/2)). The upper 4 bits of the fifth byte of a line are not used.

21.5.1.8 4BPP with 4-bit Alpha Channel (MDCDISPCTL2.GSALPHA=1)

One 4BPP with 4-bit alpha pixel occupies a byte in memory. The format of each pixel byte is as follows:

Table 21.10 Byte Format for 4BPP with 4-bit Alpha

7	6	5	4	3	2	1	0
A0				**P0**			

The Px bits are the pixel bits and the Ax bits are the alpha channel bits. The alpha-channel bits are only used during drawing and pixel copy operations and is ignored by the Display Updater which transfers pixels to the panel. The alpha value specifies the alpha-blending level between the source pixel and the background/destination pixel. Alpha value of 0000b means the pixel is transparent (invisible) and alpha value of 1111b means the pixel is 100% visible. Alpha value of 0100b is 25% visible and alpha value of 1100b is 75% visible.

The memory organization of the frame buffer has the top, leftmost pixel (first byte) at the base address of the frame buffer. The address increases from left to right of each line of the image and continues increasing onto the subsequent lines until the bottom right of the image, as shown below:

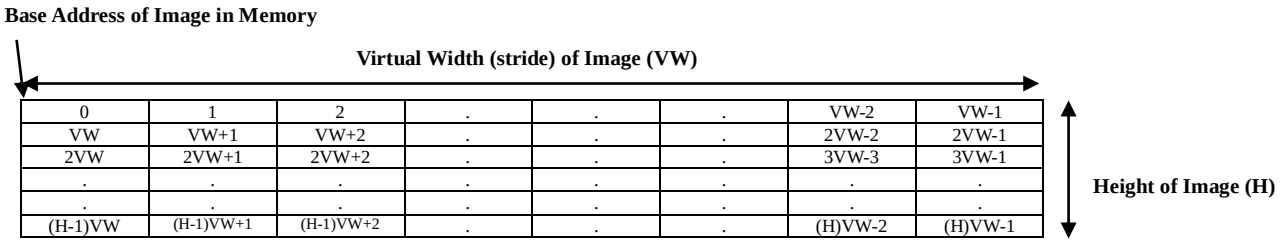


Figure 21.28 Frame Buffer for 4BPP with 4-bit Alpha Image

Figure 21.29 shows an example of a 5x2 image pattern of 4BPP (black, white, 25% white, and 75% white) pixels:

0	1	2	3	4
5	6	7	8	9

Assuming alpha value is 100% (1111b), 25% white pixel value is 0100b, and 75% white pixel value is 1100b, the bytes in memory would be as follows:

Address	Value (binary)	Value (hex)	
0	11110000	F0	0
1	11111111	FF	1
2	11111100	FC	2
3	11111111	FF	3
4	11110000	F0	4
5	11111111	FF	5
6	11110000	F0	6
7	11111111	FF	7
8	11110100	F4	8
9	11111111	FF	9

Figure 21.29 4BPP with 4-bit Alpha Image Data Example

There is 1 pixel per byte and each line occupies 5 bytes.

21.5.1.9 8BPP without Alpha Channel (MDCDISPCTL2.GSALPHA=0)

One 8BPP pixel occupies a byte in memory. The format of each pixel byte is as follows:

Table 21.11 Byte Format for 8BPP without Alpha

7	6	5	4	3	2	1	0
P0							

The memory organization of the frame buffer has the top, leftmost pixel (first byte) at the base address of the frame buffer. The address increases from left to right of each line of the image and continues increasing onto the subsequent lines until the bottom right of the image, as shown below:

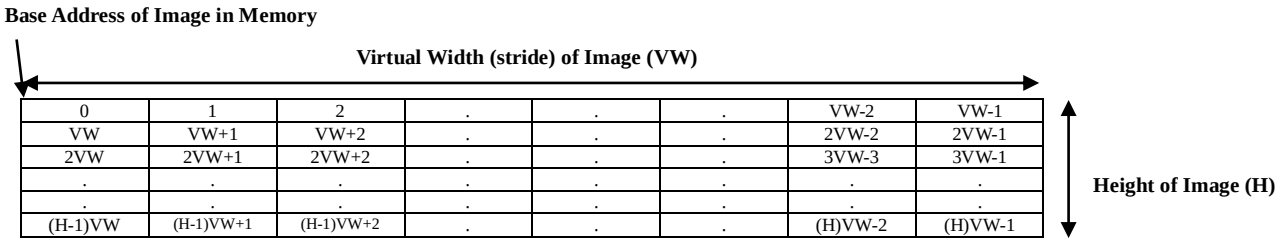


Figure 21.30 Frame Buffer for 8BPP without Alpha Image

Figure 21.31 shows an example of a 5x2 image pattern of 8BPP (black, white, 25% white, and 75% white) pixels:

0	1	2	3	4
5	6	7	8	9

Assuming 25% white pixel value is 01000000b, and 75% white pixel value is 11000000b, the bytes in memory would be as follows:

Address	Value (binary)	Value (hex)	
0	00000000	00	0
1	11111111	FF	1
2	11000000	C0	2
3	11111111	FF	3
4	00000000	00	4
5	11111111	FF	5
6	00000000	00	6
7	11111111	FF	7
8	01000000	40	8
9	11111111	FF	9

Figure 21.31 8BPP without Alpha Image Data Example

There is 1 pixel per byte and each line occupies 5 bytes.

21.5.1.10 8BPP with 8-bit Alpha Channel (MDCDISPCTL2.GSALPHA=1)

One 8BPP with 8-bit alpha pixel occupies two bytes in memory. The format of each 2-byte pixel is as follows:

Table 21.12 2-Byte Format for 8BPP with 8-bit Alpha

15	14	13	12	11	10	9	8
A0							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P0							

The memory organization of the frame buffer has the top, leftmost pixel (first byte) at the base address of the frame buffer. The address increases from left to right of each line of the image and continues increasing onto the subsequent lines until the bottom right of the image, as shown below:

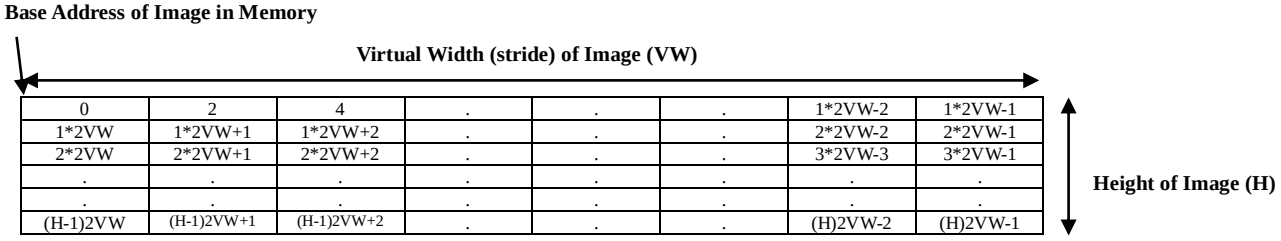


Figure 21.32 Frame Buffer for 8BPP with 8-bit Alpha Image

Figure 21.33 shows an example of a 5x2 image pattern of 8BPP (black, white, 25% white, and 75% white) pixels:

0	1	2	3	4
5	6	7	8	9

Assuming alpha value is 100% (11111111b), 25% white pixel value is 01000000b, and 75% white pixel value is 11000000b, the bytes in memory would be as follows:

Address	Value (binary)	Value (hex)	
0	00000000	00	0
1	11111111	FF	
2	11111111	FF	1
3	11111111	FF	
4	11000000	C0	2
5	11111111	FF	
6	11111111	FF	3
7	11111111	FF	
8	00000000	00	4
9	11111111	FF	
10	11111111	FF	5
11	11111111	FF	
12	00000000	00	6
13	11111111	FF	
14	11111111	FF	7
15	11111111	FF	
16	01000000	40	8
17	11111111	FF	
18	11111111	FF	9
19	11111111	FF	

Figure 21.33 8BPP with 8-bit Alpha Image Data Example

There is 1 pixel per 2 bytes and each line occupies 10 bytes.

21.5.2 Bitmap Formats

For the Copy Engine, which copies pixels from a source image location to a destination image location, the MDCGFXCTRL.BITMAPEN bit specifies if the source image for copying is a regular image (=0) or bitmap (=1). When bitmap is enabled (MDCGFXCTRL.BITMAPEN=1), and the MDCGFXCTRL.BITMAPFMT bit specifies 1-bit or 2-bit per pixel bitmap format.

21.5.2.1 1-Bit Bitmap

For 1-bit bitmap format, each bit specifies whether or not the pixel is copied to the destination image. If the bit is 0, no pixel is copied. If the bit is 1, the pixel is copied. The pixel color and alpha blending for the copied pixel are specified by the MDCGFXCOLOR register, the MDCGFXCTL.ALPHAOVR bit, and the MDCGFXCTL.ALPHAVAL[1:0] bits as shown in the following table:

Table 21.13 1-Bit Bitmap Pixel Color and Alpha Blending

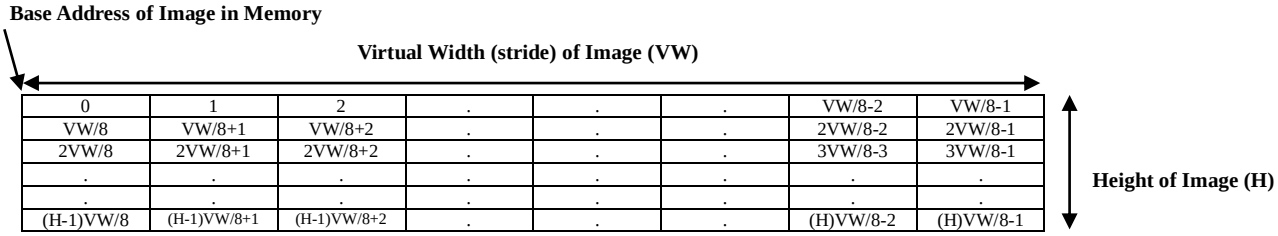
Image Data Format	MDCGFXCTL. ALPHAOVR	Pixel Color	Alpha Value
6-Bit Color (2-bit Alpha)	0	MDCGFXCOLOR[5:0]	MDCGFXCOLOR[7:6]
	1		MDCGFXCTL.ALPHAVAL[1:0]
3-Bit Color (1-bit Alpha)	-	MDCGFXCOLOR[2:0]	(100%)
1BPP EPD/Grayscale, GSALPHA=0	-	MDCGFXCOLOR[0]	(100%)
1BPP EPD/Grayscale, GSALPHA=1	-		(100%)
2BPP EPD/Grayscale, GSALPHA=0	-	MDCGFXCOLOR[1:0]	(100%)
2BPP EPD/Grayscale, GSALPHA=1	0	MDCGFXCOLOR[1:0]	MDCGFXCOLOR[3:2]
2BPP EPD/Grayscale, GSALPHA=1	1		MDCGFXCTL.ALPHAVAL[1:0]
4BPP EPD/Grayscale, GSALPHA=0	-	MDCGFXCOLOR[3:0]	(100%)
4BPP EPD/Grayscale, GSALPHA=1	0	MDCGFXCOLOR[3:0]	MDCGFXCOLOR[7:4]
4BPP EPD/Grayscale, GSALPHA=1	1	MDCGFXCOLOR[3:0]	{MDCGFXCTL.ALPHAVAL[1:0], MDCGFXCTL.ALPHAVAL[1:0]}
8BPP EPD/Grayscale, GSALPHA=0	-	MDCGFXCOLOR[7:0]	(100%)
8BPP EPD/Grayscale, GSALPHA=1	-		{MDCGFXCTL.ALPHAVAL[1:0], MDCGFXCTL.ALPHAVAL[1:0], MDCGFXCTL.ALPHAVAL[1:0], MDCGFXCTL.ALPHAVAL[1:0]}

For the 1-bit bitmap, eight 1-bit pixels are packed into a byte in memory. The format of each byte is as follows:

Table 21.14 1-Bit Bitmap Byte Format

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

Px = 0 means the pixel is not written to the destination and Px = 1 means the pixel is written. P0 is the leftmost pixel on the display and P7 is the rightmost pixel. The memory organization of the bitmap image has the top, eight leftmost pixel data (first byte) at the base address of the bitmap image. The address increases from left to right groups of eight pixels of each line of the image and continues increasing onto the subsequent lines until the bottom right of the image, as shown below:



0	1	2	.	.	.	VW/8-2	VW/8-1
VW/8	VW/8+1	VW/8+2	.	.	.	2VW/8-2	2VW/8-1
2VW/8	2VW/8+1	2VW/8+2	.	.	.	3VW/8-3	3VW/8-1
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
(H-1)VW/8	(H-1)VW/8+1	(H-1)VW/8+2	.	.	.	(H)VW/8-2	(H)VW/8-1

Figure 21.34 Frame Buffer for 1-bit Bitmap Image

Figure 21.35 shows an example of a 10 × 11 1-bit bitmap image for character '8'.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	31	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109

The shaded pixels have bit value 0 (transparent, pixel not copied to destination) and the white pixels have bit value 1 (pixel is copied to destination). In this example, the bytes in memory would be as follows:

Address	Value (binary)	Value (hex)								
0	0b01111000	0x78	7	6	5	4	3	2	1	0
1	0b*000000*00	0x00							9	8
2	0b11001100	0xCC	17	16	15	14	13	12	11	10
3	0b*000000*00	0x00							19	18
4	0b10000100	0x84	27	26	25	24	23	22	21	20
5	0b*000000*00	0x00							29	28
6	0b10000100	0x84	37	36	35	34	33	32	31	30
7	0b*000000*00	0x00							39	38
8	0b11111100	0xFC	47	46	45	44	43	42	41	40
9	0b*000000*00	0x00							49	48
10	0b01111000	0x78	57	56	55	54	53	52	51	50
11	0b*000000*00	0x00							59	58
12	0b11111100	0xFC	67	66	65	64	63	62	61	60
13	0b*000000*00	0x00							69	68
14	0b10000100	0x84	77	76	75	74	73	72	71	70
15	0b*000000*00	0x00							79	78
16	0b10000100	0x84	87	86	85	84	83	82	81	80
17	0b*000000*00	0x00							89	88
18	0b11001100	0xCC	97	96	95	94	93	92	91	90
19	0b*000000*00	0x00							99	98
20	0b01111000	0x78	107	106	105	104	103	102	101	100
21	0b*000000*00	0x00							109	108

Figure 21.35 1-bit Bitmap Image Data Example

There are 8 pixels per byte and each line occupies 2 bytes (roundup(10/8)). The upper 6 bits of the second byte of a line are not used.

21.5.2.2 2-Bit Bitmap

For 2-bit bitmap format, if the 2-bit bitmap pixel value is 00b, the pixel is not copied to the destination image. If 2-bit bitmap pixel value is not 00b, the pixel color and alpha blending for the copied pixel are specified by the MDCGFXCOLOR register, the MDCGFXCTL.ALPHAOVR bit, and the MDCGFXCTL.ALPHAVAL[1:0] bits as shown in the following table:

Table 21.15 2-Bit Bitmap Pixel Color and Alpha Blending

Image Data Format	MDCGFXCTL. ALPHAOVR	Pixel Color	Alpha Value
6-Bit Color (2-bit Alpha)	0	MDCGFXCOLOR[5:0]	2-bit bitmap pixel value
	1		MDCGFXCTL.ALPHAVAL[1:0]
3-Bit Color (1-bit Alpha)	-	MDCGFXCOLOR[2:0]	(100%)
1BPP EPD/Grayscale, GSALPHA=0	-	MDCGFXCOLOR[0]	(100%)
1BPP EPD/Grayscale, GSALPHA=1	-		(100%)
2BPP EPD/Grayscale, GSALPHA=0	-	MDCGFXCOLOR[1:0]	(100%)
2BPP EPD/Grayscale, GSALPHA=1	0	MDCGFXCOLOR[1:0]	2-bit bitmap pixel value
2BPP EPD/Grayscale, GSALPHA=1	1		MDCGFXCTL.ALPHAVAL[1:0]
4BPP EPD/Grayscale, GSALPHA=0	-	MDCGFXCOLOR[3:0]	(100%)
4BPP EPD/Grayscale, GSALPHA=1	0	MDCGFXCOLOR[3:0]	{2-bit bitmap pixel value, 2-bit bitmap pixel value}
4BPP EPD/Grayscale, GSALPHA=1	1	MDCGFXCOLOR[3:0]	{MDCGFXCTL.ALPHAVAL[1:0], MDCGFXCTL.ALPHAVAL[1:0]}
8BPP EPD/Grayscale, GSALPHA=0	-	MDCGFXCOLOR[7:0]	(100%)
8BPP EPD/Grayscale, GSALPHA=1	0		{2-bit bitmap pixel value, 2-bit bitmap pixel value, 2-bit bitmap pixel value, 2-bit bitmap pixel value}
8BPP EPD/Grayscale, GSALPHA=1	1		{MDCGFXCTL.ALPHAVAL[1:0], MDCGFXCTL.ALPHAVAL[1:0], MDCGFXCTL.ALPHAVAL[1:0], MDCGFXCTL.ALPHAVAL[1:0]}

For the 2-bit bitmap format, four 2-bit pixels are packed into a byte in memory. The format of each byte is as follows:

Table 21.16 2-Bit Bitmap Byte Format

7	6	5	4	3	2	1	0
P3[1]	P3[0]	P2[1]	P2[0]	P1[1]	P1[0]	P0[1]	P0[0]

Each Px[1:0] value specifies the 2-bit bitmap value for the pixel. P0[1:0] is the leftmost pixel of the bitmap image and P3[1:0] is the rightmost pixel. The memory organization of the bitmap image has the top, four leftmost pixel data (first byte) at the base address of the bitmap image. The address increases from left to right groups of four pixels of each line of the image and continues increasing onto the subsequent lines until the bottom right of the image, as shown below:

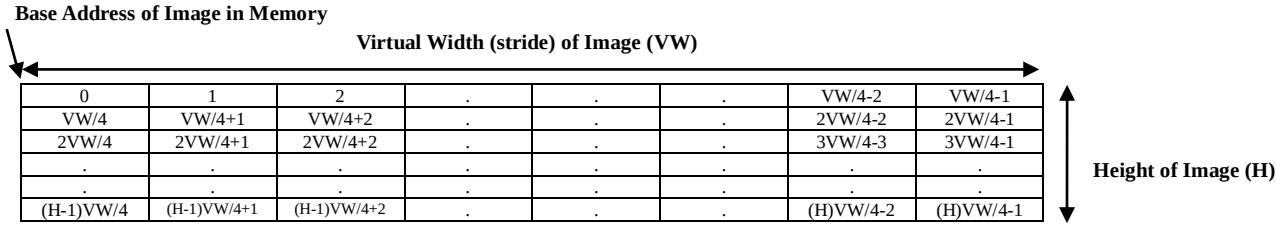
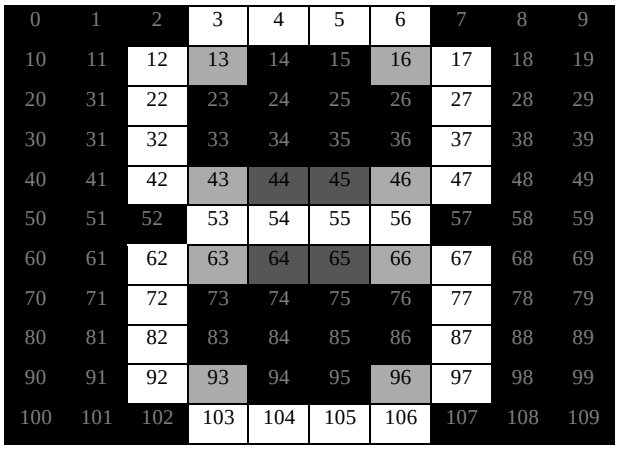


Figure 21.36 Frame Buffer for 1-bit Bitmap Image

Figure 21.37 shows an example of a 10 × 11 2-bit bitmap image for character '8' for 2BPP grayscale with GSALPHA=1 (alpha enabled). The MDCGFXCOLOR value is 3 (white).



The white pixels have an alpha value of 0b11, the light gray pixels have an alpha value of 0b10, the dark gray pixels have an alpha value of 0b01, and the black pixels have an alpha value of 0b00 (transparent). In this example, the bytes in memory would be as follows:

Address	Value (binary)	Value (hex)				
0	0b11000000	C0	3	2	1	0
1	0b00111111	3F	7	6	5	4
2	0b*0000*0000	00			9	8
3	0b10110000	B0	13	12	11	10
4	0b11100000	E0	17	16	15	14
5	0b*0000*0000	00			19	18
6	0b00110000	30	23	22	21	20
7	0b11000000	C0	27	26	25	24
8	0b*0000*0000	00			29	28
9	0b00110000	30	33	32	31	30
10	0b11000000	C0	37	36	35	34
11	0b*0000*0000	00			39	38
12	0b10110000	B0	43	42	41	40
13	0b11100101	E5	47	46	45	44
14	0b*0000*0000	00			49	48
15	0b11000000	C0	53	52	51	50
16	0b00111111	3F	57	56	55	54
17	0b*0000*0000	00			59	58
18	0b10110000	B0	63	62	61	60
19	0b11100101	E5	67	66	65	64
20	0b*0000*0000	00			69	68
21	0b00110000	30	73	72	71	70
22	0b11000000	C0	77	76	75	74
23	0b*0000*0000	00			79	78
24	0b00110000	30	83	82	81	80
25	0b11000000	C0	87	86	85	84
26	0b*0000*0000	00			89	88
27	0b10110000	B0	93	92	91	90
28	0b11100000	E0	97	96	95	94
29	0b*0000*0000	00			99	98
30	0b11000000	C0	103	102	101	100
31	0b00111111	3F	107	106	105	104
32	0b*0000*0000	00			109	108

Figure 21.37 2-bit Bitmap Image Data Example

21.6 Operations

21.6.1 Initialization

The MDC should be initialized with the procedure shown below.
1. Configure the VMD power supply. (Refer to "Voltage Booster" section.)
2. For other initial settings, refer to the descriptions for each MDC function.

21.6.2 Display Updater

The display updater handles the transfer of pixel data from the frame buffer to the display panel. The display updater supports four main types of interfaces: 6-bit color panel, SPI panel, Grayscale panels (3 types of interfaces), EPD panels (2 types of interfaces). For 6-bit color, SPI, and EPD panels, the image can be rotated in 0, 90, 180, or 270 degree during data transfer. For example, a 160×120-pixel image can be transferred to the 120×160-pixel display panel by rotating in 90 or 270 degree.

The following table shows the control register bits and panel interface types selected:

Table 21.17 Panel Interface Types Selection

Panel Type	MDCDISPCTL. DISPEPD	MDCDISPCTL. DISPGS	MDCDISPCTL. DISPSPI	MDCDISPCTL. SPITYPE	MDCDISPCTL. GSIF[1:0]
6-Bit Color	0	0	0	-	-
1-Bit B&W SPI	0	0	1	0	-
3-Bit Color SPI	0	0	1	1	-
Grayscale 8-bit Parallel	0	1	-	-	0xb
Grayscale 3-Wire Serial	0	1	-	-	10b
Grayscale 4-Wire Serial	0	1	-	-	11b
EPD 3-Wire Serial	1	-	-	-	01b
EPD 4-Wire Serial	1	-	-	-	10b

The following table shows the panel interface pins and their function for the different panel interfaces:

Table 21.18 Panel Interface Pin Functions

Pin Name	Panel Interface						
	6-Bit Color	SPI	Grayscale 8-Bit Parallel	Grayscale 3-Wire Serial	Grayscale 4-Wire serial	EPD 3-Wire Serial	EPD 4-Wire Serial
XRST	XRST	-	A0	-	A0	-	-
VST	VST	SCLK	XRD	SCL	SCL	SCL	SCL
HST	HST	SCS	XCS	XCS	XCS	CSB	CSB
ENB	ENB	SI	XWR	SD	SD	SDA	SDA
VCK	VCK	-	D0	-	-	-	DC
HCK	HCK	-	D1	-	-	-	-
RED0	RED0	-	D2	-	-	-	-
RED1	RED1	-	D3	-	-	-	-
GRN0	GRN0	-	D4	-	-	-	-
GRN1	GRN1	-	D5	-	-	-	-
BLU0	BLU0	-	D6	-	-	-	-
BLU1	BLU1	-	D7	-	-	-	-
VCOM	VCOM	EXTCOMIN	-	-	-	-	-
XFRP	XFRP	-	-	-	-	-	-

21.6.2.1 6-bit Color Updater

Section 9.3.1 shows the timing diagram and parameters for the 6-bit color panel interface. As shown in Section 9.3.1, the 6-bit color updater provides flexibility in programming the panel interface timing by setting the MDCDISP*.TIM* and MDCDISPCTL2 registers.

The 6-bit color updater also supports partial updates of the panel by specifying the start and end line numbers (MDCDISPSTARTY and MDCDISPENDY registers) to update (first line is line 0). During a frame update, the lines that are updated (between the specified start and end line numbers inclusive) will only have VCK toggling with normal timing, while ENB, HCK, HST, RED[1:0], GRN[1:0], and BLU[1:0] all stay at LOW level. Fast VCK mode is supported through the MDCDISPCTL2.FASTVCK bit. If the FASTVCK bit is 0, the VCK HIGH/LOW pulse width is normal horizontal line timing and is the same for updated and none-updated lines. However, if the FASTVCK bit is 1, the VCK HIGH/LOW pulses can be shorted and the pulse with is specified by the MDCDISPPRM1413.TIM13[7:0] register.

6-bit color panel updater initialization procedure

To set up the 6-bit color panel, the following register initialization sequence is needed:

1. Set the MDCDISPVCOMDIV.DISPVCOMDIV[15:0] bits. (Set VCOM period)
2. Configure the following MDCDISPCTL register bits:
   - Set MDCDISPCTL.DISPEPD bit to 0. (Deselect EPD panel interface)
   - Set MDCDISPCTL.DISPGS bit to 0. (Deselect grayscale panel interface)
   - Set MDCDISPCTL.DISPSPI bit to 0. (Select 6-bit color panel interface)
   - Set MDCDISPCTL.ROTSEL[1:0] bits value. (Select swivel orientation)
   - Set the MDCDISPCTL.VCOMEN bit to 1. (Turn VCOM/XFRP output on)
   - Set MDCDISPCTL.DISPINVERT bit value. (Enable/disable pixel inversion)
3. Configure the following MDCDISPCTL2 register bits:
   - Set MDCDISPCTL2.FASTVCK bit to 0. (Set normal VCK for partial update)
   - Set MDCDISPCTL2.ENBPHASE bit value. (Select ENB phase for panel)
   - Set MDCDISPCTL2.HSTFALL bit value. (Select HST fall position/timing)
   - Set MDCDISPCTL2.VSTFALL bit value. (Select VST fall position/timing)
4. Configure the following MDCDISPCLKDIV register bits:
   - MDCDISPCLKDIV.CLKDIV[7:0] bits (Set panel timing unit (T))
   - MDCDISPCLKDIV.TIM0[7:0] bits (Set VCK rise to HST rise time and VCK fall to VST fall time)
5. Configure the following MDCDISPPRM21 register bits:
   - MDCDISPPRM21.TIM1[7:0] bits (Set VST rise to VCK rise time)
   - MDCDISPPRM21.TIM2[7:0] bits (Set HST rise to HCK rise time)
6. Configure the following MDCDISPPRM43 register bits:
   - MDCDISPPRM43.TIM3[7:0] bits (Set Data to HCK rise/fall time)
   - MDCDISPPRM43.TIM4[7:0] bits (Set VCK rise/fall to ENB rise time)
7. Configure the following MDCDISPPRM65 register bits:
   - MDCDISPPRM65.TIM5[7:0] bits (Set ENB high time)
   - MDCDISPPRM65.TIM6[7:0] bits (Set XRST rise delay time, XRST rise to VST rise time, and data transfer end to XRST fall time)
8. Configure the following MDCDISPPRM87 register bits:
   - MDCDISPPRM87.TIM7[7:0] bits (Set HCK rise/fall to Data and HCK rise/fall to HST fall time)
   - MDCDISPPRM87.TIM8[7:0] bits (Set end of line to next VCK rise/fall time)
9. Configure the following MDCDISPPRM109 register bits:
   - MDCDISPPRM109.TIM9[7:0] bits (Set HCK count for start of pixel data)
   - MDCDISPPRM109.TIM10[7:0] bits (Set number of extra HCK counts after end of line pixel data)

10. Configure the following MDCDISPPRM1211 register bits:
    - MDCDISPPRM1211.TIM11[7:0] bits (Set VCK count for start of pixel data)
    - MDCDISPPRM1211.TIM12[7:0] bits (Set number of extra VCK counts after last line of pixel data)
11. Configure the following MDCDISPPRM1413 register bits:
    - MDCDISPPRM1413.TIM13[7:0] bits (Set VCK high/low width for fast VCK partial update)
12. Set the MDCDISPWIDTH.DISPWIDTH[9:0] bits. (Set display width)
13. Set the MDCDISPHEIGHT.DISPHEIGHT[9:0] bits. (Set display height)
14. Set the MDCDISPFRMBUFF0.FRMBUFFADDR[15:0] and MDCDISPFRMBUFF1.FRMBUFFADDR[31:16] bits. (Set frame buffer base address)
15. Set the MDCDISPSTRIDE.DISPSTRIDE[9:0] bits. (Set frame buffer stride)
    The frame buffer image can be wider than the display width to allow for panning.
16. Set the MDCDISPSTARTY.STARTY[9:0] bits. (Set display update starting line number)
17. Set the MDCDISPENDY.ENDY[9:0] bits. (Set display update ending line number)

Triggering a display update

To update the display, follow the procedure shown below.

1. Write 1 to the MDCBSTPWR.VMDBUP bit. (Increase response speed)
2. For a partial update, set the following:
    - Set MDCDISPSTARTY and MDCDISPENDY registers (Start and end lines)
    - Set MDCDISPCTL2.FASTVCK bit (Select normal or fast VCK)
3. Set the following bits when using the interrupt:
    - Write 1 to the MDCINTCTL.UPDIF bit. (Clear interrupt flag)
    - Set the MDCINTCTL.UPDIE bit to 1. (Enable MDC interrupt)
4. Write 1 to the MDCTRIGCTL.UPDTRIG bit. (Start display update)

When the update is finished, the MDCTRIGCTL.UPDTRIG bit is automatically cleared to 0 and the MDCINTCTL.UPDIF bit is set to 1. After that, set the MDCBSTPWR.VMDBUP bit to 0, and the MDCBSTPWR.REGECO bit to 1.

21.6.2.2 SPI Updater

Section 9.3.2 shows the timing diagram and parameters for the SPI panel interface. As shown in Section 9.3.2, the SPI updater provides flexibility in programming the panel interface timing by setting the MDCDISP*.TIM* and MDCDISPCTL2 registers.

SPI panels have two input signals for controlling the COM signal: EXTMODE and EXTCOM. If EXTMODE=1, a COM signal needs to be connected to the EXTCOM input of the panel. The VCOM output of the S1D13C00 can be connected to EXTCOM. If EXTMODE=0, the MDC has to periodically send a dummy command to toggle the internal COM bit of the panel when the there are no updates to the display. In this case, there is a control register bit, MDCDISPCTL.AUTOCOM, which can be used to automatically send the dummy commands to periodically toggle the COM bit (MODE[1] bit of the command sent to the panel) on every rising edge of the VCOM output.

To set up the SPI panel, the following register initialization sequence is needed:

1. Set the MDCDISPVCOMDIV.DISPVCOMDIV[15:0] bits. (Set VCOM period)
2. Configure the following MDCDISPCTL register bits:
   - Set MDCDISPCTL.DISPEPD bit to 0. (Deselect EPD panel interface)
   - Set MDCDISPCTL.DISPGS bit to 0. (Deselect grayscale panel interface)
   - Set the MDCDISPCTL.DISPSPI bit to 1. (Select SPI panel interface)
   - Set MDCDISPCTL.SPITYPE bit value. (Select SPI panel type: 1-bit black-and-white or 3-bit color)
   - Set MDCDISPCTL.ROTSEL[1:0] bits value. (Select swivel orientation)
   - Set the MDCDISPCTL.VCOMEN bit to 1. (Turn COM output on)
   - Set MDCDISPCTL.DISPINVERT bit value. (Enable/disable pixel inversion)
   - Set MDCDISPCTL.ADDRLSB bit value. (Select address bits order (LSB/MSB first))
   - Set MDCDISPCTL.AUTOMCOM bit to inverse of EXTMODE input of SPI panel.

   If 3-bit color is selected, configure the following bit as well:
   - Set MDCDISPCTL.RGBORD bit value. (Select color bits order (R/B first))
3. Set the MDCDISPCTL2.CSPOL bit to 0. (Configure active-high chip-select)
4. Configure the following MDCDISPCLKDIV register bits:
   - MDCDISPCLKDIV.CLKDIV[7:0] bits (Set SCLK frequency)
   - MDCDISPCLKDIV.TIM0[7:0] bits (Set SCS rise to SCLK rise time)
5. Configure the following MDCDISPPRM21 register bits:
   - MDCDISPPRM21.TIM1[7:0] bits (Set SCLK fall to SCS fall time)
   - MDCDISPPRM21.TIM2[7:0] bits (Set data transfer period (number of SCLK cycles))
6. Configure the following MDCDISPPRM43 register bits:
   - MDCDISPPRM43.TIM3[7:0] bits (Set Mode bits value)
   - MDCDISPPRM43.TIM4[2:0] bits (Set number of Mode bits to send)
7. Configure the following MDCDISPPRM65 register bits:
   - MDCDISPPRM65.TIM5[3:0] bits (Set number of Address bits to send)
   - MDCDISPPRM65.TIM6[7:0] bits (Set SCS fall to next SCS rise time)
8. Set the MDCDISPWIDTH.DISPWIDTH[9:0] bits. (Set display width)
9. Set the MDCDISPHEIGHT.DISPHEIGHT[9:0] bits. (Set display height)
10. Set the MDCDISPFRMBUFF0.FRMBUFFADDR[15:0] and MDCDISPFRMBUFF1.FRMBUFFADDR[31:16] bits. (Set frame buffer base address)
11. Set the MDCDISPSTRIDE.DISPSTRIDE[9:0] bits. (Set frame buffer stride)
    The frame buffer image can be wider than the display width to allow for panning.
12. Set the MDCDISPSTARTY.STARTY[9:0] bits. (Set display update starting line number)
13. Set the MDCDISPENDY.ENDY[9:0] bits. (Set display update ending line number)

Note: SCLK high width will be one t_{SYSCLK} longer than SCLK low width when the MDCDISPCLKDIV.CLKDIV[7:0] register is an even value.

Triggering a display update

To update the display, follow the procedure shown below.

1. Write 1 to the MDCBSTPWR.VMDBUP bit. (Increase response speed)
2. Set the following bits when using the interrupt:
   - Write 1 to the MDCINTCTL.UPDIF bit. (Clear interrupt flag)
   - Set the MDCINTCTL.UPDIE bit to 1. (Enable MDC interrupt)
3. Write 1 to the MDCTRIGCTL.UPDTRIG bit. (Start display update)

When the update is finished, the MDCTRIGCTL.UPDTRIG bit is automatically cleared to 0 and the MDCINTCTL.UPDIF bit is set to 1. After that, set the MDCBSTPWR.VMDBUP bit to 0, and the MDCBSTPWR.REGECO bit to 1.

21.6.2.3 Grayscale Panel Updater

Section 9.3.3 shows the timing diagram and parameters for the grayscale panel interfaces. As shown in Section 9.3.3, the grayscale updater provides flexibility in programming the panel interface timing by setting the MDCDISP*.TIM* and MDCDISPCTL2 registers.

There are 3 types of grayscale panel interfaces: 8-bit Parallel, 3-wire Serial, and 4-wire Serial. All three interfaces have the same byte sequence for writing to registers and display memory of the panel. A command byte is indicated by A0 = 0 and a parameter byte is indicated by A0 = 1. A command sequence starts with A0 = 0 and a command byte followed by parameter bytes with A0 = 1. The number of parameter bytes depends on the command.

Figure 21.38 and Figure 21.39 show the byte sequence and the timing diagrams of the grayscale panel interface types, respectively.

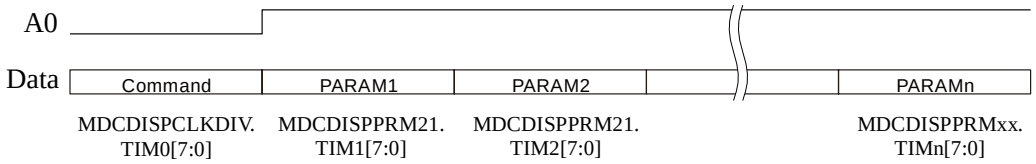


Figure 21.38 Byte Sequence for Grayscale Panel

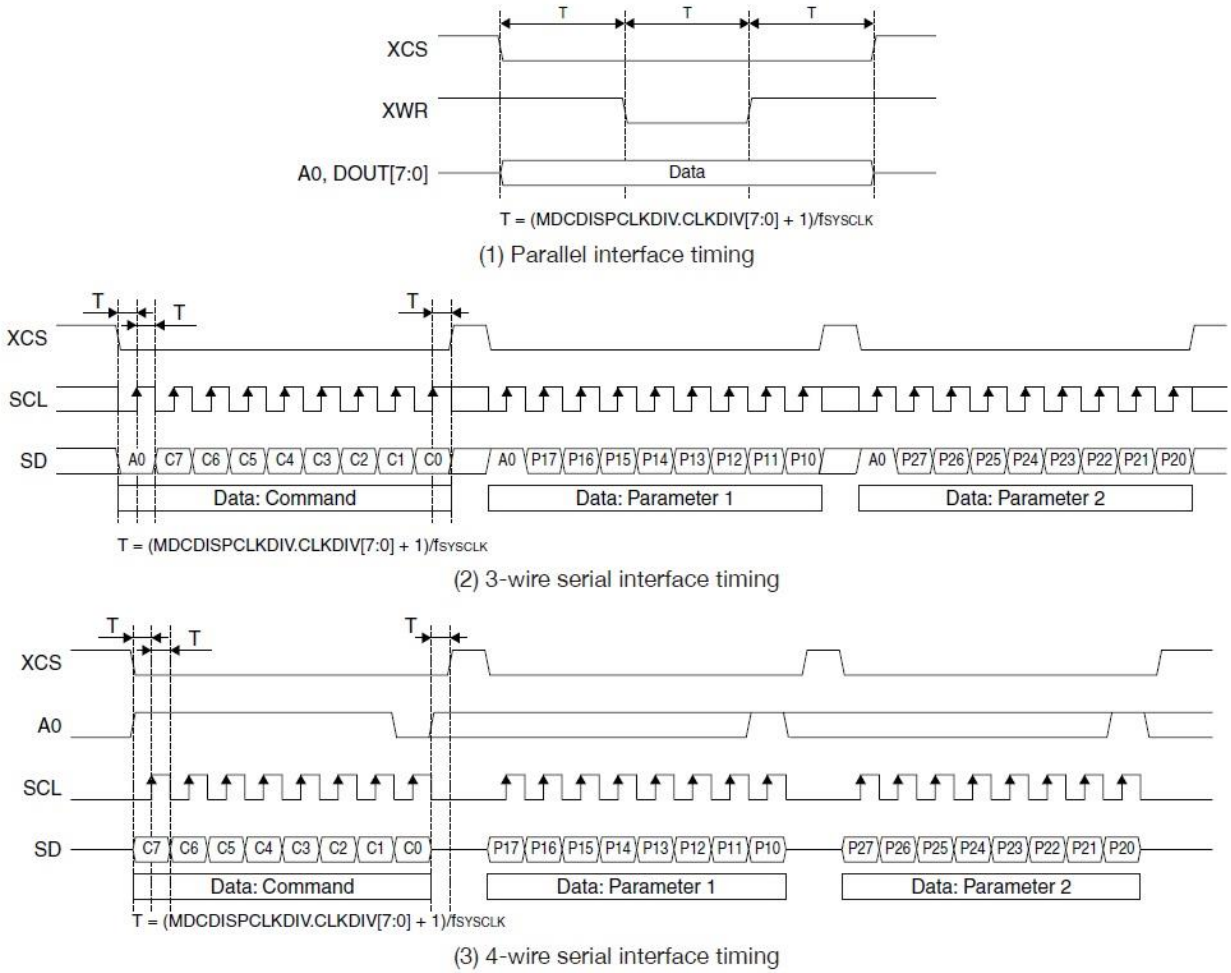


Figure 21.39 Grayscale Panel Timing Diagrams

Grayscale panel updater initialization procedure

To set up the grayscale panel, the following register initialization sequence is needed:

1. Configure the following MDCDISPCTL register bits:
   - Set MDCDISPCTL.DISPEPD bit to 0. (Deselect EPD panel interface)
   - Set MDCDISPCTL.DISPGS bit to 1. (Select grayscale panel interface)
   - Set MDCDISPCTL.GSIF[1:0] bits value. (Select grayscale panel interface type: 8-bit Parallel, 3-wire Serial, or 4-wire Serial)
   - Set the MDCDISPCTL.VCOMEN bit to 0. (VCOM output is not used)
   - Set MDCDISPCTL.DISPINVERT bit value. (Enable/disable pixel inversion)
2. Configure the following MDCDISPCLKDIV register bits:
   - MDCDISPCLKDIV.CLKDIV[7:0] bits (Set panel timing unit T) T = [S x (CLKDIV[7:0] + 1)] where S is the system clock period
3. Configure MDCTRIGCTRL.GSBPP[1:0] grayscale format (Select 1, 2, 4, or 8 BPP)
4. Configure the following MDCDISPCTL2 register bits:
   - Set MDCDISPCTL2.CSPOL bit to 0. (Configure active-low chip-select)
   - MDCDISPCTL2.GSALPHA bit (Disable/enable alpha channel for the selected BPP format)
5. Set the MDCDISPWIDTH.DISPWIDTH[9:0] bits. (Set display width)
6. Set the MDCDISPHEIGHT.DISPHEIGHT[9:0] bits. (Set display height)
7. Set the MDCDISPFRMBUFF0.FRMBUFFADDR[15:0] and MDCDISPFRMBUFF1.FRMBUFFADDR[31:16] bits. (Set frame buffer base address)
8. Set the MDCDISPSTRIDE.DISPSTRIDE[9:0] bits. (Set frame buffer stride)
   The frame buffer image can be wider than the display width to allow for panning.
9. Set the MDCDISPSTARTY.STARTY[9:0] bits. (Set display update starting line number)
10. Set the MDCDISPENDY.ENDY[9:0] bits. (Set display update ending line number)

Sending a command

To send a command to the grayscale panel, follow the procedure shown below.

1. Set the command and parameters to be sent to the panel to the following registers/bits:
   - MDCDISPCLKDIV.TIM0[7:0] (Command byte)
   - MDCDISPPRM21.TIM1[7:0] (Parameter 1 byte)
   - MDCDISPPRM21.TIM2[7:0] (Parameter 2 byte)
   - . .
   - . .
   - MDCDISPPRMxx.TIMn[7:0] (Parameter n byte)
2. Set the MDCDISPCTL.UPDFUNC bit to 1. (Select command write function)
3. Set the following bits when using the interrupt:
   - Write 1 to the MDCINTCTL.UPDIF bit. (Clear interrupt flag)
   - Set the MDCINTCTL.UPDIE bit to 1. (Enable MDC interrupt)
4. Configure the following MDCTRIGCTL register bits:
   - Set the MDCTRIGCTL.NPARAM[3:0] bits (Set number of parameter bytes to send)
   - Set MDCTRIGCTL.UPDTRIG bit to 1 (Start command write sequence)

When the command write is finished, the MDCTRIGCTL.UPDTRIG bit is automatically cleared to 0 and the MDCINTCTL.UPDIF bit is set to 1.

Triggering a display update

For the display update function (MDCDISPCTL.UPDFUNC=0), when triggered, the display updater will send a command byte (MDCDISPCLKDIV.TIM0[7:0] register, typically a "write RAM" command), followed by a dummy parameter byte (MDCDISPPRM21.TIM1[7:0]), and then pixel data.

To update the display, follow the procedure shown below.

1. Set the "write RAM" command and dummy parameter bytes to be sent to the panel to the following registers/bits:
   - MDCDISPCLKDIV.TIM0[7:0] bits (Set "write RAM" command byte)
   - MDCDISPPRM21.TIM1[7:0] bits (Set dummy parameter byte)
2. Set the MDCDISPCTL.UPDFUNC bit to 0. (Select display update function)
3. Set the following bits when using the interrupt:
   - Write 1 to the MDCINTCTL.UPDIF bit. (Clear interrupt flag)
   - Set the MDCINTCTL.UPDIE bit to 1. (Enable MDC interrupt)
4. Write 1 to the MDCTRIGCTL.UPDTRIG bit. (Start display update)

When the update is finished, the MDCTRIGCTL.UPDTRIG bit is automatically cleared to 0 and the MDCINTCTL.UPDIF bit is set to 1.

The sequence of pixel data sent to the panel depends on the bits-per-pixel (BPP) setting and width (in pixels) of the panel. The following tables show the pixel data sequence sent for each BPP setting for a panel width of W pixels and panel height of H pixels: (P[0,0] is top left pixel, P[(W-1), (H-1)] is bottom right pixel)

Table 21.19 1BPP Pixel Data Sequence

Byte Sequence	7	6	5	4	3	2	1	0
1	Command Byte[7:0]							
2	Dummy Parameter Byte [7:0]							
3	P[0,7]	P[0.6]	P[0,5]	P[0,4]	P[0,3]	P[0,2]	P[0,1]	P[0,0]
4	P[1,7]	P[1,6]	P[1,5]	P[1,4]	P[1,3]	P[1,2]	P[1,1]	P[1,0]
.	.	.	.	.	.	.	.	.
9	P[(W-2),7]	P[(W-2),6]	P[(W-2),5]	P[(W-2),4]	P[(W-2),3]	P[(W-2),2]	P[(W-2),1]	P[(W-2),0]
W + 2	P[(W-1),7]	P[(W-1),6]	P[(W-1),5]	P[(W-1),4]	P[(W-1),3]	P[(W-1),2]	P[(W-1),1]	P[(W-1),0]
W + 3	P[0,15]	P[0,14]	P[0,13]	P[0,12]	P[0,11]	P[0,10]	P[0,9]	P[0,8]
W + 4	P[1,15]	P[1,14]	P[1,13]	P[1,12]	P[1,11]	P[1,10]	P[1,9]	P[1,8]
.	.	.	.	.	.	.	.	.
(2xW) + 2	P[(W-1),15]	P[(W-1),14]	P[(W-1),13]	P[(W-1),12]	P[(W-1),11]	P[(W-1),10]	P[(W-1),9]	P[(W-1),8]
(2xW) + 3	P[0,23]	P[0,22]	P[0,21]	P[0,20]	P[0,19]	P[0,18]	P[0,17]	P[0,16]
(2xW) + 4	P[1,23]	P[1,22]	P[1,21]	P[1,20]	P[1,19]	P[1,18]	P[1,17]	P[1,16]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
((H/8)-1)xW + 2	P[(W-1),(H-9)]	P[(W-1),(H-10)]	P[(W-1),(H-11)]	P[(W-1),(H-12)]	P[(W-1),(H-13)]	P[(W-1),(H-14)]	P[(W-1),(H-15)]	P[(W-1),(H-16)]
((H/8)-1)xW + 3	P[0,(H-1)]	P[0,(H-2)]	P[0,(H-3)]	P[0,(H-4)]	P[0,(H-5)]	P[0,(H-6)]	P[0,(H-7)]	P[0,(H-8)]
.	.	.	.	.	.	.	.	.
(H/8)xW + 2	P[(W-1),(H-1)]	P[(W-1),(H-2)]	P[(W-1),(H-3)]	P[(W-1),(H-4)]	P[(W-1),(H-5)]	P[(W-1),(H-6)]	P[(W-1),(H-7)]	P[(W-1),(H-8)]

Table 21.20 2BPP Pixel Data Sequence

Byte Sequence	7	6	5	4	3	2	1	0
1	Command Byte[7:0]							
2	Dummy Parameter Byte [7:0]							
3	P[0.3]		P[0,2]		P[0,1]		P[0,0]	
4	P[1,3]		P[1,2]		P[1,1]		P[1,0]	
.	.		.		.		.	
9	P[(W-2),3]		P[(W-2),2]		P[(W-2),1]		P[(W-2),0]	
W + 2	P[(W-1),3]		P[(W-1),2]		P[(W-1),1]		P[(W-1),0]	
W + 3	P[0,7]		P[0,6]		P[0,5]		P[0,4]	
W + 4	P[1,7]		P[1,6]		P[1,5]		P[1,4]	
.	.		.		.		.	
(2xW) + 2	P[(W-1),7]		P[(W-1),6]		P[(W-1),5]		P[(W-1),4]	
(2xW) + 3	P[0,11]		P[0,10]		P[0,9]		P[0,8]	
(2xW) + 4	P[1,11]		P[1,10]		P[1,9]		P[1,8]	
.	.		.		.		.	
.	.		.		.		.	
((H/4)-1)xW + 2	P[(W-1),(H-5)]		P[(W-1),(H-6)]		P[(W-1),(H-7)]		P[(W-1),(H-8)]	
((H/4)-1)xW + 3	P[0,(H-1)]		P[0,(H-2)]		P[0,(H-3)]		P[0,(H-4)]	
.	.		.		.		.	
(H/4)xW + 2	P[(W-1),(H-1)]		P[(W-1),(H-2)]		P[(W-1),(H-3)]		P[(W-1),(H-4)]	

Table 21.21 4BPP Pixel Data Sequence

Byte Sequence	7	6	5	4	3	2	1	0
1	Command Byte[7:0]							
2	Dummy Parameter Byte [7:0]							
3	P[0,1]				P[0,0]			
4	P[1,1]				P[1,0]			
.	.				.			
9	P[(W-2),1]				P[(W-2),0]			
W + 2	P[(W-1),1]				P[(W-1),0]			
W + 3	P[0,3]				P[0,2]			
W + 4	P[1,3]				P[1,2]			
.	.				.			
(2xW) + 2	P[(W-1),3]				P[(W-1),2]			
(2xW) + 3	P[0,5]				P[0,4]			
(2xW) + 4	P[1,5]				P[1,4]			
.	.				.			
.	.				.			
((H/2)-1)xW + 2	P[(W-1),(H-3)]				P[(W-1),(H-4)]			
((H/2)-1)xW + 3	P[0,(H-1)]				P[0,(H-2)]			
.	.				.			
(H/2)xW + 2	P[(W-1),(H-1)]				P[(W-1),(H-2)]			

Table 21.22 8BPP Pixel Data Sequence

Byte Sequence	7	6	5	4	3	2	1	0
1	Command Byte[7:0]							
2	Dummy Parameter Byte [7:0]							
3	P[0,0]							
4	P[1,0]							
.	.							
9	P[(W-2),0]							
W + 2	P[(W-1),0]							
W + 3	P[0,1]							
W + 4	P[1,1]							
.	.							
(2xW) + 2	P[(W-1),1]							
(2xW) + 3	P[0,2]							
(2xW) + 4	P[1,2]							
.	.							
.	.							
(H-1)xW + 2	P[(W-1),(H-2)]							
(H-1)xW + 3	P[0,(H-1)]							
.	.							
HxW + 2	P[(W-1),(H-1)]							

21.6.2.4 EPD Panel Updater

Section 9.3.4 shows the timing diagram and parameters for the EPD panel interfaces. As shown in Section 9.3.4, the EPD updater provides flexibility in programming the panel interface timing by setting the MDCDISP*.TIM* and MDCDISPCTL2 registers.

There are 2 types of EPD panel interfaces: 3-wire Serial and 4-wire Serial. Both interfaces have the same command sequence for writing to and reading from the panel. A write or read command sequence starts with a command byte, indicated by D/C = 0, followed by data bytes, indicated by D/C = 1. For the 4-wire serial interface, there is a dedicated D/C pin. For the 3-wire serial interface, each byte of command or data sent has 9 clocks with the D/C value sent as the first bit followed by the 8-bit command or data.

NOTE: Due to a limitation in the chip, for the 4-wire serial interface, D/C does not go high during the data portion of a read command sequence.

The number of data bytes sent depends on the pixel format (1/2/4/8/BPP, selected by GSBPP=MDCTRIGCTRL.GSBPP[1:0]), display width (W=MDCDISPWIDTH[9:0] register), and display height (H=MDCDISPHEIGHT[9:0] register) as follows:

$$\text{Number of data bytes} = (W \times H) / (2^{(3-GSBPP)})$$

The command byte value to send is in the MDCDISPCLKDIV.TIM0[7:0] register and the MSB is sent first. The data bytes to send (write command sequence) or receive (read command sequence) are located in internal memory address location specified by the MDCDISPFRMBUFF[31:0] register. The MDCDISPCTL.UPDFUNC bit determines whether a display update is a write command sequence (MDCDISPCTL.UPDFUNC=1) or a read command sequence (MDCDISPCTL.UPDFUNC=0).

For 3-wire serial interface read command sequence, the MDCDISPCTL.DCRDEN bit determines whether or not the D/C bit is driven to 1 during the data bytes. If MDCDISPCTL.DCRDEN=0, the D/C bit is not driven during the data bytes. If MDCDISPCTL.DCRDEN=1, the first bit of the SDA pin (D/C) is driven to 1 and the SDA pin is tri-stated for the rest of the 8 data bits.

Figure 21.40 and Figure 21.41 show the byte sequence and the timing diagrams of the EPD panel interface types, respectively.

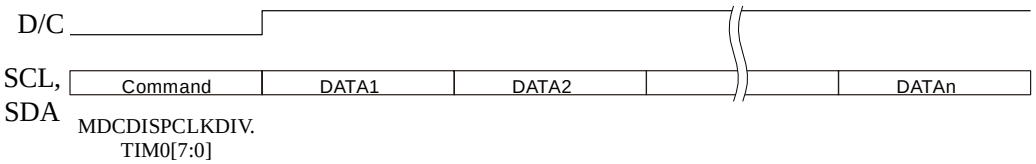


Figure 21.40 Byte Sequence for EPD Panel

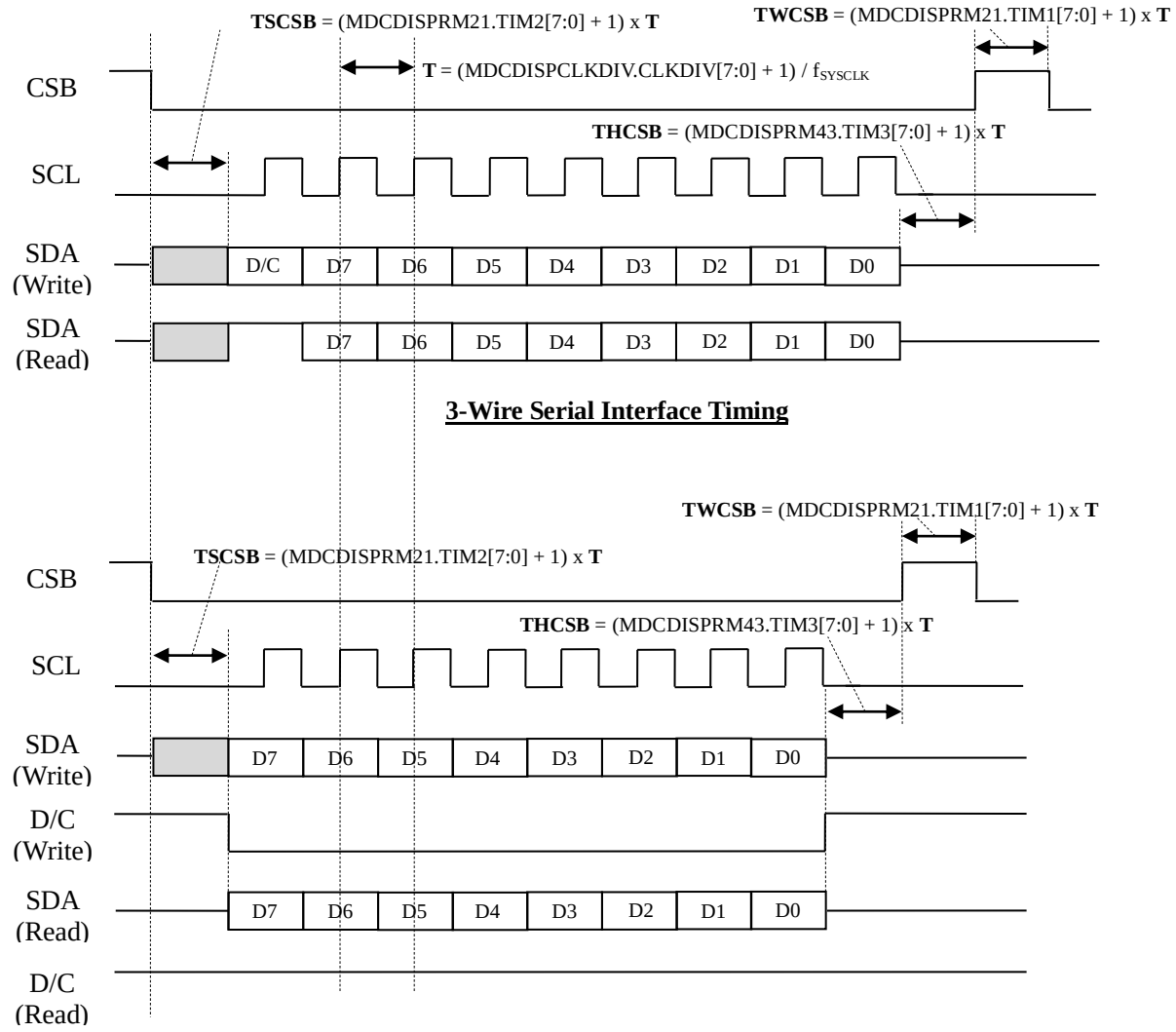


Figure 21.41 EPD Timing Diagrams

EPD panel updater initialization procedure

To set up the EPD panel, the following register initialization sequence is needed:

1. Configure the following MDCDISPCTL register bits:
   - Set MDCDISPCTL.DISPEPD bit to 1. (Select EPD panel interface)
   - Set MDCDISPCTL.ROTSEL[1:0] bits value. (Select swivel orientation)
   - Set MDCDISPCTL.CSPULSE bit value. (Select CSB pulse HIGH between data bytes)
   - Set MDCDISPCTL.DCRDEN bit value. (Select D/C bit assertion on read)
   - Set MDCDISPCTL.ADDRLSB bit value. (Select pixel bit order)
   - Set MDCDISPCTL.RGBORD bit value. (Select pixel order)
   - Set MDCDISPCTL.GSIF[1:0] bits value. (Select EPD panel interface type: 3-wire Serial or 4-wire Serial)
   - Set the MDCDISPCTL.VCOMEN bit to 0. (VCOM output is not used)
   - MDCDISPCTL.DISPINVERT bit (Enable/disable pixel inversion)
2. Configure the following MDCDISPCLKDIV register bits:
   - MDCDISPCLKDIV.CLKDIV[7:0] bits (Set panel timing unit T)

T = [S x (CLKDIV[7:0] + 1)]
where S is the system clock period

3. Configure the following MDCDISPPRM21 register bits:
   - MDCDISPPRM21.TIM1[7:0] bits (Set TWCSB timing value)
   - MDCDISPPRM21.TIM2[7:0] bits (Set TSCSB timing value)
4. Configure the following MDCDISPPRM43 register bits:
   - MDCDISPPRM43.TIM3[7:0] bits (Set THCSB timing value)
5. Set the MDCDISPSTRIDE.DISPSTRIDE[9:0] bits. (Set frame buffer stride)
   The frame buffer image can be wider than the display width to allow for panning.
6. Configure the following MDCDISPCTL2 register bits:
   - Set MDCDISPCTL2.CSPOL bit to 0. (Configure active-low chip-select)
   - MDCDISPCTL2.GSALPHA bit (Disable/enable alpha channel for the selected BPP format)

Writing bytes to panel registers

The EPD panel has commands for configuring internal registers. The write command sequence is a command byte followed by one or more parameter bytes. The number of parameter bytes depends on the command.

To trigger a write command sequence, follow the procedure shown below.

1. Write the command byte value to the MDCDISPCLKDIV.TIM0[7:0] register.
2. Set the MDCDISPCTL.UPDFUNC bit to 1. (Select write command sequence)
3. Set MDCDISPCTL.ADDRLSB bit to 1. (Select MSB sent first. Parameter bytes of most panels are MSB first.)
4. Set the MDCTRIGCTRL.GSBPP[1:0] bits to 11b. (Select 8BPP, one byte per pixel)
5. Set the MDCDISPWIDTH.DISPWIDTH[9:0] bits value. (Set number of parameter bytes to send)
6. Set the MDCDISPHEIGHT.DISPHEIGHT[9:0] bits to 1. (Set to 1 so that number of parameter bytes is solely determined by width)
7. Set the MDCDISPFRMBUFF0.FRMBUFFADDR[15:0] and MDCDISPFRMBUFF1.FRMBUFFADDR[31:16] bits to point to memory address where the parameter byte values to send are located. (Set parameters buffer base address)
8. Set the following bits when using the interrupt:
   - Write 1 to the MDCINTCTL.UPDIF bit. (Clear interrupt flag)
   - Set the MDCINTCTL.UPDIE bit to 1. (Enable MDC interrupt)
9. Write 1 to the MDCTRIGCTL.UPDTRIG bit. (Start display update)

When the write command sequence is finished, the MDCTRIGCTL.UPDTRIG bit is automatically cleared to 0 and the MDCINTCTL.UPDIF bit is set to 1.

Reading bytes from panel registers

Some EPD panels have commands to read bytes from their internal registers or memory. The read command sequence is a command byte sent to the panel followed by one or more data bytes received from the panel. The number of data bytes depends on the command.

To trigger a read command sequence, follow the procedure shown below.

1. Write the command byte value to the MDCDISPCLKDIV.TIM0[7:0] register.
2. Set the MDCDISPCTL.UPDFUNC bit to 0. (Select read command sequence)
3. Set MDCDISPCTL.ADDRLSB bit to 1. (Select MSB sent first. Data bytes of most panels are MSB first.)
4. Set the MDCTRIGCTRL.GSBPP[1:0] bits to 11b. (Select 8BPP, one byte per pixel)
5. Set the MDCDISPWIDTH.DISPWIDTH[9:0] bits value. (Set number of data bytes to receive)
6. Set the MDCDISPHEIGHT.DISPHEIGHT[9:0] bits to 1. (Set to 1 so that number of data bytes is solely determined by width)

7. Set the MDCDISPFRMBUFF0.FRMBUFFADDR[15:0] and MDCDISPFRMBUFF1.FRMBUFFADDR[31:16] bits to point to memory address where the received data byte values are to be stored. (Set data buffer base address)
8. Set the following bits when using the interrupt:
   - Write 1 to the MDCINTCTL.UPDIF bit. (Clear interrupt flag)
   - Set the MDCINTCTL.UPDIE bit to 1. (Enable MDC interrupt)
9. Write 1 to the MDCTRIGCTL.UPDTRIG bit. (Start display update)

When the read command sequence is finished, the MDCTRIGCTL.UPDTRIG bit is automatically cleared to 0 and the MDCINTCTL.UPDIF bit is set to 1. The received bytes can then be read from the data buffer.

Triggering a display update

To update the display and send pixels to the panel from the frame buffer, follow the procedure shown below.

1. Write the "display update" command byte value to the MDCDISPCLKDIV.TIM0[7:0] register.
2. Set the MDCDISPCTL.UPDFUNC bit to 1. (Select write command sequence)
3. Set the MDCDISPCTL.ADDRLSB bit value. (Select bit order to send for each pixel. MSB or LSB sent first.)
4. Set the MDCDISPCTL.RGBORD bit value. (Select pixel order to send within each byte. Most significant pixel or least significant pixel sent first.)
5. Set the MDCTRIGCTRL.GSBPP[1:0] bits value. (Select bits-per-pixel format, 1/2/4/8)
6. Set the MDCDISPWIDTH.DISPWIDTH[9:0] bits value. (Set display width in pixels)
7. Set the MDCDISPHEIGHT.DISPHEIGHT[9:0] bits value. (Set display height in pixels)
8. Set the MDCDISPFRMBUFF0.FRMBUFFADDR[15:0] and MDCDISPFRMBUFF1.FRMBUFFADDR[31:16] bits to point to memory address where the pixel data to send is located. (Set frame buffer base address)
9. Set the following bits when using the interrupt:
   - Write 1 to the MDCINTCTL.UPDIF bit. (Clear interrupt flag)
   - Set the MDCINTCTL.UPDIE bit to 1. (Enable MDC interrupt)
10. Write 1 to the MDCTRIGCTL.UPDTRIG bit. (Start display update)

When the display update is finished, the MDCTRIGCTL.UPDTRIG bit is automatically cleared to 0 and the MDCINTCTL.UPDIF bit is set to 1.

Reading pixel data from panel

Some EPD panels may have commands to read pixel data from the panel. The procedure to trigger pixel data read from the panel is the same as the procedure for display update except step 2 (MDCDISPCTL.UPDFUNC bit is set to 0 for read command sequence) and step 8 (frame buffer address should be a RAM location where the pixel data received is written).

Pixel data bit order and pixel order

The sequence of pixel data bytes sent or received between the S1D13C00 and the panel depends on the pixel format (1/2/4/8 BPP), bit order setting (MDCDISPCTL.ADDRLSB bit), pixel order setting (MDCDISPCTL.RGBORD bit), display width, and display height. The following tables show the pixel data sequence for bit order setting B, pixel order setting P, panel width of W pixels (W is multiple of 8/BPP), and panel height of H pixels: (D[<byte>, <bit>], <byte> is the byte number in frame buffer, <bit> is bit number within the byte, LSP means least significant pixel, MSP means most significant pixel)

Table 21.23 1BPP Pixel Data Sequence, P=0 (LSP first, B is don't care)

Byte Sequence	Serial Data Bits (SD7 sent first)							
	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
1	Command Byte[7:0]							
2	D[0,0]	D[0,1]	D[0,2]	D[0,3]	D[0,4]	D[0,5]	D[0,6]	D[0,7]
3	D[1,0]	D[1,1]	D[1,2]	D[1,3]	D[1,4]	D[1,5]	D[1,6]	D[1,7]
.	.	.	.	.	.	.	.	.
(W/8) + 1	D[(W/8)-1,0]	D[(W/8)-1,1]	D[(W/8)-1,2]	D[(W/8)-1,3]	D[(W/8)-1,4]	D[(W/8)-1,5]	D[(W/8)-1,6]	D[(W/8)-1,7]
(W/8) + 2	D[(W/8),0]	D[(W/8),1]	D[(W/8),2]	D[(W/8),3]	D[(W/8),4]	D[(W/8),5]	D[(W/8),6]	D[(W/8),7]
(W/8) + 3	D[(W/8)+1,0]	D[(W/8)+1,1]	D[(W/8)+1,2]	D[(W/8)+1,3]	D[(W/8)+1,4]	D[(W/8)+1,5]	D[(W/8)+1,6]	D[(W/8)+1,7]
.	.	.	.	.	.	.	.	.
2x(W/8) + 1	D[2x(W/8)-1,0]	D[2x(W/8)-1,1]	D[2x(W/8)-1,2]	D[2x(W/8)-1,3]	D[2x(W/8)-1,4]	D[2x(W/8)-1,5]	D[2x(W/8)-1,6]	D[2x(W/8)-1,7]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
(H-1)x(W/8) + 1	D[(H-1)x(W/8)-1,0]	D[(H-1)x(W/8)-1,1]	D[(H-1)x(W/8)-1,2]	D[(H-1)x(W/8)-1,3]	D[(H-1)x(W/8)-1,4]	D[(H-1)x(W/8)-1,5]	D[(H-1)x(W/8)-1,6]	D[(H-1)x(W/8)-1,7]
(H-1)x(W/8) + 2	D[(H-1)x(W/8),0]	D[(H-1)x(W/8),1]	D[(H-1)x(W/8),2]	D[(H-1)x(W/8),3]	D[(H-1)x(W/8),4]	D[(H-1)x(W/8),5]	D[(H-1)x(W/8),6]	D[(H-1)x(W/8),7]
.	.	.	.	.	.	.	.	.
Hx(W/8) + 1	D[Hx(W/8)-1,0]	D[Hx(W/8)-1,1]	D[Hx(W/8)-1,2]	D[Hx(W/8)-1,3]	D[Hx(W/8)-1,4]	D[Hx(W/8)-1,5]	D[Hx(W/8)-1,6]	D[Hx(W/8)-1,7]

Table 21.24 1BPP Pixel Data Sequence, P=1 (MSP first, B is don't care)

Byte Sequence	Serial Data Bits (SD7 sent first)							
	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
1	Command Byte[7:0]							
2	D[0,7]	D[0,6]	D[0,5]	D[0,4]	D[0,3]	D[0,2]	D[0,1]	D[0,0]
3	D[1,7]	D[1,6]	D[1,5]	D[1,4]	D[1,3]	D[1,2]	D[1,1]	D[1,0]
.	.	.	.	.	.	.	.	.
(W/8) + 1	D[(W/8)-1,7]	D[(W/8)-1,6]	D[(W/8)-1,5]	D[(W/8)-1,4]	D[(W/8)-1,3]	D[(W/8)-1,2]	D[(W/8)-1,1]	D[(W/8)-1,0]
(W/8) + 2	D[(W/8),7]	D[(W/8),6]	D[(W/8),5]	D[(W/8),4]	D[(W/8),3]	D[(W/8),2]	D[(W/8),1]	D[(W/8),0]
(W/8) + 3	D[(W/8)+1,7]	D[(W/8)+1,6]	D[(W/8)+1,5]	D[(W/8)+1,4]	D[(W/8)+1,3]	D[(W/8)+1,2]	D[(W/8)+1,1]	D[(W/8)+1,0]
.	.	.	.	.	.	.	.	.
2x(W/8) + 1	D[2x(W/8)-1,7]	D[2x(W/8)-1,6]	D[2x(W/8)-1,5]	D[2x(W/8)-1,4]	D[2x(W/8)-1,3]	D[2x(W/8)-1,2]	D[2x(W/8)-1,1]	D[2x(W/8)-1,0]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
(H-1)x(W/8) + 1	D[(H-1)x(W/8)-1,7]	D[(H-1)x(W/8)-1,6]	D[(H-1)x(W/8)-1,5]	D[(H-1)x(W/8)-1,4]	D[(H-1)x(W/8)-1,3]	D[(H-1)x(W/8)-1,2]	D[(H-1)x(W/8)-1,1]	D[(H-1)x(W/8)-1,0]
(H-1)x(W/8) + 2	D[(H-1)x(W/8),7]	D[(H-1)x(W/8),6]	D[(H-1)x(W/8),5]	D[(H-1)x(W/8),4]	D[(H-1)x(W/8),3]	D[(H-1)x(W/8),2]	D[(H-1)x(W/8),1]	D[(H-1)x(W/8),0]
.	.	.	.	.	.	.	.	.
Hx(W/8) + 1	D[Hx(W/8)-1,7]	D[Hx(W/8)-1,6]	D[Hx(W/8)-1,5]	D[Hx(W/8)-1,4]	D[Hx(W/8)-1,3]	D[Hx(W/8)-1,2]	D[Hx(W/8)-1,1]	D[Hx(W/8)-1,0]

Table 21.25 2BPP Pixel Data Sequence, B=0 (LSB first), P=0 (LSP first)

Byte Sequence	Serial Data Bits (SD7 sent first)							
	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
1	Command Byte[7:0]							
2	D[0,0]	D[0,1]	D[0,2]	D[0,3]	D[0,4]	D[0,5]	D[0,6]	D[0,7]
3	D[1,0]	D[1,1]	D[1,2]	D[1,3]	D[1,4]	D[1,5]	D[1,6]	D[1,7]
.	.	.	.	.	.	.	.	.
(W/4) + 1	D[(W/4)-1,0]	D[(W/4)-1,1]	D[(W/4)-1,2]	D[(W/4)-1,3]	D[(W/4)-1,4]	D[(W/4)-1,5]	D[(W/4)-1,6]	D[(W/4)-1,7]
(W/4) + 2	D[(W/4),0]	D[(W/4),1]	D[(W/4),2]	D[(W/4),3]	D[(W/4),4]	D[(W/4),5]	D[(W/4),6]	D[(W/4),7]
(W/4) + 3	D[(W/4)+1,0]	D[(W/4)+1,1]	D[(W/4)+1,2]	D[(W/4)+1,3]	D[(W/4)+1,4]	D[(W/4)+1,5]	D[(W/4)+1,6]	D[(W/4)+1,7]
.	.	.	.	.	.	.	.	.
2x(W/4) + 1	D[2x(W/4)-1,0]	D[2x(W/4)-1,1]	D[2x(W/4)-1,2]	D[2x(W/4)-1,3]	D[2x(W/4)-1,4]	D[2x(W/4)-1,5]	D[2x(W/4)-1,6]	D[2x(W/4)-1,7]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
(H-1)x(W/4) + 1	D[(H-1)x(W/4)-1,0]	D[(H-1)x(W/4)-1,1]	D[(H-1)x(W/4)-1,2]	D[(H-1)x(W/4)-1,3]	D[(H-1)x(W/4)-1,4]	D[(H-1)x(W/4)-1,5]	D[(H-1)x(W/4)-1,6]	D[(H-1)x(W/4)-1,7]
(H-1)x(W/4) + 2	D[(H-1)x(W/4),0]	D[(H-1)x(W/4),1]	D[(H-1)x(W/4),2]	D[(H-1)x(W/4),3]	D[(H-1)x(W/4),4]	D[(H-1)x(W/4),5]	D[(H-1)x(W/4),6]	D[(H-1)x(W/4),7]
.	.	.	.	.	.	.	.	.
Hx(W/4) + 1	D[Hx(W/4)-1,0]	D[Hx(W/4)-1,1]	D[Hx(W/4)-1,2]	D[Hx(W/4)-1,3]	D[Hx(W/4)-1,4]	D[Hx(W/4)-1,5]	D[Hx(W/4)-1,6]	D[Hx(W/4)-1,7]

Table 21.26 2BPP Pixel Data Sequence, B=1 (MSB first), P=0 (LSP first)

Byte Sequence	Serial Data Bits (SD7 sent first)							
	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
1	Command Byte[7:0]							
2	D[0,1]	D[0,0]	D[0,3]	D[0,2]	D[0,5]	D[0,4]	D[0,7]	D[0,6]
3	D[1,1]	D[1,0]	D[1,3]	D[1,2]	D[1,5]	D[1,4]	D[1,7]	D[1,6]
.	.	.	.	.	.	.	.	.
(W/4) + 1	D[(W/4)-1,1]	D[(W/4)-1,0]	D[(W/4)-1,3]	D[(W/4)-1,2]	D[(W/4)-1,5]	D[(W/4)-1,4]	D[(W/4)-1,7]	D[(W/4)-1,6]
(W/4) + 2	D[(W/4),1]	D[(W/4),0]	D[(W/4),3]	D[(W/4),2]	D[(W/4),5]	D[(W/4),4]	D[(W/4),7]	D[(W/4),6]
(W/4) + 3	D[(W/4)+1,1]	D[(W/4)+1,0]	D[(W/4)+1,3]	D[(W/4)+1,2]	D[(W/4)+1,5]	D[(W/4)+1,4]	D[(W/4)+1,7]	D[(W/4)+1,6]
.	.	.	.	.	.	.	.	.
2x(W/4) + 1	D[2x(W/4)-1,1]	D[2x(W/4)-1,0]	D[2x(W/4)-1,3]	D[2x(W/4)-1,2]	D[2x(W/4)-1,5]	D[2x(W/4)-1,4]	D[2x(W/4)-1,7]	D[2x(W/4)-1,6]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
(H-1)x(W/4) + 1	D[(H-1)x(W/4)-1,1]	D[(H-1)x(W/4)-1,0]	D[(H-1)x(W/4)-1,3]	D[(H-1)x(W/4)-1,2]	D[(H-1)x(W/4)-1,5]	D[(H-1)x(W/4)-1,4]	D[(H-1)x(W/4)-1,7]	D[(H-1)x(W/4)-1,6]
(H-1)x(W/4) + 2	D[(H-1)x(W/4),1]	D[(H-1)x(W/4),0]	D[(H-1)x(W/4),3]	D[(H-1)x(W/4),2]	D[(H-1)x(W/4),5]	D[(H-1)x(W/4),4]	D[(H-1)x(W/4),7]	D[(H-1)x(W/4),6]
.	.	.	.	.	.	.	.	.
Hx(W/4) + 1	D[Hx(W/4)-1,1]	D[Hx(W/4)-1,0]	D[Hx(W/4)-1,3]	D[Hx(W/4)-1,2]	D[Hx(W/4)-1,5]	D[Hx(W/4)-1,4]	D[Hx(W/4)-1,7]	D[Hx(W/4)-1,6]

Table 21.27 2BPP Pixel Data Sequence, B=0 (LSB first), P=1 (MSP first)

Byte Sequence	Serial Data Bits (SD7 sent first)							
	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
1	Command Byte[7:0]							
2	D[0,6]	D[0,7]	D[0,4]	D[0,5]	D[0,2]	D[0,3]	D[0,0]	D[0,1]
3	D[1,6]	D[1,7]	D[1,4]	D[1,5]	D[1,2]	D[1,3]	D[1,0]	D[1,1]
.	.	.	.	.	.	.	.	.
(W/4) + 1	D[(W/4)-1,6]	D[(W/4)-1,7]	D[(W/4)-1,4]	D[(W/4)-1,5]	D[(W/4)-1,2]	D[(W/4)-1,3]	D[(W/4)-1,0]	D[(W/4)-1,1]
(W/4) + 2	D[(W/4),6]	D[(W/4),7]	D[(W/4),4]	D[(W/4),5]	D[(W/4),2]	D[(W/4),3]	D[(W/4),0]	D[(W/4),1]
(W/4) + 3	D[(W/4)+1,6]	D[(W/4)+1,7]	D[(W/4)+1,4]	D[(W/4)+1,5]	D[(W/4)+1,2]	D[(W/4)+1,3]	D[(W/4)+1,0]	D[(W/4)+1,1]
.	.	.	.	.	.	.	.	.
2x(W/4) + 1	D[2x(W/4)-1,6]	D[2x(W/4)-1,7]	D[2x(W/4)-1,4]	D[2x(W/4)-1,5]	D[2x(W/4)-1,2]	D[2x(W/4)-1,3]	D[2x(W/4)-1,0]	D[2x(W/4)-1,1]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
(H-1)x(W/4) + 1	D[(H-1)x(W/4)-1,6]	D[(H-1)x(W/4)-1,7]	D[(H-1)x(W/4)-1,4]	D[(H-1)x(W/4)-1,5]	D[(H-1)x(W/4)-1,2]	D[(H-1)x(W/4)-1,3]	D[(H-1)x(W/4)-1,0]	D[(H-1)x(W/4)-1,1]
(H-1)x(W/4) + 2	D[(H-1)x(W/4),6]	D[(H-1)x(W/4),7]	D[(H-1)x(W/4),4]	D[(H-1)x(W/4),5]	D[(H-1)x(W/4),2]	D[(H-1)x(W/4),3]	D[(H-1)x(W/4),0]	D[(H-1)x(W/4),1]
.	.	.	.	.	.	.	.	.
Hx(W/4) + 1	D[Hx(W/4)-1,6]	D[Hx(W/4)-1,7]	D[Hx(W/4)-1,4]	D[Hx(W/4)-1,5]	D[Hx(W/4)-1,2]	D[Hx(W/4)-1,3]	D[Hx(W/4)-1,0]	D[Hx(W/4)-1,1]

Table 21.28 2BPP Pixel Data Sequence, B=1 (MSB first), P=1 (MSP first)

Byte Sequence	Serial Data Bits (SD7 sent first)							
	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
1	Command Byte[7:0]							
2	D[0,7]	D[0,6]	D[0,5]	D[0,4]	D[0,3]	D[0,2]	D[0,1]	D[0,0]
3	D[1,7]	D[1,6]	D[1,5]	D[1,4]	D[1,3]	D[1,2]	D[1,1]	D[1,0]
.	.	.	.	.	.	.	.	.
(W/4) + 1	D[(W/4)-1,7]	D[(W/4)-1,6]	D[(W/4)-1,5]	D[(W/4)-1,4]	D[(W/4)-1,3]	D[(W/4)-1,2]	D[(W/4)-1,1]	D[(W/4)-1,0]
(W/4) + 2	D[(W/4),7]	D[(W/4),6]	D[(W/4),5]	D[(W/4),4]	D[(W/4),3]	D[(W/4),2]	D[(W/4),1]	D[(W/4),0]
(W/4) + 3	D[(W/4)+1,7]	D[(W/4)+1,6]	D[(W/4)+1,5]	D[(W/4)+1,4]	D[(W/4)+1,3]	D[(W/4)+1,2]	D[(W/4)+1,1]	D[(W/4)+1,0]
.	.	.	.	.	.	.	.	.
2x(W/4) + 1	D[2x(W/4)-1,7]	D[2x(W/4)-1,6]	D[2x(W/4)-1,5]	D[2x(W/4)-1,4]	D[2x(W/4)-1,3]	D[2x(W/4)-1,2]	D[2x(W/4)-1,1]	D[2x(W/4)-1,0]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
(H-1)x(W/4) + 1	D[(H-1)x(W/4)-1,7]	D[(H-1)x(W/4)-1,6]	D[(H-1)x(W/4)-1,5]	D[(H-1)x(W/4)-1,4]	D[(H-1)x(W/4)-1,3]	D[(H-1)x(W/4)-1,2]	D[(H-1)x(W/4)-1,1]	D[(H-1)x(W/4)-1,0]
(H-1)x(W/4) + 2	D[(H-1)x(W/4),7]	D[(H-1)x(W/4),6]	D[(H-1)x(W/4),5]	D[(H-1)x(W/4),4]	D[(H-1)x(W/4),3]	D[(H-1)x(W/4),2]	D[(H-1)x(W/4),1]	D[(H-1)x(W/4),0]
.	.	.	.	.	.	.	.	.
Hx(W/4) + 1	D[Hx(W/4)-1,7]	D[Hx(W/4)-1,6]	D[Hx(W/4)-1,5]	D[Hx(W/4)-1,4]	D[Hx(W/4)-1,3]	D[Hx(W/4)-1,2]	D[Hx(W/4)-1,1]	D[Hx(W/4)-1,0]

Table 21.29 4BPP Pixel Data Sequence, B=0 (LSB first), P=0 (LSP first)

Byte Sequence	Serial Data Bits (SD7 sent first)							
	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
1	Command Byte[7:0]							
2	D[0,0]	D[0,1]	D[0,2]	D[0,3]	D[0,4]	D[0,5]	D[0,6]	D[0,7]
3	D[1,0]	D[1,1]	D[1,2]	D[1,3]	D[1,4]	D[1,5]	D[1,6]	D[1,7]
.	.	.	.	.	.	.	.	.
(W/2) + 1	D[(W/2)-1,0]	D[(W/2)-1,1]	D[(W/2)-1,2]	D[(W/2)-1,3]	D[(W/2)-1,4]	D[(W/2)-1,5]	D[(W/2)-1,6]	D[(W/2)-1,7]
(W/2) + 2	D[(W/2),0]	D[(W/2),1]	D[(W/2),2]	D[(W/2),3]	D[(W/2),4]	D[(W/2),5]	D[(W/2),6]	D[(W/2),7]
(W/2) + 3	D[(W/2)+1,0]	D[(W/2)+1,1]	D[(W/2)+1,2]	D[(W/2)+1,3]	D[(W/2)+1,4]	D[(W/2)+1,5]	D[(W/2)+1,6]	D[(W/2)+1,7]
.	.	.	.	.	.	.	.	.
2x(W/2) + 1	D[2x(W/2)-1,0]	D[2x(W/2)-1,1]	D[2x(W/2)-1,2]	D[2x(W/2)-1,3]	D[2x(W/2)-1,4]	D[2x(W/2)-1,5]	D[2x(W/2)-1,6]	D[2x(W/2)-1,7]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
(H-1)x(W/2) + 1	D[(H-1)x(W/2)-1,0]	D[(H-1)x(W/2)-1,1]	D[(H-1)x(W/2)-1,2]	D[(H-1)x(W/2)-1,3]	D[(H-1)x(W/2)-1,4]	D[(H-1)x(W/2)-1,5]	D[(H-1)x(W/2)-1,6]	D[(H-1)x(W/2)-1,7]
(H-1)x(W/2) + 2	D[(H-1)x(W/2),0]	D[(H-1)x(W/2),1]	D[(H-1)x(W/2),2]	D[(H-1)x(W/2),3]	D[(H-1)x(W/2),4]	D[(H-1)x(W/2),5]	D[(H-1)x(W/2),6]	D[(H-1)x(W/2),7]
.	.	.	.	.	.	.	.	.
Hx(W/2) + 1	D[Hx(W/2)-1,0]	D[Hx(W/2)-1,1]	D[Hx(W/2)-1,2]	D[Hx(W/2)-1,3]	D[Hx(W/2)-1,4]	D[Hx(W/2)-1,5]	D[Hx(W/2)-1,6]	D[Hx(W/2)-1,7]

Table 21.30 4BPP Pixel Data Sequence, B=1 (MSB first), P=0 (LSP first)

Byte Sequence	Serial Data Bits (SD7 sent first)							
	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
1	Command Byte[7:0]							
2	D[0,3]	D[0,2]	D[0,1]	D[0,0]	D[0,7]	D[0,6]	D[0,5]	D[0,4]
3	D[1,3]	D[1,2]	D[1,1]	D[1,0]	D[1,7]	D[1,6]	D[1,5]	D[1,4]
.	.	.	.	.	.	.	.	.
(W/2) + 1	D[(W/2)-1,3]	D[(W/2)-1,2]	D[(W/2)-1,1]	D[(W/2)-1,0]	D[(W/2)-1,7]	D[(W/2)-1,6]	D[(W/2)-1,5]	D[(W/2)-1,4]
(W/2) + 2	D[(W/2),3]	D[(W/2),2]	D[(W/2),1]	D[(W/2),0]	D[(W/2),7]	D[(W/2),6]	D[(W/2),5]	D[(W/2),4]
(W/2) + 3	D[(W/2)+1,3]	D[(W/2)+1,2]	D[(W/2)+1,1]	D[(W/2)+1,0]	D[(W/2)+1,7]	D[(W/2)+1,6]	D[(W/2)+1,5]	D[(W/2)+1,4]
.	.	.	.	.	.	.	.	.
2x(W/2) + 1	D[2x(W/2)-1,3]	D[2x(W/2)-1,2]	D[2x(W/2)-1,1]	D[2x(W/2)-1,0]	D[2x(W/2)-1,7]	D[2x(W/2)-1,6]	D[2x(W/2)-1,5]	D[2x(W/2)-1,4]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
(H-1)x(W/2) + 1	D[(H-1)x(W/2)-1,3]	D[(H-1)x(W/2)-1,2]	D[(H-1)x(W/2)-1,1]	D[(H-1)x(W/2)-1,0]	D[(H-1)x(W/2)-1,7]	D[(H-1)x(W/2)-1,6]	D[(H-1)x(W/2)-1,5]	D[(H-1)x(W/2)-1,4]
(H-1)x(W/2) + 2	D[(H-1)x(W/2),3]	D[(H-1)x(W/2),2]	D[(H-1)x(W/2),1]	D[(H-1)x(W/2),0]	D[(H-1)x(W/2),7]	D[(H-1)x(W/2),6]	D[(H-1)x(W/2),5]	D[(H-1)x(W/2),4]
.	.	.	.	.	.	.	.	.
Hx(W/2) + 1	D[Hx(W/2)-1,3]	D[Hx(W/2)-1,2]	D[Hx(W/2)-1,1]	D[Hx(W/2)-1,0]	D[Hx(W/2)-1,7]	D[Hx(W/2)-1,6]	D[Hx(W/2)-1,5]	D[Hx(W/2)-1,4]

Table 21.31 4BPP Pixel Data Sequence, B=0 (LSB first), P=1 (MSP first)

Byte Sequence	Serial Data Bits (SD7 sent first)							
	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
1	Command Byte[7:0]							
2	D[0,4]	D[0,5]	D[0,6]	D[0,7]	D[0,0]	D[0,1]	D[0,2]	D[0,3]
3	D[1,4]	D[1,5]	D[1,6]	D[1,7]	D[1,0]	D[1,1]	D[1,2]	D[1,3]
.	.	.	.	.	.	.	.	.
(W/2) + 1	D[(W/2)-1,4]	D[(W/2)-1,5]	D[(W/2)-1,6]	D[(W/2)-1,7]	D[(W/2)-1,0]	D[(W/2)-1,1]	D[(W/2)-1,2]	D[(W/2)-1,3]
(W/2) + 2	D[(W/2),4]	D[(W/2),5]	D[(W/2),6]	D[(W/2),7]	D[(W/2),0]	D[(W/2),1]	D[(W/2),2]	D[(W/2),3]
(W/2) + 3	D[(W/2)+1,4]	D[(W/2)+1,5]	D[(W/2)+1,6]	D[(W/2)+1,7]	D[(W/2)+1,0]	D[(W/2)+1,1]	D[(W/2)+1,2]	D[(W/2)+1,3]
.	.	.	.	.	.	.	.	.
2x(W/2) + 1	D[2x(W/2)-1,4]	D[2x(W/2)-1,5]	D[2x(W/2)-1,6]	D[2x(W/2)-1,7]	D[2x(W/2)-1,0]	D[2x(W/2)-1,1]	D[2x(W/2)-1,2]	D[2x(W/2)-1,3]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
(H-1)x(W/2) + 1	D[(H-1)x(W/2)-1,4]	D[(H-1)x(W/2)-1,5]	D[(H-1)x(W/2)-1,6]	D[(H-1)x(W/2)-1,7]	D[(H-1)x(W/2)-1,0]	D[(H-1)x(W/2)-1,1]	D[(H-1)x(W/2)-1,2]	D[(H-1)x(W/2)-1,3]
(H-1)x(W/2) + 2	D[(H-1)x(W/2),4]	D[(H-1)x(W/2),5]	D[(H-1)x(W/2),6]	D[(H-1)x(W/2),7]	D[(H-1)x(W/2),0]	D[(H-1)x(W/2),1]	D[(H-1)x(W/2),2]	D[(H-1)x(W/2),3]
.	.	.	.	.	.	.	.	.
Hx(W/2) + 1	D[Hx(W/2)-1,4]	D[Hx(W/2)-1,5]	D[Hx(W/2)-1,6]	D[Hx(W/2)-1,7]	D[Hx(W/2)-1,0]	D[Hx(W/2)-1,1]	D[Hx(W/2)-1,2]	D[Hx(W/2)-1,3]

Table 21.32 4BPP Pixel Data Sequence, B=1 (MSB first), P=1 (MSP first)

Byte Sequence	Serial Data Bits (SD7 sent first)							
	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
1	Command Byte[7:0]							
2	D[0,7]	D[0,6]	D[0,5]	D[0,4]	D[0,3]	D[0,2]	D[0,1]	D[0,0]
3	D[1,7]	D[1,6]	D[1,5]	D[1,4]	D[1,3]	D[1,2]	D[1,1]	D[1,0]
.	.	.	.	.	.	.	.	.
(W/2) + 1	D[(W/2)-1,7]	D[(W/2)-1,6]	D[(W/2)-1,5]	D[(W/2)-1,4]	D[(W/2)-1,3]	D[(W/2)-1,2]	D[(W/2)-1,1]	D[(W/2)-1,0]
(W/2) + 2	D[(W/2),7]	D[(W/2),6]	D[(W/2),5]	D[(W/2),4]	D[(W/2),3]	D[(W/2),2]	D[(W/2),1]	D[(W/2),0]
(W/2) + 3	D[(W/2)+1,7]	D[(W/2)+1,6]	D[(W/2)+1,5]	D[(W/2)+1,4]	D[(W/2)+1,3]	D[(W/2)+1,2]	D[(W/2)+1,1]	D[(W/2)+1,0]
.	.	.	.	.	.	.	.	.
2x(W/2) + 1	D[2x(W/2)-1,7]	D[2x(W/2)-1,6]	D[2x(W/2)-1,5]	D[2x(W/2)-1,4]	D[2x(W/2)-1,3]	D[2x(W/2)-1,2]	D[2x(W/2)-1,1]	D[2x(W/2)-1,0]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
(H-1)x(W/2) + 1	D[(H-1)x(W/2)-1,7]	D[(H-1)x(W/2)-1,6]	D[(H-1)x(W/2)-1,5]	D[(H-1)x(W/2)-1,4]	D[(H-1)x(W/2)-1,3]	D[(H-1)x(W/2)-1,2]	D[(H-1)x(W/2)-1,1]	D[(H-1)x(W/2)-1,0]
(H-1)x(W/2) + 2	D[(H-1)x(W/2),7]	D[(H-1)x(W/2),6]	D[(H-1)x(W/2),5]	D[(H-1)x(W/2),4]	D[(H-1)x(W/2),3]	D[(H-1)x(W/2),2]	D[(H-1)x(W/2),1]	D[(H-1)x(W/2),0]
.	.	.	.	.	.	.	.	.
Hx(W/2) + 1	D[Hx(W/2)-1,7]	D[Hx(W/2)-1,6]	D[Hx(W/2)-1,5]	D[Hx(W/2)-1,4]	D[Hx(W/2)-1,3]	D[Hx(W/2)-1,2]	D[Hx(W/2)-1,1]	D[Hx(W/2)-1,0]

Table 21.33 8BPP Pixel Data Sequence, B=0 (LSB first, P is don't care)

Byte Sequence	Serial Data Bits (SD7 sent first)							
	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
1	Command Byte[7:0]							
2	D[0,0]	D[0,1]	D[0,2]	D[0,3]	D[0,4]	D[0,5]	D[0,6]	D[0,7]
3	D[1,0]	D[1,1]	D[1,2]	D[1,3]	D[1,4]	D[1,5]	D[1,6]	D[1,7]
.	.	.	.	.	.	.	.	.
W + 1	D[W-1,0]	D[W-1,1]	D[W-1,2]	D[W-1,3]	D[W-1,4]	D[W-1,5]	D[W-1,6]	D[W-1,7]
W + 2	D[W,0]	D[W,1]	D[W,2]	D[W,3]	D[W,4]	D[W,5]	D[W,6]	D[W,7]
W + 3	D[W+1,0]	D[W+1,1]	D[W+1,2]	D[W+1,3]	D[W+1,4]	D[W+1,5]	D[W+1,6]	D[W+1,7]
.	.	.	.	.	.	.	.	.
2xW + 1	D[2xW-1,0]	D[2xW-1,1]	D[2xW-1,2]	D[2xW-1,3]	D[2xW-1,4]	D[2xW-1,5]	D[2xW-1,6]	D[2xW-1,7]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
(H-1)xW + 1	D[(H-1)xW-1,0]	D[(H-1)xW-1,1]	D[(H-1)xW-1,2]	D[(H-1)xW-1,3]	D[(H-1)xW-1,4]	D[(H-1)xW-1,5]	D[(H-1)xW-1,6]	D[(H-1)xW-1,7]
(H-1)xW + 2	D[(H-1)xW,0]	D[(H-1)xW,1]	D[(H-1)xW,2]	D[(H-1)xW,3]	D[(H-1)xW,4]	D[(H-1)xW,5]	D[(H-1)xW,6]	D[(H-1)xW,7]
.	.	.	.	.	.	.	.	.
HxW + 1	D[HxW-1,0]	D[HxW-1,1]	D[HxW-1,2]	D[HxW-1,3]	D[HxW-1,4]	D[HxW-1,5]	D[HxW-1,6]	D[HxW-1,7]

Table 21.34 8BPP Pixel Data Sequence, B=1 (MSB first, P is don't care)

Byte Sequence	Serial Data Bits (SD7 sent first)							
	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
1	Command Byte[7:0]							
2	D[0,7]	D[0,6]	D[0,5]	D[0,4]	D[0,3]	D[0,2]	D[0,1]	D[0,0]
3	D[1,7]	D[1,6]	D[1,5]	D[1,4]	D[1,3]	D[1,2]	D[1,1]	D[1,0]
.	.	.	.	.	.	.	.	.
W + 1	D[W-1,7]	D[W-1,6]	D[W-1,5]	D[W-1,4]	D[W-1,3]	D[W-1,2]	D[W-1,1]	D[W-1,0]
W + 2	D[W,7]	D[W,6]	D[W,5]	D[W,4]	D[W,3]	D[W,2]	D[W,1]	D[W,0]
W + 3	D[W+1,7]	D[W+1,6]	D[W+1,5]	D[W+1,4]	D[W+1,3]	D[W+1,2]	D[W+1,1]	D[W+1,0]
.	.	.	.	.	.	.	.	.
2xW + 1	D[2xW-1,7]	D[2xW-1,6]	D[2xW-1,5]	D[2xW-1,4]	D[2xW-1,3]	D[2xW-1,2]	D[2xW-1,1]	D[2xW-1,0]
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
(H-1)xW + 1	D[(H-1)xW-1,7]	D[(H-1)xW-1,6]	D[(H-1)xW-1,5]	D[(H-1)xW-1,4]	D[(H-1)xW-1,3]	D[(H-1)xW-1,2]	D[(H-1)xW-1,1]	D[(H-1)xW-1,0]
(H-1)xW + 2	D[(H-1)xW,7]	D[(H-1)xW,6]	D[(H-1)xW,5]	D[(H-1)xW,4]	D[(H-1)xW,3]	D[(H-1)xW,2]	D[(H-1)xW,1]	D[(H-1)xW,0]
.	.	.	.	.	.	.	.	.
HxW + 1	D[HxW-1,7]	D[HxW-1,6]	D[HxW-1,5]	D[HxW-1,4]	D[HxW-1,3]	D[HxW-1,2]	D[HxW-1,1]	D[HxW-1,0]

21.6.2.5 Frame buffer-to-panel rotation

For 6-bit color, SPI, and EPD panels, the frame buffer image can be rotated 0, 90, 180, or 270 degrees when the panel is updated by setting MDCDISPCTL.ROTSEL[1:0].

For 0 and 180-degree rotation, the width and height registers should programmed as follows:
MDCGFXOWIDTH.OWIDTH[9:0] = MDCDISPWIDTH.DISPWIDTH[9:0] = [panel width]
MDCGFXOHEIGHT.OHEIGHT[9:0] = MDCDISPHEIGHT.DISPHEIGHT[9:0] = [panel height]
MDCGFXOWIDTH.OSTRIDE[9:0] = MDCDISPSTRIDE.DISPSTRIDE[9:0] = frame buffer image stride

For 90 and 270-degree rotation, the width and height registers should programmed as follows:
MDCGFXOWIDTH.OWIDTH[9:0] = MDCDISPWIDTH.DISPHEIGHT[9:0] = [panel height]
MDCGFXOHEIGHT.OHEIGHT[9:0] = MDCDISPHEIGHT.DISPWIDTH[9:0] = [panel width]
MDCGFXOWIDTH.OSTRIDE[9:0] = MDCDISPSTRIDE.DISPSTRIDE[9:0] = frame buffer image stride

21.6.3 Drawing Engine

The drawing engine draws a line, rectangle, or ellipse, to a frame buffer. The top left corner of the rectangular frame buffer has a reference coordinate of (0, 0), and the X and Y coordinate parameters for the drawing functions are positive values relative to the reference coordinate. X coordinates are positive values to the right of the top left corner and Y coordinates are positive values below the top left corner.

Note: The drawing functions do not update the display. It is necessary to execute the display update function.

Line Drawing

The line drawing function draws a line with a specified thickness and line color.
The following shows a procedure to draw a line:

1. Set the MDCGFXOBADDR0.OBASEADDR[15:0] and MDCGFXOBADDR1.OBASEADDR[31:16] bits. (Specify frame buffer base address)
2. Set the MDCGFXOSTRIDE.OSTRIDE[9:0] bits. (Specify frame buffer stride)
3. Set the MDCGFXIXCENTER.IXCENTER[9:0] bits. (Specify X coordinate (X1) of starting point of line)*1
4. Set the MDCGFXIYCENTER.IYCENTER[9:0] bits. (Specify Y coordinate (Y1) of starting point of line)*1
5. Set the MDCGFXOXCENTER.OXCENTER[9:0] bits. (Specify X coordinate (X2) of ending point of line)*1
6. Set the MDCGFXOYCENTER.OYCENTER[9:0] bits. (Specify Y coordinate (Y2) of ending point of line)*1
   *1 Specify coordinates relative to the frame buffer top left corner coordinates of (0, 0).
7. Set the MDCGFXIWIDTH.IWIDTH[9:0] bits. (Specify line thickness)
8. Set the MDCGFXCOLOR.COLOR[7:0] bits. (Specify pen color/gray level)
9. Set the following bits when using the interrupt:
   - Write 1 to the MDCINTCTL.GFXIF bit. (Clear interrupt flag)
   - Set the MDCINTCTL.GFXIE bit to 1. (Enable MDC interrupt)
10. Set the MDCGFXCTL.GFXFUNC[2:0] bits to 0x3. (Select line drawing function)
11. Write 1 to the MDCTRIGCTL.GFXTRIG bit. (Start line drawing)

When the line drawing is finished, the MDCTRIGCTL.GFXTRIG is automatically cleared to 0 and the MDCINTCTL.GFXIF bit is set to 1.
The following diagram shows an example of line drawing on a 180 × 180 6-bit color panel.

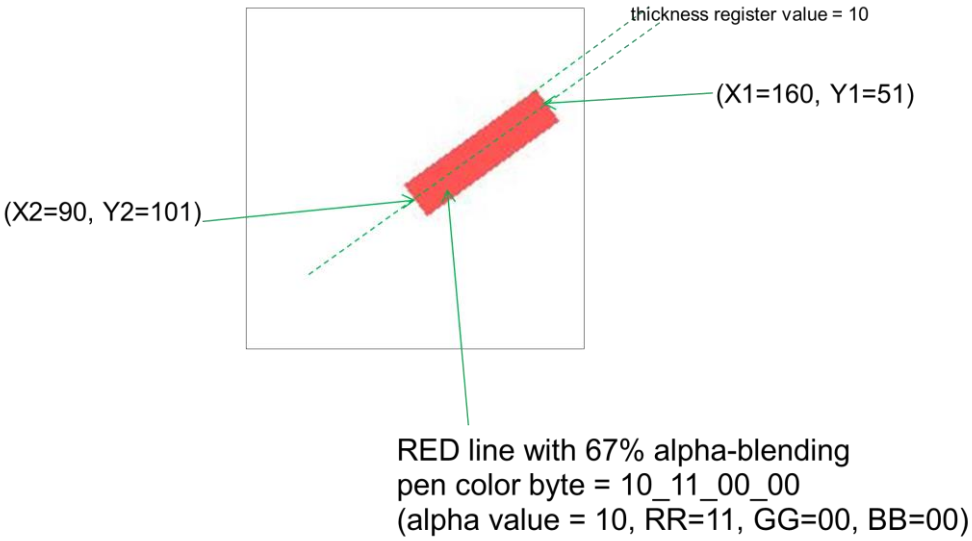


Figure 21.42 Line Drawing Example

The line thickness (MDCGFXIWIDTH.IWIDTH[9:0] bits) value is 10 but actual thickness is 2 × 10 + 1 = 21 pixels (2 × MDCGFXIWIDTH.IWIDTH[9:0] bits + 1).

Rectangle Drawing

The rectangle drawing function draws a filled or unfilled rectangle.
The following shows a procedure to draw a rectangle:

1. Set the MDCGFXOBADDR0.OBASEADDR[15:0] and MDCGFXOBADDR1.OBASEADDR[31:16] bits. (Specify frame buffer base address)
2. Set the MDCGFXOSTRIDE.OSTRIDE[9:0] bits. (Specify frame buffer stride)
3. Set the MDCGFXIXCENTER.IXCENTER[9:0] bits. (Specify X coordinate (X1) of top left corner of rectangle)*1
4. Set the MDCGFXIYCENTER.IYCENTER[9:0] bits. (Specify Y coordinate (Y1) of top left corner of rectangle)*1
5. Set the MDCGFXOXCENTER.OXCENTER[9:0] bits. (Specify X coordinate (X2) of bottom right corner of rectangle)*1
6. Set the MDCGFXOYCENTER.OYCENTER[9:0] bits. (Specify Y coordinate (Y2) of bottom right corner of rectangle)*1

   *1 Specify coordinates relative to the frame buffer top left corner coordinates of (0, 0).
7. Set the MDCGFXIWIDTH.IWIDTH[9:0] bits. (Specify vertical line thickness for unfilled rectangle drawing)*2
8. Set the MDCGFXIHEIGHT.IHEIGHT[9:0] bits. (Specify horizontal line thickness for unfilled rectangle drawing)*2

   *2 If vertical line thickness > [((X2 – X1) – 1) / 2], it is internally clipped to [((X2 – X1) – 1) / 2].
   If horizontal line thickness > [((Y2 – Y1) – 1) / 2], it is internally clipped to [((Y2 – Y1) – 1) / 2].
9. Set the MDCGFXCOLOR.COLOR[7:0] bits. (Specify pen color/gray level)*3

   *3 These bits specify the fill color/gray level for filled rectangle or the edge line color/gray level for unfilled rectangle.
10. Set the following bits when using the interrupt:
    - Write 1 to the MDCINTCTL.GFXIF bit. (Clear interrupt flag)
    - Set the MDCINTCTL.GFXIE bit to 1. (Enable MDC interrupt)
11. Configure the following MDCGFXCTL register bits:
    - Set the MDCGFXCTL.GFXFUNC[2:0] bits to 0x2. (Select rectangle drawing function)
    - MDCGFXCTL.FILLEN bit (Enable/disable fill option)
12. Write 1 to the MDCTRIGCTL.GFXTRIG bit. (Start rectangle drawing)

When the rectangle drawing is finished, the MDCTRIGCTL.GFXTRIG is automatically cleared to 0 and the MDCINTCTL.GFXIF bit is set to 1.
The following diagram shows an example of filled and unfilled rectangle drawing on a 6-bit color panel.

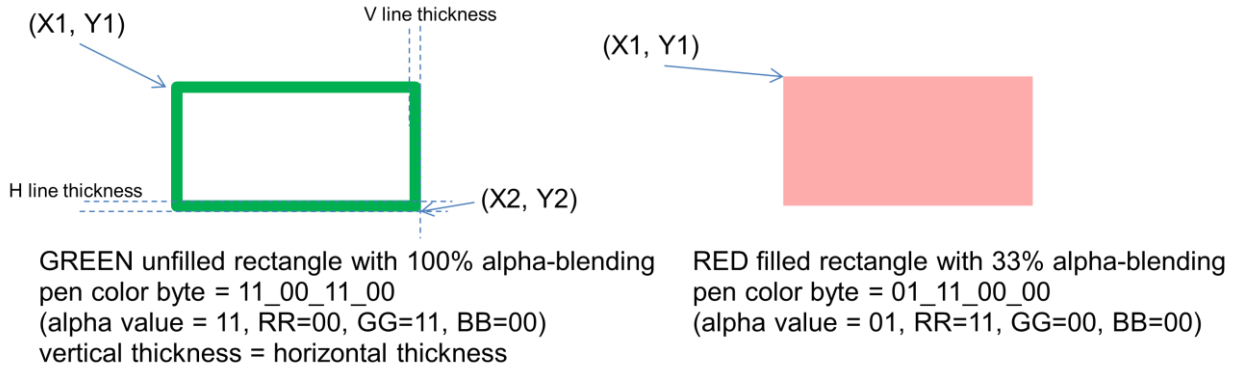


Figure 21.43 Rectangle Drawing Example

Ellipse Drawing

The ellipse drawing function draws a filled or unfilled ellipse.
The following shows a procedure to draw an ellipse:

1. Set the MDCGFXOBADDR0.OBASEADDR[15:0] and MDCGFXOBADDR1.OBASEADDR[31:16] bits. (Specify frame buffer base address)
2. Set the MDCGFXOSTRIDE.OSTRIDE[9:0] bits. (Specify frame buffer stride)
3. Set the MDCGFXIXCENTER.IXCENTER[9:0] bits. (Specify X coordinate of ellipse center)*1
4. Set the MDCGFXIYCENTER.IYCENTER[9:0] bits. (Specify Y coordinate of ellipse center)*1
   *1 Specify coordinates relative to the frame buffer top left corner coordinates of (0, 0).
5. Set the MDCGFXOXCENTER.OXCENTER[9:0] bits. (Specify X radius (XR))*2
6. Set the MDCGFXOYCENTER.OYCENTER[9:0] bits. (Specify Y radius (YR))*2
   *2 XR > 0, YR > 0. If XR = 0 or YR = 0, nothing happens.
7. Set the MDCGFXIWIDTH.IWIDTH[9:0] bits. (Specify thickness of X-axis crossing for unfilled ellipse drawing)*3
8. Set the MDCGFXIHEIGHT.IHEIGHT[9:0] bits. (Specify thickness of Y-axis crossing for unfilled ellipse drawing)*3
   *3 The line thickness is "tapered" in the transition between the X-axis and Y-axis crossings.
   If XR < 3, the thickness of X-axis crossing is internally clipped to 0.
   If YR < 3, the thickness of Y-axis crossing is internally clipped to 0.
   Thickness of X-axis crossing ≤ (XR/2)
   Thickness of Y-axis crossing ≤ (YR/2)
9. Set the MDCGFXCOLOR.COLOR[7:0] bits. (Specify pen color/gray level)*4
   *4 These bits specify the fill color/gray level for filled ellipse or the edge line color/gray level for unfilled ellipse.
10. Set the following bits when using the interrupt:
    - Write 1 to the MDCINTCTL.GFXIF bit. (Clear interrupt flag)
    - Set the MDCINTCTL.GFXIE bit to 1. (Enable MDC interrupt)
11. Configure the following MDCGFXCTL register bits:
    - Set the MDCGFXCTL.GFXFUNC[2:0] bits to 0x4. (Select ellipse drawing function)
    - MDCGFXCTL.FILLEN bit (Enable/disable fill option)
12. Write 1 to the MDCTRIGCTL.GFXTRIG bit. (Start ellipse drawing)

When the ellipse drawing is finished, the MDCTRIGCTL.GFXTRIG is automatically cleared to 0 and the MDCINTCTL.GFXIF bit is set to 1.
The following diagram shows an example of filled and unfilled ellipse drawing on a 6-bit color panel.

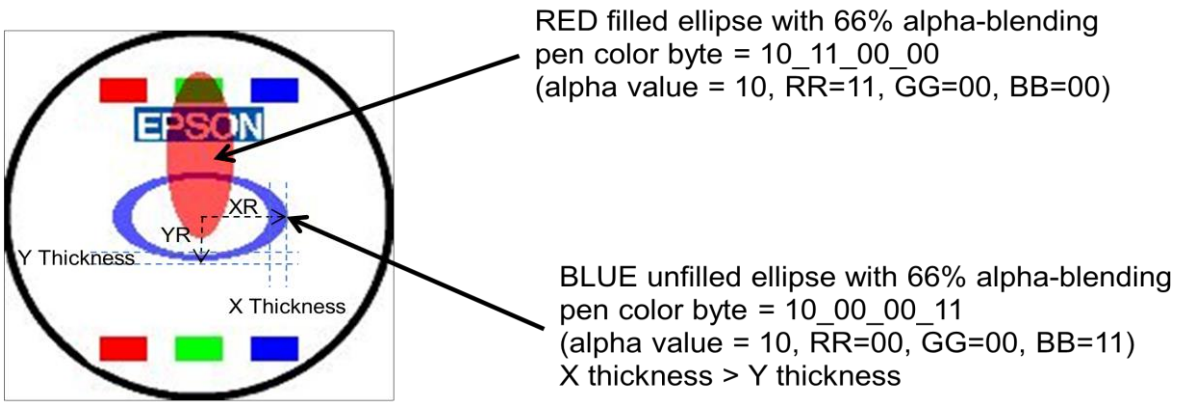


Figure 21.44 Ellipse Drawing Example

21.6.4 Copy Engine

Overview and Definitions

The copy engine copies pixels from a window of a source image or bitmap to a window of a destination image with transformation (scaling + rotation or horizontal/vertical shear).

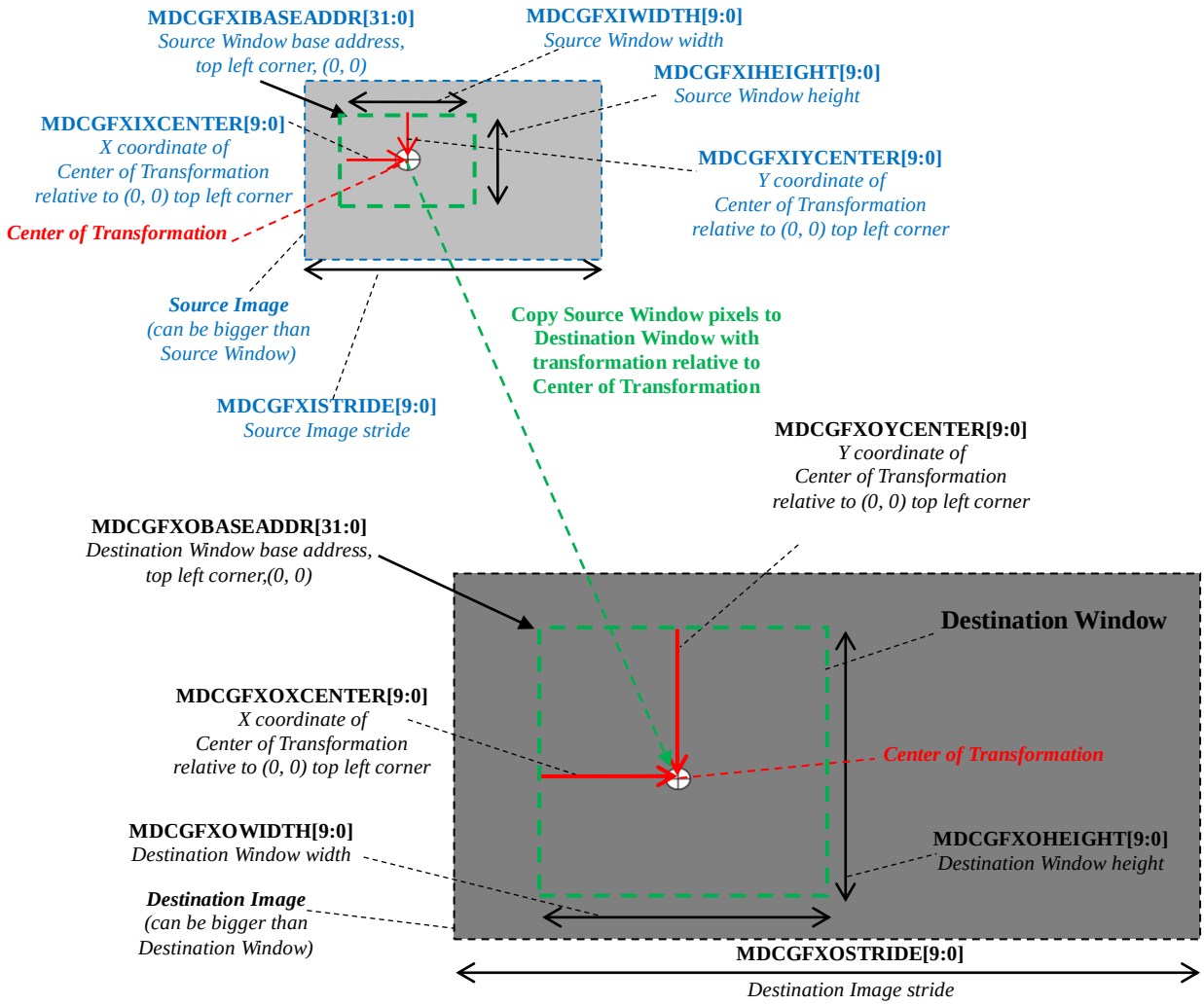


Figure 21.45 Copy Engine Definitions

Note: The drawing functions do not update the display. It is necessary to execute the display update function.

Source Image and Window

Pixels are copied from a window within the source image (or bitmap) to a window within the destination image with transformation applied. Typically, the source window is the same size as the source image, but it can be smaller to allow for panning within the source image. The source window and image are defined by the following parameters:

- Source (input) window base address (MDCGFXIBADDR1/0.IBASEADDR[31:0] bits)
  - This points to the memory location of the top left corner of the source window.
  - The top left corner of the source window has coordinates of (0, 0).
  - Panning of the source image is achieved by changing this base address.
- Source (input) window width (MDCGFXIWIDTH.IWIDTH[9:0] bits)
  - This is the width of the source window in pixels.
- Source (input) window height (MDCGFXIHEIGHT.IHEIGHT[9:0] bits)
  - This is the height of the source window in pixels.
- Source (input) image stride (MDCGFXISTRIDE.ISTRIDE[9:0] bits)
  - This is the stride (number of pixels per line) of the source image.
- Source (input) window center of transformation X coordinate (MDCGFXIXCENTER.IXCENTER[9:0]

bits)
- This is the X coordinate, in pixels relative to the top left corner coordinates of (0, 0), of the center of transformation for the source window.
- This is a positive value which is to the right of the (0, 0) top left corner.
- The transformation (scale, rotate, shear) is applied relative to this center.

- Source (input) window center of transformation Y coordinate (MDCGFXIYCENTER.IYCENTER[9:0] bits)
  - This is the Y coordinate, in pixels relative to the top left corner coordinates of (0, 0), of the center of transformation for the source window.
  - This is a positive value which is below the (0, 0) top left corner.
  - The transformation (scale, rotate, shear) is applied relative to this center.

Destination Image and Window

The destination image can be bigger than the destination window to allow for panning of the destination image. Source pixels, after applying the transformation, that are outside the destination window are not copied (written). The copy engine calculates an output window when the transformation is applied to the source window. (The results of the calculations can be read from the MDCGFXOWLEFT.OWLEFT[9:0], MDCGFXOWRIGHT.OWRIGHT[9:0], MDCGFXOWTOP.OWTOP[9:0], and MDCGFXOWBOT.OWBOT[9:0] bits.) If the output window is within the destination window, then pixels are copied only to the smaller output window within the destination window. If any of the output window edges fall outside the destination window edges, they are clipped to the destination window edges and pixels are not copied to locations outside the destination window.

The destination window and image are defined by the following parameters:

- Destination (output) window base address (MDCGFXOBASEADDR1/0.OBASEADDR[31:0] bits)
  - This points to the memory location of the top left corner of the destination window.
  - The top left corner of the destination window has coordinates of (0, 0).
  - Panning of the destination image is achieved by changing this base address.
- Destination (output) window width (MDCGFXOWIDTH.OWIDTH[9:0] bits)
  - This is the width of the destination window in pixels.
- Destination (output) window height (MDCGFXOHEIGHT.OHEIGHT[9:0] bits)
  - This is the height of the destination window in pixels.
- Destination (output) image stride (MDCGFXOSTRIDE.OSTRIDE[9:0] bits)
  - This is the stride (number of pixels per line) of the destination image.
- Destination (output) window center of transformation X coordinate (MDCGFXOXCENTER.OXCENTER[9:0] bits)
  - This is the X coordinate, in pixels relative to the top left corner coordinates of (0, 0), of the center of transformation for the destination window.
  - This is a positive value which is to the right of the (0, 0) top left corner.
  - The transformed image in the destination window is centered at this location.
- Destination (output) window center of transformation Y coordinate (MDCGFXOYCENTER.OYCENTER[9:0] bits)
  - This is the Y coordinate, in pixels relative to the top left corner coordinate of (0, 0), of the center of transformation for the destination window.
  - This is a positive value which is below the (0, 0) top left corner.
  - The transformed image in the destination window is centered at this location.

Source Pixel Alpha Channel

The source pixels for the SPI 1-bit black-and-white and 3-bit color formats contain a 1-bit alpha channel which specifies whether or not the source pixel should be copied. A 0 in the alpha channel indicates that the pixel should not be copied to the destination and a 1 indicates that it should be copied. Effectively, the alpha channel bit specifies pixel transparency.

The source pixels for the 6-bit color format contain a 2-bit alpha channel. The pixel value copied to the destination location is the alpha-blend of the destination pixel with the source pixel based on the 2-bit alpha channel value of the source pixel. A value of 0b00 (0%) means the source pixel is not copied and a value of 0b11 (100%) means that the destination pixel is over-written with the source pixel. A value of 0b01 means 33% blending ratio for the source pixel and a value of 0b10 means 67% blending ratio.

The MDCGFXCTL.ALPHAOVRRD bit can be used to override the alpha channel value of the source pixels. When the MDCGFXCTL.ALPHAOVRRD bit is 1, the MDCGFXCTL.ALPHAVAL[1:0] bits are used as the alpha value for all the source pixels.

Source Bitmaps

The MDCGFXCTL.BITMAPEN bit is used to specify whether the source is a regular image (0) or a bitmap (1). If the MDCGFXCTL.BITMAPEN bit is 1 which selects bitmap, the MDCGFXCTL.BITMAPFMT bit specifies whether the bitmap format is 1-bit per pixel or 2-bit per pixel.

For the 1-bit format, a value of 0 means the source pixel is transparent (not copied to the destination location), and a value of 1 means the pixel color specified by the MDCGFXCOLOR.COLOR[7:0] bits (dependent on which panel color format is selected) is written to the destination location.

The 2-bit bitmap format is only applicable to the 6-bit color panel. For the 2-bit format, the bitmap value specifies the alpha-blending ratio of the destination pixel with the pixel color specified by the MDCGFXCOLOR.COLOR[5:0] bits. Bitmaps are typically used for fonts or simple icons/shapes and the 2-bit format can be used to provide “smoothing” of edges by giving the edge pixels a 33% or 67% alpha-blending value.

Fill Option

For the copy engine, if the MDCGFXCTL.FILLEN bit is 1, all source pixels are overridden with a fill color specified by the MDCGFXCOLOR.COLOR[7:0] bits, and, effectively, the destination pixels are filled with the MDCGFXCOLOR.COLOR[7:0] bits having the “shape” of the source non-transparent image or bitmap pixels.

Source Window Horizontal and Vertical Flip

The MDCGFXCTL.CPYNEGX and MDCGFXCTL.CPYNEGY bits can be used to negate (flip) the X and Y coordinate values of the source window pixels relative to the center of transformation. If the MDCGFXCTL.CPYNEGX bit is 1, the source window image is flipped about the Y-axis that runs through the source window’s center of transformation. If the MDCGFXCTL.CPYNEGY bit is 1, the source window image is flipped about the X-axis that runs through the source window’s center of transformation.

Notes:

- When the MDCGFXCTL.CPYNEGX bit = 1, the MDCGFXIXCENTER.IXCENTER[9:0] bits must be set to the horizontal center of the source image.
- When the MDCGFXCTL.CPYNEGY bit = 1, the MDCGFXIYCENTER.IYCENTER[9:0] bits must be set to the vertical center of the source image.

Image/Bitmap Copying with Scaling and Rotation

The COPYROTSCALE function copies pixels from a source window to a destination window with rotation and scaling transformation. The rotation and scaling transformations are combined when the COPYROTSCALE function is triggered.
The following shows a procedure to execute the COPYROTSCALE function:

Window configuration

1. Set the MDCGFXIBADDR0.IBASEADDR[15:0] and MDCGFXIBADDR1.IBASEADDR[31:16] bits. (Specify source window base address)
2. Set the MDCGFXIXCENTER.IXCENTER[9:0] bits. (Specify X coordinate of center of rotation/scaling in source window)
3. Set the MDCGFXIYCENTER.IYCENTER[9:0] bits. (Specify Y coordinate of center of rotation/scaling in source window)
4. Set the MDCGFXIWIDTH.IWIDTH[9:0] bits. (Specify width of source window)
5. Set the MDCGFXIHEIGHT.IHEIGHT[9:0] bits. (Specify height of source window)
6. Set the MDCGFXISTRIDE.ISTRIDE[9:0] bits. (Specify source image stride)
7. Set the MDCGFXOBADDR0.OBASEADDR[15:0] and MDCGFXOBADDR1.OBASEADDR[31:16] bits. (Specify destination window base address)
8. Set the MDCGFXOXCENTER.OXCENTER[9:0] bits. (Specify X coordinate of center of rotation/scaling in destination window)
9. Set the MDCGFXOYCENTER.OYCENTER[9:0] bits. (Specify Y coordinate of center of rotation/scaling in destination window)
10. Set the MDCGFXOWIDTH.OWIDTH[9:0] bits. (Specify width of destination window)
11. Set the MDCGFXOHEIGHT.OHEIGHT[9:0] bits. (Specify height of destination window)
12. Set the MDCGFXOSTRIDE.OSTRIDE[9:0] bits. (Specify destination image stride)

Setting for rotation

13. Set the MDCGFXROTVAL.ROTVAL[8:0] bits. (Specify rotation angle)

Settings for scaling

14. Set the MDCGFXXLSCALE.XLSCALE[13:0] bits. (Specify left half scaling factor)
15. Set the MDCGFXXRSCALE.XRSCALE[13:0] bits. (Specify right half scaling factor)
16. Set the MDCGFXYTSCALE.YTSCALE[13:0] bits. (Specify top half scaling factor)
17. Set the MDCGFXYBSCALE.YBSCALE[13:0] bits. (Specify bottom half scaling factor)

Executing COPYROTSCALE function

18. Set the MDCGFXCOLOR.COLOR[7:0] bits. (Specify fill color/gray level)*1
    *1 Specify only when the MDCGFXCTL.BITMAPEN bit = 1 or the MDCGFXCTL.FILLEN bit = 1.
19. Set the following bits when using the interrupt:
    - Write 1 to the MDCINTCTL.GFXIF bit. (Clear interrupt flag)
    - Set the MDCINTCTL.GFXIE bit to 1. (Enable MDC interrupt)
20. Configure the following MDCGFXCTL register bits:
    - Set the MDCGFXCTL.GFXFUNC[2:0] bits to 0x0. (Select COPYROTSCALE function)
    - MDCGFXCTL.ALPHAOVRRD bit (Enable/disable alpha value override)
    - MDCGFXCTL.ALPHAVAL[1:0] bit (Specify override alpha value)
    - MDCGFXCTL.BITMAPEN bit (Select image copy or bitmap copy)
    - MDCGFXCTL.BITMAPFMT bit (Select bitmap format (1-bit or 2-bit))
    - MDCGFXCTL.FILLEN bit (Enable/disable fill option)
    - MDCGFXCTL.CPYNEGX bit (Enable/disable horizontal flip option)
    - MDCGFXCTL.CPYNEGY bit (Enable/disable vertical flip option)
21. Write 1 to the MDCTRIGCTL.GFXTRIG bit. (Start copying)

When the copying is finished, the MDCTRIGCTL.GFXTRIG is automatically cleared to 0 and the MDCINTCTL.GFXIF bit is set to 1.

Rotation Parameter

The MDCGFXROTVAL.ROTVAL[8:0] bits specify the counterclockwise angle of rotation of the source image about its center of transformation. The value of the MDCGFXROTLVAL.ROTLVAL[8:0] bits can be calculated from a degree value by the following formula:

MDCGFXROTVAL[8:0] = (<*angle in degrees*> x 512) / 360

Scaling Parameters

There are four scaling parameters corresponding to top half (MDCGFXYTSCALE.YTSCALE[13:0] bits), bottom half (MDCGFXYBSCALE.YBSCALE[13:0] bits), left half (MDCGFXXLSCALE.XLSCALE[13:0] bits), and right half (MDCGFXXRSCALE.XRSCALE[13:0] bits) of the source window relative to the center of transformation. The scaling factor is the value of the parameter divided by 256. For example, if the value is 512, the scaling is × 2 (double the size).

The following diagram shows an example of image copy with asymmetric scaling and rotation of 45 degrees counterclockwise:

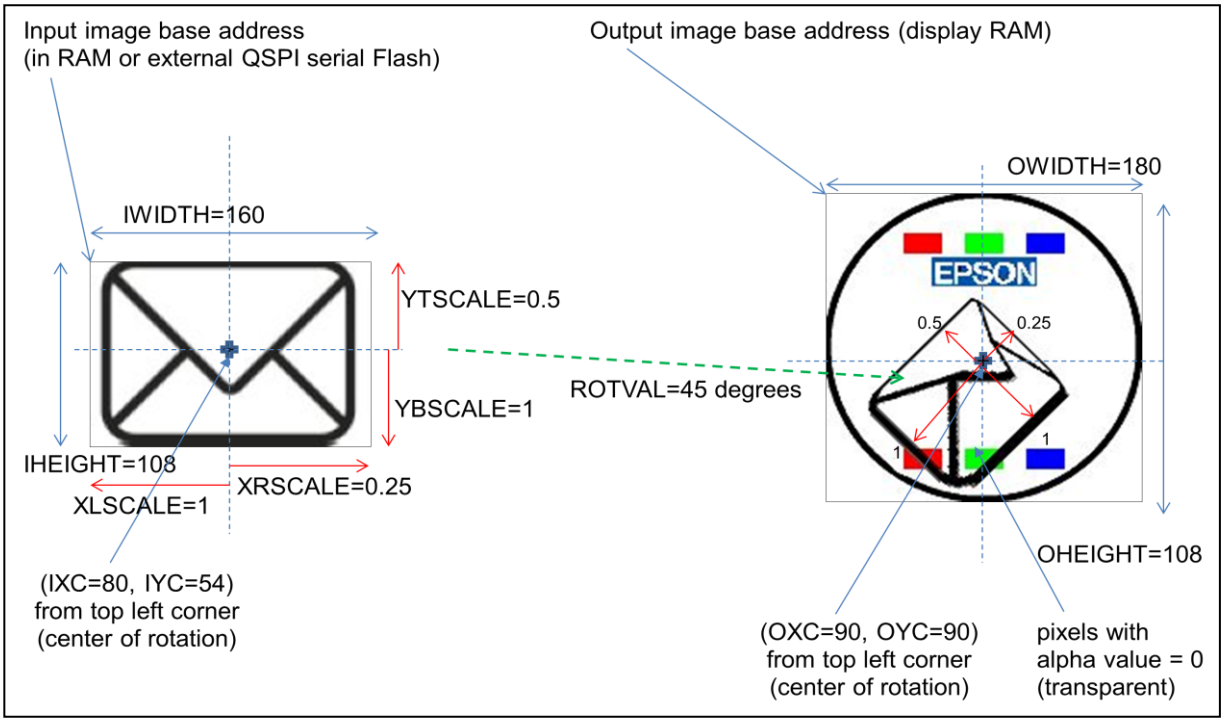


Figure 21.46 Image Copy with Scaling and Rotation Example

The following diagram shows an example of image copy with the use of the MDCGFXCTL.FILLEN bit = 1 (solid fill) to perform "erase" function of the output image:

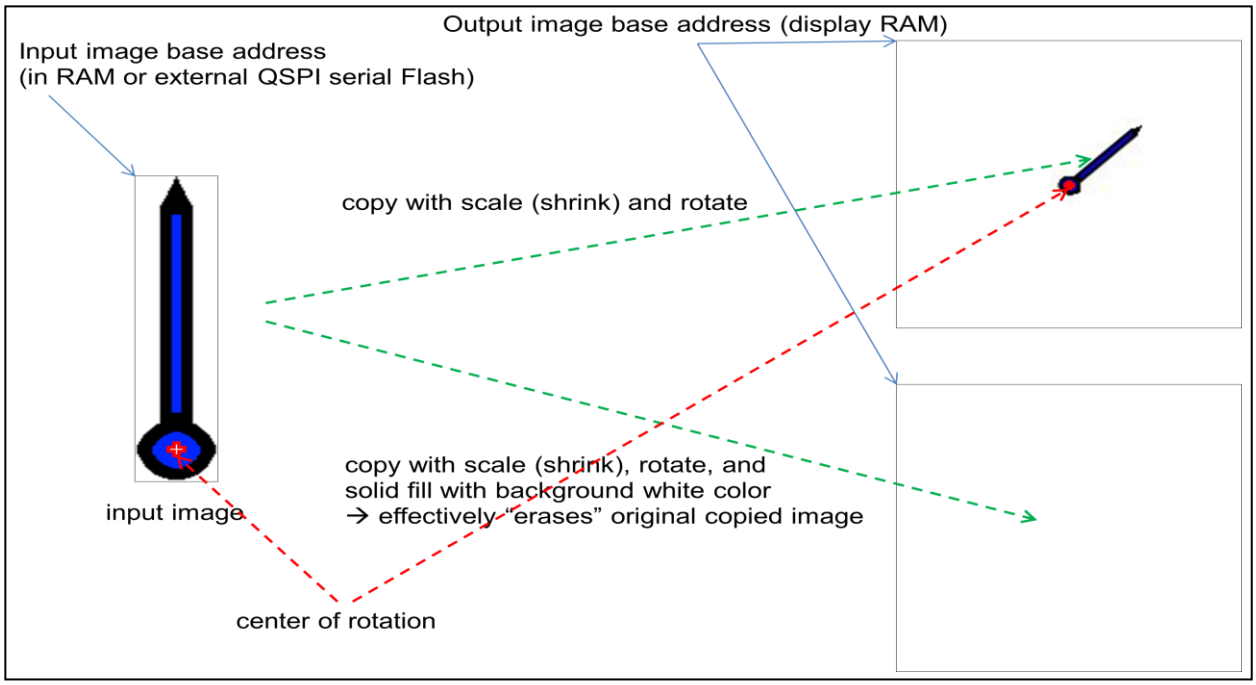


Figure 21.47 Example of Image Copy with Fill Enabled

The following diagram shows an example of bitmap copy with rotation of 90 degrees counterclockwise:

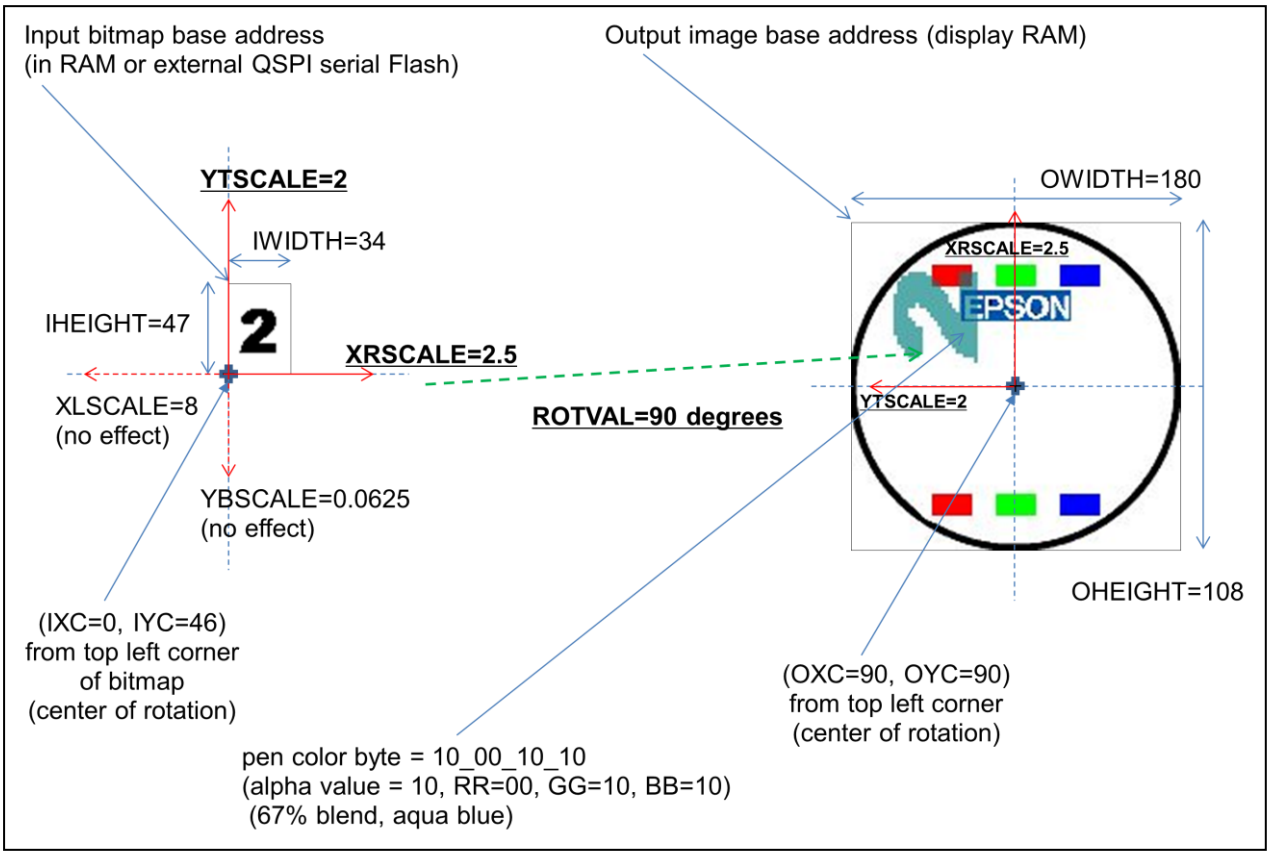


Figure 21.48 Bitmap Copy with Scaling and Rotation Example

Because of the location of the center of scaling/rotation of the input bitmap, the MDCGFXXLSCALE.XLSCALE[13:0] and MDCGFXYBSCALE.YBSCALE[13:0] bits have no effect.

Image/Bitmap Copying with Horizontal/Vertical Shear

The COPYHVSHEAR function copies a source/input image or bitmap to a destination/output location with horizontal and vertical shearing.
The following shows a procedure to execute the COPYHVSHEAR function:

Window configuration

1. Set the MDCGFXIBADDR0.IBASEADDR[15:0] and MDCGFXIBADDR1.IBASEADDR[31:16] bits. (Specify source window base address)
2. Set the MDCGFXIXCENTER.IXCENTER[9:0] bits. (Specify X coordinate of center of shearing in source window)
3. Set the MDCGFXIYCENTER.IYCENTER[9:0] bits. (Specify Y coordinate of center of shearing in source window)
4. Set the MDCGFXIWIDTH.IWIDTH[9:0] bits. (Specify width of source window)
5. Set the MDCGFXIHEIGHT.IHEIGHT[9:0] bits. (Specify height of source window)
6. Set the MDCGFXISTRIDE.ISTRIDE[9:0] bits. (Specify source image stride)
7. Set the MDCGFXOBADDR0.OBASEADDR[15:0] and MDCGFXOBADDR1.OBASEADDR[31:16] bits. (Specify destination window base address)
8. Set the MDCGFXOXCENTER.OXCENTER[9:0] bits. (Specify X coordinate of center of shearing in destination window)
9. Set the MDCGFXOYCENTER.OYCENTER[9:0] bits. (Specify Y coordinate of center of shearing in destination window)
10. Set the MDCGFXOWIDTH.OWIDTH[9:0] bits. (Specify width of destination window)
11. Set the MDCGFXOHEIGHT.OHEIGHT[9:0] bits. (Specify height of destination window)
12. Set the MDCGFXOSTRIDE.OSTRIDE[9:0] bits. (Specify destination image stride)

Setting for shearing

13. Configure the following MDCGFXSHEAR register bits:
    - MDCGFXSHEAR.XSHEAR[6:0] bits (Specify horizontal shearing factor)
    - MDCGFXSHEAR.YSHEAR[6:0] bits (Specify vertical shearing factor)

Executing COPYHVSHEAR function

14. Set the MDCGFXCOLOR.COLOR[7:0] bits. (Specify fill color/gray level)*1
    *1 Specify only when the MDCGFXCTL.BITMAPEN bit = 1 or the MDCGFXCTL.FILLEN bit = 1.
15. Set the following bits when using the interrupt:
    - Write 1 to the MDCINTCTL.GFXIF bit. (Clear interrupt flag)
    - Set the MDCINTCTL.GFXIE bit to 1. (Enable MDC interrupt)
16. Configure the following MDCGFXCTL register bits:
    - Set the MDCGFXCTL.GFXFUNC[2:0] bits to 0x1. (Select COPYHVSHEAR function)
    - MDCGFXCTL.ALPHAOVRRD bit (Enable/disable alpha value override)
    - MDCGFXCTL.ALPHAVAL[1:0] bit (Specify override alpha value)
    - MDCGFXCTL.BITMAPEN bit (Select image copy or bitmap copy)
    - MDCGFXCTL.BITMAPFMT bit (Select bitmap format (1-bit or 2-bit))
    - MDCGFXCTL.FILLEN bit (Enable/disable fill option)
    - MDCGFXCTL.SHEARNEGX bit (Enable/disable negative horizontal shear)
    - MDCGFXCTL.SHEARNEGY bit (Enable/disable negative vertical shear)
    - MDCGFXCTL.CPYNEGX bit (Enable/disable horizontal flip option)
    - MDCGFXCTL.CPYNEGY bit (Enable/disable vertical flip option)
17. Write 1 to the MDCTRIGCTL.GFXTRIG bit. (Start copying)

When the copying is finished, the MDCTRIGCTL.GFXTRIG is automatically cleared to 0 and the MDCINTCTL.GFXIF bit is set to 1.

Shear Parameters

The shearing transformation maps each source pixel coordinate (relative to the center of transformation of the source window) to a new coordinate in the destination window (relative to the center of transformation of the destination window) according to the following formulas:

$$OX = (IX + (XS/32) \times IY) \qquad OY = (IY + (YS/32) \times IX)$$

Where

OX: Destination pixel X coordinate relative to destination window center of transformation
OY: Destination pixel Y coordinate relative to destination window center of transformation
IX: Source pixel X coordinate relative to source window center of transformation
IY: Source pixel Y coordinate relative to source window center of transformation
XS: Horizontal shearing factor
XS = MDCGFXSHEAR.XSHEAR[6:0] (when the MDCGFXCTL.SHEARNEGX bit = 0)
XS = -MDCGFXSHEAR.XSHEAR[6:0] (when the MDCGFXCTL.SHEARNEGX bit = 1)
YS: Vertical shearing factor
YS = MDCGFXSHEAR.YSHEAR[6:0] (when the MDCGFXCTL.SHEARNEGY bit = 0)
YS = -MDCGFXSHEAR.YSHEAR[6:0] (when the MDCGFXCTL.SHEARNEGY bit = 1)

The following diagram shows an example of image copy with horizontal shear only (MDCGFXSHEAR.XSHEAR[6:0] bits = 32, MDCGFXSHEAR.YSHEAR[6:0] bits = 0, MDCGFXCTL.SHEARNEGX bit = 0, MDCGFXCTL.SHEARNEGY bit = 0):

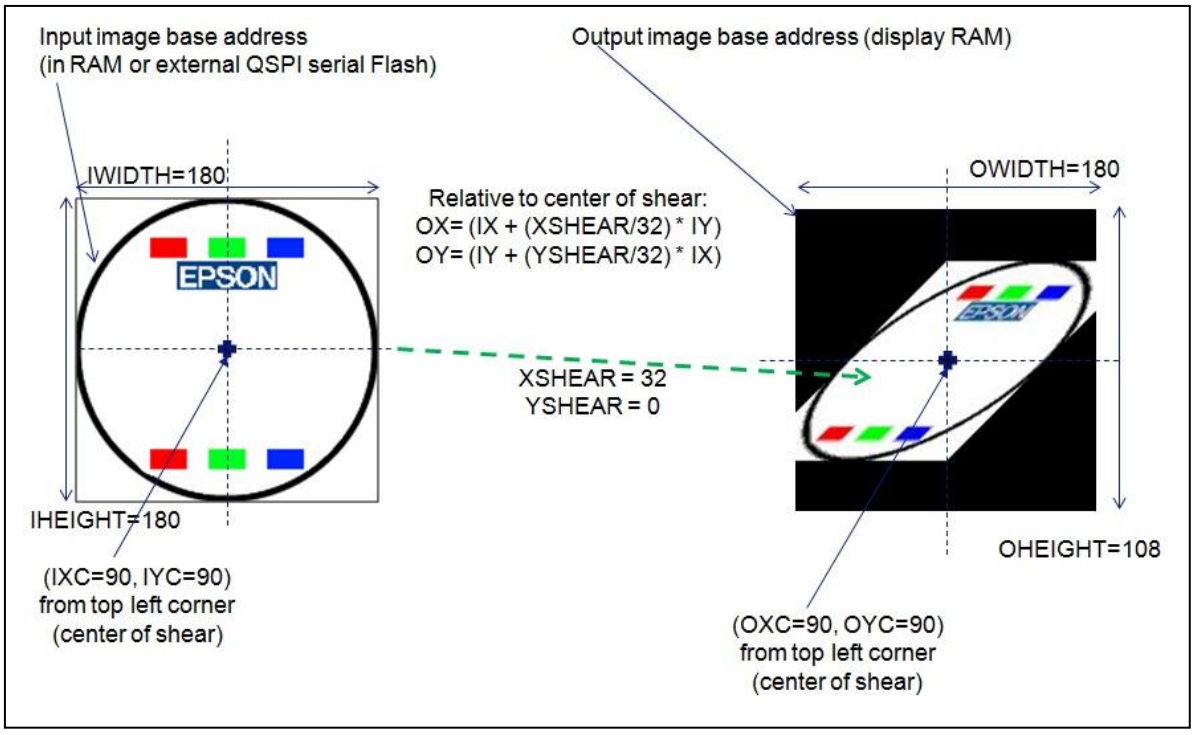


Figure 21.49 Image Copy with Horizontal Shear Example

The following diagram shows an example of image copy with vertical shear only MDCGFXSHEAR.XSHEAR[6:0] bits = 0, MDCGFXSHEAR.YSHEAR[6:0] bits = 32, MDCGFXCTL.SHEARNEGX bit = 0, MDCGFXCTL.SHEARNEGY bit = 0):

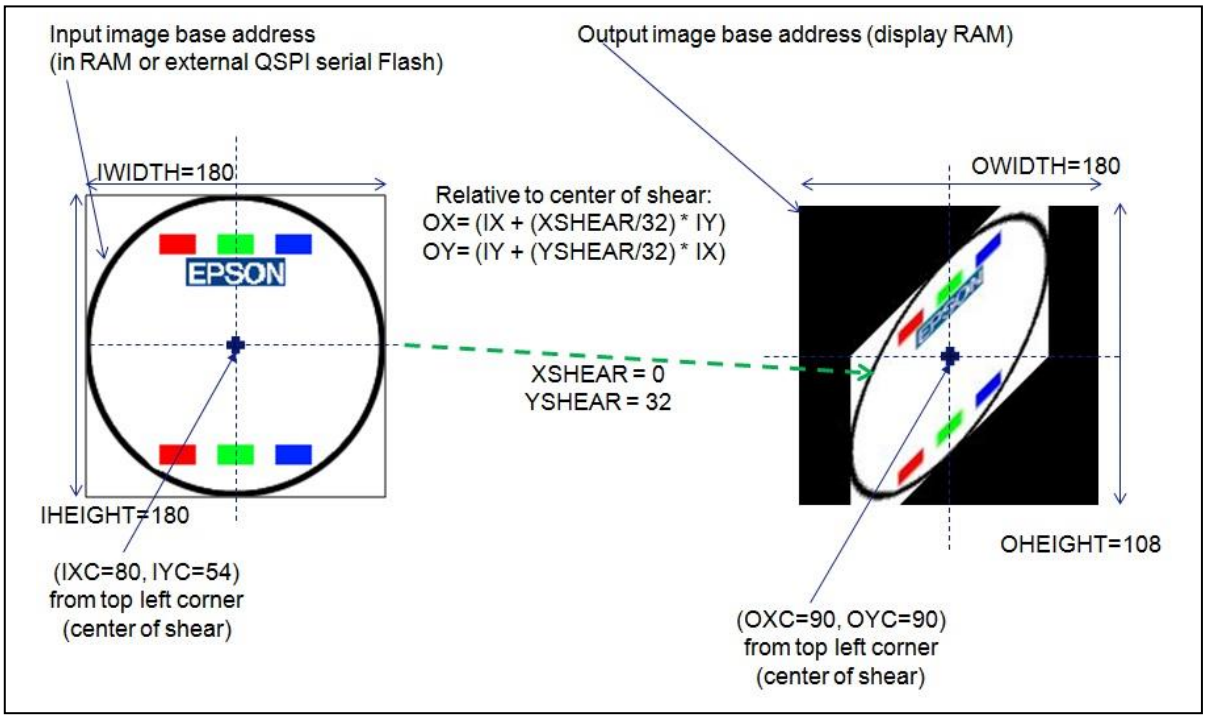


Figure 21.50 Image Copy with Vertical Shear Example

The following diagram shows an example of two bitmap copy operations, one with a horizontal-only shear and the other with a vertical-only shear, and each having different center of shear:

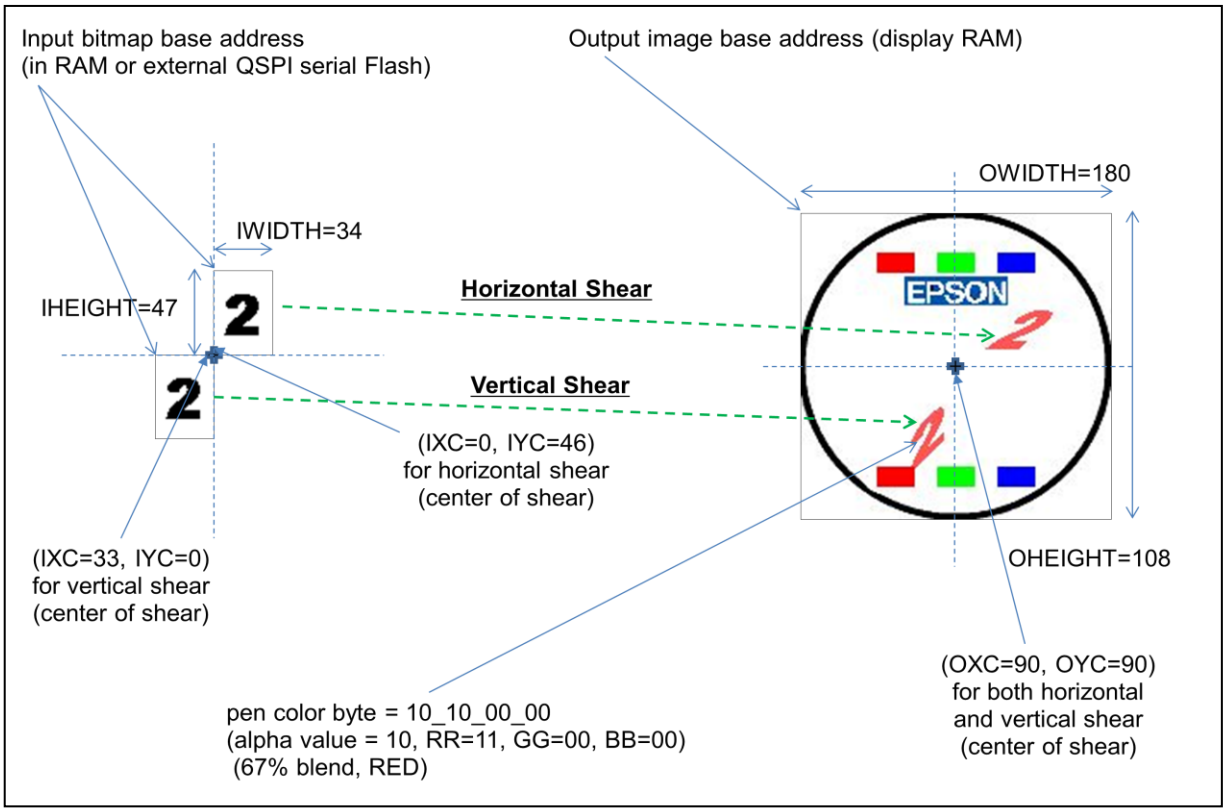


Figure 21.51 Image Copy with Horizontal and Vertical Shear Example

Notes:

- When the MDCGFXCTL.SHEARNEGX bit = 1, the MDCGFXIYCENTER.IYCENTER[9:0] bits must be set to the vertical center of the source image.
- When the MDCGFXCTL.SHEARNEGY bit = 1, the MDCGFXIXCENTER.IXCENTER[9:0] bits must be set to the horizontal center of the source image.

21.6.5 Event Processor

Overview

The Event Processor is a basic processor that has a limited instruction set (5-bit opcode). It can execute instructions when triggered by an interrupt event. The following block diagram shows the Event Processor system connections:

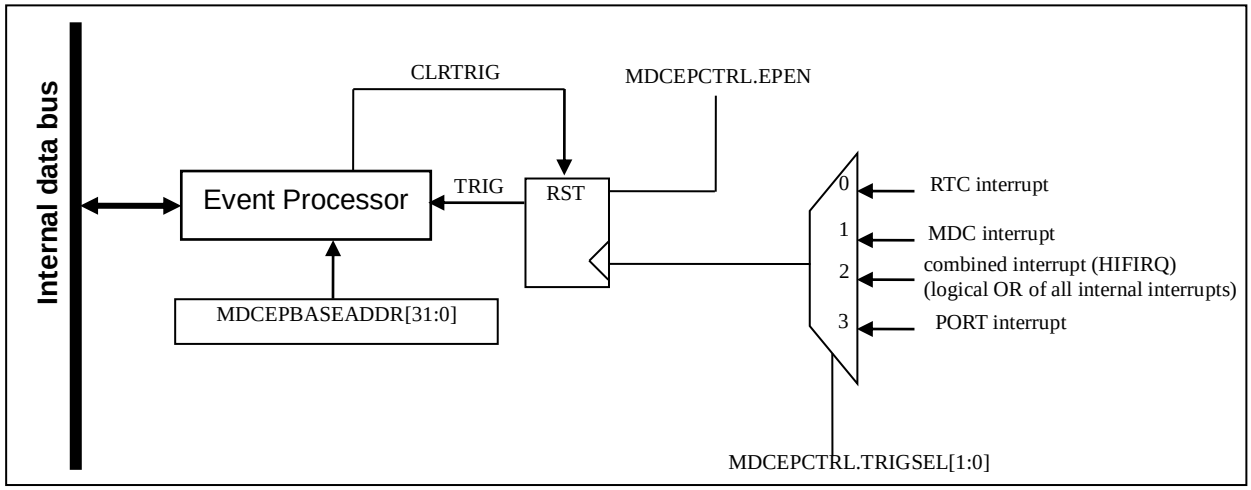


Figure 21.52 Event Processor Block Diagram

The Event Processor has access to the internal system bus and it can read and write to any location in the Memory Map. It can be used to independently execute tasks such as reading the RTC time values and updating the display without the need to wake up the host MCU. The Event Processor trigger is enabled by setting MDCEPCTRL.EPEN to 1.

The trigger signal is selectable from four sources by programming MDCEPCTRL.TRIGSEL[1:0]:
RTC interrupt (0), MDC interrupt (1), combined interrupt (2), and PORT (GPIO) interrupt (3).

The following shows the Event Processor state diagram:

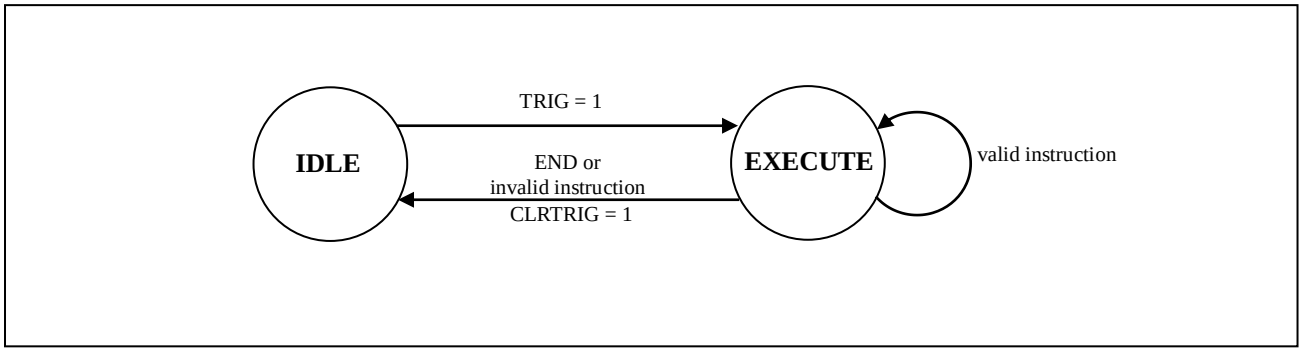


Figure 21.53 Event Processor State Diagram

After reset, the Event Processor trigger is disabled (MDCEPCTRL.EPEN = 0) and the Event Processor is in the IDLE state. When the Event Processor trigger is enabled (MDCEPCTRL.EPEN = 1), it waits for the trigger signal (TRIG) to go high. The trigger signal does high on the rising edge of the selected trigger source. When the trigger signal goes high, the Event Processor hardware turns on the System Clock and goes into the EXECUTE state to start fetching and executing instructions read from the starting base address specified by MDCEPBASEADDR[31:0]. The Event Processor will continue sequentially fetching and executing valid instructions until the END instruction or an invalid instruction code is read, at which time the Event Processor will

clear the TRIG signal (CLRTRIG=1), go back to IDLE state (waiting for the next trigger event), and turn off the system clock (only if SYSCTRL.IOSCEN=0).

NOTES:

1. While the Event Processor is in the EXECUTE state, any rising edge of the trigger source signal has no effect and will be missed. The time between the rising edges of the interrupt source is expected to be longer than the execution time of the Event Processor program. The Event Processor program is responsible for clearing the interrupt source signal to 0 so that the next interrupt event will generate a rising edge for the trigger signal.

2. The Event Processor cannot be forced back to IDLE state while it is in EXECUTE state. Clearing the MDCEPCTRL.EPEN bit to 0 while the Event Processor is in EXECUTE state will only disable the trigger source from setting the TRIG signal to 1. The Event Processor will continue to be in the EXECUTE state until an END or invalid instruction is encountered, at which point it will go to the IDLE state and turn off the system clock. If the Host software wants to monitor the status of the Event Processor, it is recommended that one of the Pxx port pins is used as an "Event Processor status" bit to a GPIO input of the Host MCU. The Event Processor code should set this port bit HIGH in the beginning and clear it to LOW before the END instruction.

Instruction Set and Registers

There are no "jump" or "branch" instructions in the Event Processor's instruction set and instructions are fetched and executed sequentially.

The Event Processor has two 32-bit data registers, **AREG**[31:0] and **BREG**[31:0], and a 24-bit address register, **ADDR**[31:8].

Each instruction is a byte with the upper 5 bits containing the instruction code and the lower 3 bits containing an optional parameter field. Depending on the instruction, the instruction byte is followed by optional parameter bytes.

The following table describes the instruction set for the Event Processor:

Table 21.35 Event Processor Instruction Set

INSTRUCTION	BYTE1		BYTE2	BYTE3	BYTE4	DESCRIPTION
	OPCODE	PARAM0				
LDADDR	00000b	-	addr[15:8]	addr[23:16]	addr[31:24]	Load **ADDR**[31:8] register with immediate 24-bit value (3 parameter bytes).
RDREG8I	00001b	<dest8>	<immval>	-	-	Load immediate byte value <imm val> into **AREG** or **BREG** (destination specified by <dest8>). <dest8> 000b -- destination is **AREG**[7:0] 001b -- destination is **AREG**[15:8] 010b -- destination is **AREG**[23:16] 011b -- destination is **AREG**[31:24] 100b -- destination is **BREG**[7:0] 101b -- destination is **BREG**[15:8] 110b -- destination is **BREG**[23:16] 111b -- destination is **BREG**[31:24]
RDREG8	00010b	<dest8>	<regaddr>	-	-	Load a byte into **AREG** or **BREG** (destination is specified by <dest8>) with value read from {**ADDR**[31:8], <regaddr>}. <dest8> 000b -- destination is **AREG**[7:0] 001b -- destination is **AREG**[15:8] 010b -- destination is **AREG**[23:16] 011b -- destination is **AREG**[31:24] 100b -- destination is **BREG**[7:0] 101b -- destination is **BREG**[15:8] 110b -- destination is **BREG**[23:16] 111b -- destination is **BREG**[31:24]
RDREG16	00011b	<dest16>	<regaddr>	-	-	Load a word into **AREG** or **BREG** (destination is specified by <dest16>) with value read from {ADDR[31:8], <regaddr>}. <dest16> x0xb -- destination is **AREG**[15:0] x1xb -- destination is **BREG**[15:0]
RDREG32	00100b	<dest32>	<regaddr>	-	-	Load a long word into **AREG** or **BREG** (destination is specified by <dest32>) with value read from {ADDR[31:8], <regaddr>}. <dest32> xx0b -- destination is **AREG**[31:0] xx1b -- destination is **BREG**[31:0]]

WRREG8	00101b	<src8>	<regaddr>	-	-	Write a byte value from **AREG** or **BREG** (source is specified by <src8>) to the location specified by {**ADDR**[31:8], <regaddr>}. <src8> 000b -- source is **AREG**[7:0] 001b -- source is **AREG**[15:8] 010b -- source is **AREG**[23:16] 011b -- source is **AREG**[31:24] 100b -- source is **BREG**[7:0] 101b -- source is **BREG**[15:8] 110b -- source is **BREG**[23:16] 111b -- source is **BREG**[31:24]
WRREG16	00110b	<src16>	<regaddr>	-	-	Write a word value from **AREG** or **BREG** (source is specified by <src16>) to the location specified by {**ADDR**[31:8], <regaddr>}. <src16> x0xb -- source is **AREG**[15:0] x1xb -- source is **BREG**[15:0]
WRREG32	00111b	<src32>	<regaddr>	-	-	Write a long word value from **AREG** or **BREG** (source is specified by <src32>) to the location specified by {**ADDR**[31:8], <regaddr>}. <src32> xx0b -- source is **AREG**[31:0] xx1b -- source is **BREG**[31:0]
WAITEVENT	01000b	<trigsel>	-	-	-	Wait for one of the 4 triggers signals to go high. <trigsel> selects which signal to wait on. The Event Processor will not fetch the next instruction until the trigger goes high. <trigsel> x00b – wait on RTC interrupt x01b – wait on MDC interrupt x10b – wait on HIFIRQ interrupt x11b – wait on PORT (GPIO) interrupt
NIBBMULT	01001b	<nibbsel>	-	-	-	Nibble multiply instruction. **BREG**[31:0] <= <nibble> * **BREG**[15:0] <nibbsel> 000b – <nibble> is **AREG**[3:0] 001b – <nibble> is **AREG**[7:4] 010b -- <nibble> is **AREG**[11:8] 011b -- <nibble> is **AREG**[15:12] 100b -- <nibble> is **AREG**[19:16] 101b -- <nibble> is **AREG**[23:20] 110b -- <nibble> is **AREG**[27:24] 111b -- <nibble> is **AREG**[31:28]
BYTEMULT	01010b	<bytesel>	-	-	-	Byte multiply instruction **BREG**[31:0] <= <byte> * **BREG**[15:0] <bytesel> x00b -- <byte> is **AREG**[7:0] x01b -- <byte> is **AREG**[15:8] x10b -- <byte> is **AREG**[23:16] x11b -- <byte> is **AREG**[31:24]
ADDREGS	01011b	-	-	-	-	**BREG**[31:0] <= **AREG**[31:0] + **BREG**[31:0]
ADDA	01100b	Addval [10:8]	Addval [7:0]	-	-	Add an 11-bit value to **AREG**[31:0]. **AREG**[31:0] <= **AREG**[31:0] + addval[10:0]
BCDCONV	01101b	-	-	-	-	Convert {**AREG**[7:4], **AREG**[3:0]} from BCD to binary

SHIFTRIGHTA	01110b	<shiftsel>	-	-	-	Shift right **AREG**[31:0] by amount determined by <shiftsel>. <shiftsel> x00b – shift right by 4 (divide by16) x01b – shift right by 8 (divide by 256) x10b – shift right by 12 (divide by 4096) x11b – shift right by 16 (divided by 65536)
END	11111b	-	-	-	-	Go back to IDLE state and turn off system clock.

Assembly Language for Event Processor

A C include file “eventprocessor.h” is provided which has macro definitions for a set of assembly language instructions to make Event Processor programming code more human-readable. The following is a snippet from the “eventprocessor.h” file which lists the assembly language instructions available:

```
//***************************************************************************
// EVENT PROCESSOR Assembler Basic Instructions
//***************************************************************************
   // Load address register (ADDR) with upper 24 bits of <Address>
   // 4-byte instruction
#define LDADDR(Address)             EP_LDADDR,((Address)>>8)&0xFF,((Address)>>16)&0xFF,((Address)>>24)&0xFF

   // Read immediate 8-bit <immval> value to <dst8> register
   // 2-byte instruction
   // <dst8> is one of AREG7_0, AREG15_8, AREG23_16, AREG31_24, BREG7_0, BREG15_8, BREG23_16, BREG31_24
#define RDREG8I(dst8,immval)        EP_RDREG8I|(dst8&0x7),immval

   // Read 8-bit value from location specified by {ADDR, <regaddr>} to <dst8> register
   // 2-byte instruction
   // <dst8> is one of AREG7_0, AREG15_8, AREG23_16, AREG31_24, BREG7_0, BREG15_8, BREG23_16, BREG31_24
#define RDREG8(dst8 ,regaddr)       EP_RDREG8 |(dst8 &0x7),regaddr
   // Read 16-bit value from location specified by {ADDR, <regaddr>} to <dst16> register
   // 2-byte instruction
   // <dst16> is one of AREG15_0, AREG31_16, BREG15_0, BREG31_16
#define RDREG16(dst16,regaddr)      EP_RDREG16|(dst16&0x7),regaddr
   // Read 32-bit value from location specified by {ADDR, <regaddr>} to <dst32> register
   // 2-byte instruction
   // <dst32> is one of AREG31_0, BREG31_0
#define RDREG32(dst32,regaddr)      EP_RDREG32|(dst32&0x7),regaddr

   // Write 8-bit value from <src8> register to location specified by {ADDR, <regaddr>}
   // 2-byte instruction
   // <src8> is one of AREG7_0, AREG15_8, AREG23_16, AREG31_24, BREG7_0, BREG15_8, BREG23_16, BREG31_24
#define WRREG8(src8  ,regaddr)      EP_WRREG8 |(src8  &0x7),regaddr
   // Write 16-bit value from <src16> register to location specified by {ADDR, <regaddr>}
   // 2-byte instruction
   // <src16> is one of AREG15_0, AREG31_16, BREG15_0, BREG31_16
#define WRREG16(src16 ,regaddr)     EP_WRREG16|(src16 &0x7),regaddr
   // Write 32-bit value from <src32> register to location specified by {ADDR, <regaddr>}
   // 2-byte instruction
   // <src32> is one of AREG31_0, BREG31_0
#define WRREG32(src32 ,regaddr)     EP_WRREG32|(src32 &0x7),regaddr

   // Wait on event/interrupt source specified by <trigsel>
   // 1-byte instruction
   // <trigsel> is one of EP_RTCINT, EP_MDCINT, EP_ALLINT, EP_GPIOINT
#define WAITEVENT(trigsel)          EP_WAITEVENT|(trigsel&0x7)

   // Nibble multiply by BREG[15:0]:  BREG[31:0] = <nibbsel> * BREG[15:0]
   // 1-byte instruction
   // <nibbsel> is one of AREG3_0, AREG7_4, AREG11_8, AREG15_12, AREG19_16, AREG23_20, AREG27_24, AREG31_28
#define NIBBLEMULT(nibbsel)         EP_NIBBLEMULT|(nibbsel&0x7)

   // Byte multiply by BREG[15:0]:  BREG[31:0] = <bytesel> * BREG[15:0]
   // 1-byte instruction
   // <bytesel> is one of AREG7_0, AREG15_8, AREG23_16, AREG31_24
#define BYTEMULT(bytesel)           EP_BYTEMULT|(bytesel&0x7)

   // Add AREG[31:0] and BREG[31:0]:  BREG[31:0] = AREG[31:0] + BREG[311:0]
   // 1-byte instruction
#define ADDREGS()                   EP_ADDREGS

   // Add 11-bit <imm11bit> value to AREG[31:0]:  AREG[31:0] = AREG[31:0] + <imm11bit>
   // 2-byte instruction
#define ADDA(imm11bit)             EP_ADDA | ((imm11bit>>8)0x7), (imm11bit & 0xFF)

   // Convert {AREG[7:4], AREG[3:0]} from BCD to binary
   // 1-byte instruction
#define BCDCONV()                   EP_BCDCONV

   // Shift right AREG[31:0] by <shiftsel>
   // 1-byte instruction
   // <shiftsel> is one of SHIFTRIGHT4, SHIFTRIGHT8, SHIFTRIGHT12, SHIFTRIGHT16
#define SHIFTRIGHTA(shiftsel)       EP_SHIFTRIGHTA|(shiftsel&0x7)

   // End instruction
   // 1-byte instruction
#define END                         EP_END

//***************************************************************************
// EVENT PROCESSOR Assembler Combined Instructions
//***************************************************************************
   // Load immediate 8-bit <imm8> value to AREG{7:0]
   // 2-byte instruction
#define LDAREG8(imm8)              RDREG8I( AREG7_0  , ( imm8>>0  ) & 0xFF )

   // Load immediate 16-bit <imm16> value to AREG{15:0]
   // 4-byte instruction
#define LDAREG16(imm16)            RDREG8I( AREG7_0  , ( imm16>>0  ) & 0xFF ),¥
                                   RDREG8I( AREG15_8 , ( imm16>>8  ) & 0xFF )

   // Load immediate 32-bit <imm32> value to AREG{31:0]
   // 8-byte instruction
#define LDAREG32(imm32)            RDREG8I( AREG7_0  , ( imm32>>0  ) & 0xFF ),¥
                                   RDREG8I( AREG15_8 , ( imm32>>8  ) & 0xFF ),¥
                                   RDREG8I( AREG23_16, ( imm32>>16 ) & 0xFF ),¥
```

```
                                RDREG8I( AREG31_24, ( imm32>>24 ) & 0xFF )

   // Load immediate 8-bit <imm8> value to BREG{7:0]
   // 2-byte instruction
#define LDBREG8(imm8)             RDREG8I( BREG7_0  , ( imm8>>0  ) & 0xFF )

   // Load immediate 16-bit <imm16> value to BREG{15:0]
   // 4-byte instruction
#define LDBREG16(imm16)           RDREG8I( BREG7_0  , ( imm16>>0  ) & 0xFF ),¥
                                  RDREG8I( BREG15_8 , ( imm16>>8  ) & 0xFF )

   // Load immediate 32-bit <imm32> value to BREG{31:0]
   // 8-byte instruction
#define LDBREG32(imm32)           RDREG8I( BREG7_0  , ( imm32>>0  ) & 0xFF ),¥
                                  RDREG8I( BREG15_8 , ( imm32>>8  ) & 0xFF ),¥
                                  RDREG8I( BREG23_16, ( imm32>>16 ) & 0xFF ),¥
                                  RDREG8I( BREG31_24, ( imm32>>24 ) & 0xFF )


   // Direct write 8-bit value:  Write immediate 8-bit <imm8> value to location specified by <address>
   // NOTE:  AREG[7:0] is used as the holding register for the data
   // 8-byte instruction
#define VALWR8D(address, imm8)       LDAREG8(imm8), LDADDR(address), WRREG8(AREG7_0, (address)&0xFF)

   // Direct write 16-bit value:  Write immediate 16-bit <imm16> value to location specified by <address>
   // NOTE:  AREG[15:0] is used as the holding register for the data
   // 10-byte instruction
#define VALWR16D(address, imm16)     LDAREG16(imm16), LDADDR(address), WRREG16(AREG15_0, (address)&0xFF)

   // Direct write 32-bit value:  Write immediate 32-bit <imm32> value to location specified by <address>
   // NOTE:  AREG[31:0] is used as the holding register for the data
   // 14-byte instruction
#define VALWR32D(address, imm32)     LDAREG32(imm32), LDADDR(address), WRREG32(AREG31_0, (address)&0xFF)

   // Direct read 8-bit value:  Read 8-bit value from location specified by <address> to <dst8> register
   // 6-byte instruction
   // <dst8> is one of AREG7_0, AREG15_8, AREG23_16, AREG31_24, BREG7_0, BREG15_8, BREG23_16, BREG31_24
#define REGRD8D(dst8, address)        LDADDR(address), RDREG8(dst8, (address)&0xFF)

   // Direct read 16-bit value:  Read 16-bit value from location specified by <address> to <dst16> register
   // 6-byte instruction
   // <dst16> is one of AREG15_0, AREG31_16, BREG15_0, BREG31_16
#define REGRD16D(dst16, address)        LDADDR(address), RDREG16(dst16, (address)&0xFF)

   // Direct read 32-bit value:  Read 32-bit value from location specified by <address> to <dst32> register
   // 6-byte instruction
   // <dst32> is one of AREG31_0, BREG31_0
#define REGRD32D(dst32, address)        LDADDR(address), RDREG32(dst32, (address)&0xFF)

   // Direct write 8-bit value:  Write 8-bit value from <src8> register to location specified by <address>
   // 6-byte instruction
#define REGWR8D(src8, address)       LDADDR(address), WRREG8(src8, (address)&0xFF)

   // Direct write 16-bit value:  Write 16-bit value from <src16> register to location specified by <address>
   // 6-byte instruction
#define REGWR16D(src16, address)      LDADDR(address), WRREG16(src16, (address)&0xFF)

   // Direct write 32-bit value:  Write 32-bit value from <src32> register to location specified by <address>
   // 6-byte instruction
#define REGWR32D(src32, address)      LDADDR(address), WRREG32(src32, (address)&0xFF)


//*****************************************************************************
// EVENT PROCESSOR Assembler B Register Indirect Read/Write Instructions
// (to be expanded/generated by assembler)
//*****************************************************************************
   // Indirect 8-bit read:  Read 8-bit value from location specified by BREG[31:0] into <dst8> register
   // 30-byte instruction
   // <dst8> is one of AREG7_0, AREG15_8, AREG23_16, AREG31_24
#define REGRD8BI(dst8)       // dummy definition, <dst8> is AREG7_0, AREG15_8, AREG23_16, or AREG31_24

   // Indirect 16-bit read:  Read 16-bit value from location specified by BREG[31:0] into <dst16> register
   // 30-byte instruction
   // <dst16> is one of AREG15_0, AREG31_16
#define REGRD16BI(dst16)     // dummy definition, <dst16> is AREG15_0 or AREG31_16

   // Indirect 32-bit read:  Read 32-bit value from location specified by BREG[31:0] into AREG[31:0]
   // 30-byte instruction
#define REGRD32BI()          // dummy definition

   // Indirect 8-bit write:  Write 8-bit value from <src8> register to location specified by BREG[31:0]
   // 30-byte instruction
#define REGWR8BI(src8)       // dummy definition, <src8> is AREG7_0, AREG15_8, AREG23_16, or AREG31_24

   // Indirect 16-bit write:  Write 16-bit value from <src16> register to location specified by BREG[31:0]
   // 30-byte instruction
#define REGWR16BI(src16)     // dummy definition, <src16> is AREG15_0 or AREG31_16

   // Indirect 32-bit write:  Write 32-bit value from AREG[31:0] to location specified by BREG[31:0]
   // 30-byte instruction
#define REGWR32BI()          // dummy definition
```

Assembly Programming for Event Processor in C Language

The following is a recommended method for programming and running Event Processor code in the C language:

1. Add a '`#include "eventprocessor.h`"' statement in the C application source file.

2. For each Event Processor program, declare a constant byte array and initialize it with Event Processor assembly instructions as shown in the following example:

```
const uint8_t myEPprog[] = {
    LDAREG(0xAA),                  // Load 0xAA to AREG[7:0]
    REGWR8D(AREG7_0, 0x40001234),  // Write AREG[7:0] to location 0x40001234
    END
};
```

3. Copy the byte array contents of the Event Processor program from the Host MCU flash/ROM memory to an off-screen RAM location in the S1D13C00 and program the MDCEPBASEADDR[31:0] register with the address of the RAM location.
4. Program the MDCEPCTRL.TRIGSEL[1:0] bits to select the interrupt trigger signal to use.
5. Set MDCEPCTRL.EPEN bit 1 to enable the Event Processor trigger.

Assembler for Event Processor

An assembler software (EPas.exe) is available which enables writing assembly language code in a file with assembly instructions that are terminated by ';'. The assembler accepts an input file with .ep suffix and generates a C include file with .h suffix which contains the constant byte array declaration and initialization with assembly language instructions. The .h file can then be included in the C application program for downloading and executing the Event Processor code in the S1D13C00.

The following shows an example assembler input file called "sampleprog.ep" and the generated "sampleprog.h" file:

Source "sampleprog.ep" file

```
//------------------------------
// ORIGIN 200100F2
   LDAREG16(0x1234);                    // Write 0x1234 to location 0x20005678
   REGWR16D(AREG15_0, 0x20005678);
      // Calculate location of 3-character weekday string
   REGRD8D(AREG7_0, RTC_WKYR+1); // Read weekday value to AREG[7:0]
   LDBREG16(3);                  // BREG[15:0] = 3
   BYTEMULT(AREG7_0);            // BREG[31:0] = 3*weekday
   LDAREG32(RAM_BASE+0x16000);   // AREG[31:0] = Base of weekday strings table
   ADDREGS();                    // BREG[31:0] = BREG[31:0] + AREG[31:0]

   REGRD32BI();                  // Special "BREG indirect" read instruction
                                 // Read 3-character string to AREG[23:0]
   END;
```

Generated "sampleprog.h" file after "EPas.exe sampleprog.ep" is executed:

```
const uint8_t sampleprog[] = {
//------------------------------
// ORIGIN 200100F2
   LDAREG16(0x1234),                    // Write 0x1234 to location 0x20005678
   REGWR16D(AREG15_0, 0x20005678),
      // Calculate location of 3-character weekday string
   REGRD8D(AREG7_0, RTC_WKYR+1), // Read weekday value to AREG[7:0]
   LDBREG16(3),                  // BREG[15:0] = 3
   BYTEMULT(AREG7_0),            // BREG[31:0] = 3*weekday
   LDAREG32(RAM_BASE+0x16000),   // AREG[31:0] = Base of weekday strings table
   ADDREGS(),                    // BREG[31:0] = BREG[31:0] + AREG[31:0]

// REGRD32BI(AREG31_0) expansion
LDADDR(0x2001012D),           // Fill in RDREG32(AREG31_0, 0) second parameter
WRREG8(BREG7_0, 0x2D),
LDADDR(0x20010129),           // Fill in LDADDR(0) parameter
WRREG8(BREG15_8, 0x29),
LDADDR(0x2001012A),
WRREG8(BREG23_16, 0x2A),
LDADDR(0x2001012B),
WRREG8(BREG31_24, 0x2B),
LDADDR(0),                /* <- dummy 0 value */
RDREG32(AREG31_0, 0),     /* <- dummy 0 value */
                  // Special "BREG indirect" read instruction
                                 // Read 3-character string to AREG[23:0]
   END
};
// Code Size:  61 bytes
```

21.7 Interrupts

The MDC has a function to generate the interrupts shown in Table 21.36.

Table 21.36 MDC Interrupt Function

Interrupt	Interrupt flag	Set condition	Clear condition
Graphics	MDCINTCTL.GFXIF	When Drawing or Copying engine is finished.	Writing 1
Display Update	MDCINTCTL.UPDIF	When Display Updater is finished updating the panel.	Writing 1
VCNT counter match	MDCINTCTL.VCNTIF	During display update, when the VCNT counter matches the VCNTCOMP value.	Writing 1

The MDC provides interrupt enable bits corresponding to each interrupt flag. An interrupt request is sent to the Host MCU only when the interrupt flag, of which interrupt has been enabled by the interrupt enable bit, is set. Figure 21.54 shows a diagram of the MDCINT interrupt signal. The MDCINT signal is logical ORed with interrupt signals from other peripherals to generate the HIFIRQ interrupt signal to the host MCU.
See Section 22 (“Interrupts”) for more details on the HIFIRQ output.

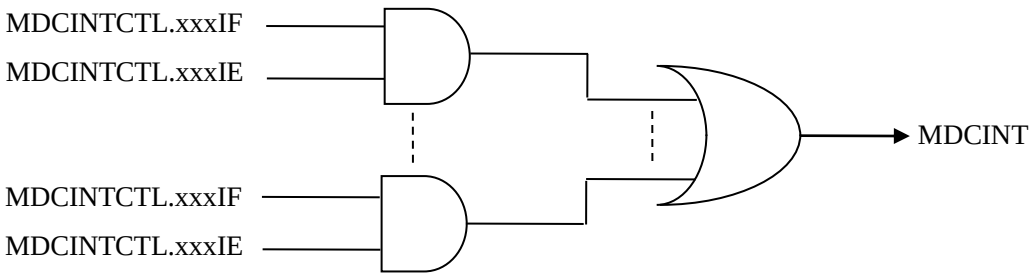


Figure 21.54 MDCINT Interrupt Circuit

22. Interrupts

Each peripheral in the S1D13C00 has an active-high interrupt output signal and the interrupt output signals from all peripherals are ORed together to generate the HIFRQ output to the host MCU. The SYSCTRL.IRQPOL bit is used to configure the polarity of the HIFIRQ output. Figure 22.1 shows the circuit for the HIFIRQ output.

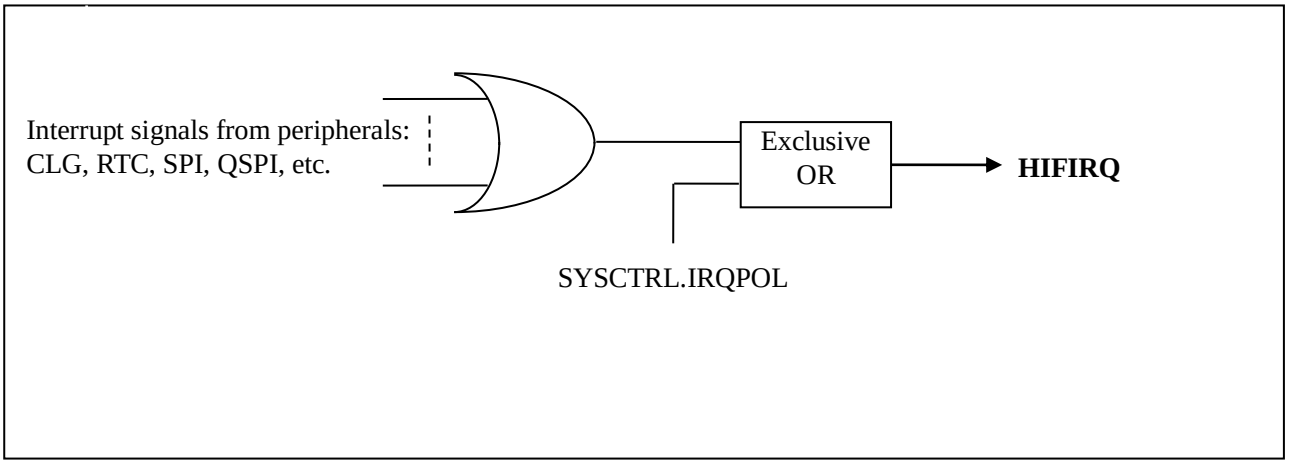
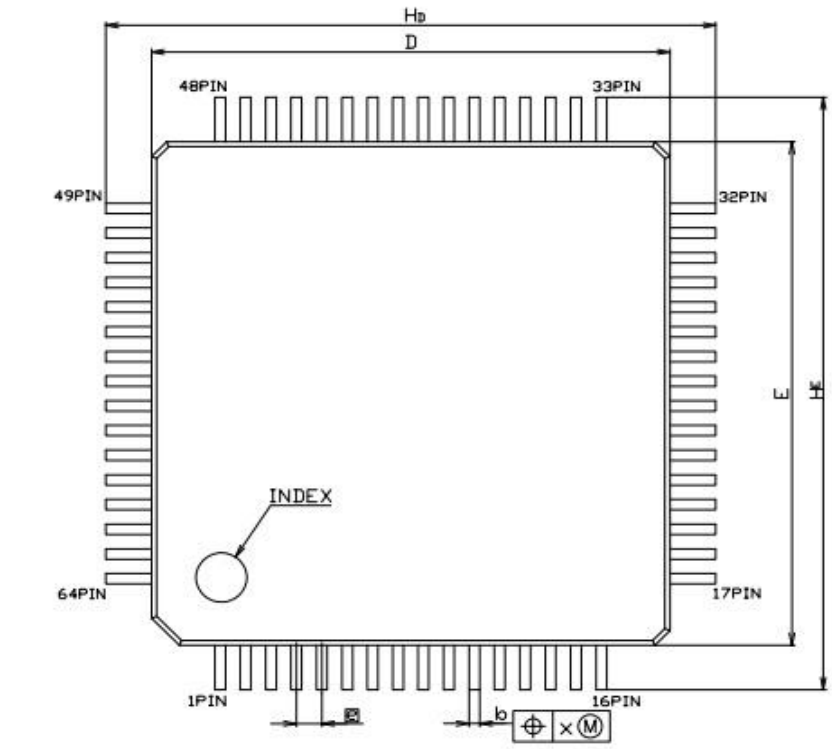


Figure 22.1 HIFIRQ Output Circuit

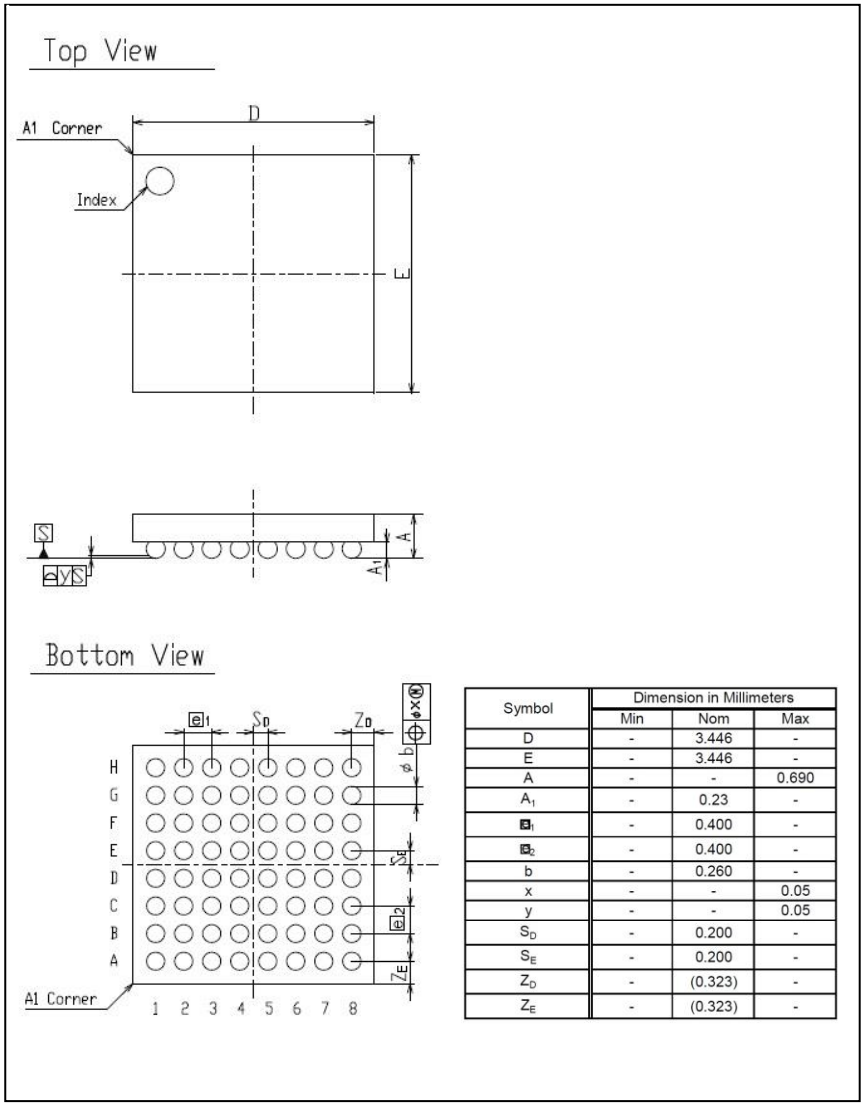
Each peripheral has its own individual interrupt enable control registers.

23. Mechanical Data



Symbol	Dimension in Millimeters		
	Min	Nom	Max
D	-	10.00	-
E	-	10.00	-
A	-	-	1.20
A_1	-	0.10	-
A_2	-	1.00	-
e	-	0.50	-
b	0.17	-	0.27
c	0.09	-	0.20
θ	0	-	10
L	0.30	-	0.75
L_1	-	1.00	-
H_D	-	12.00	-
H_E	-	12.00	-
x	-	-	0.08
y	-	-	0.08

Figure 23.1 S1D13C00 TQFP13-64 Package Diagram



Symbol	Dimension in Millimeters		
	Min	Nom	Max
D	-	3.446	-
E	-	3.446	-
A	-	-	0.690
A_1	-	0.23	-
e_1	-	0.400	-
e_2	-	0.400	-
b	-	0.260	-
x	-	-	0.05
y	-	-	0.05
S_D	-	0.200	-
S_E	-	0.200	-
Z_D	-	(0.323)	-
Z_E	-	(0.323)	-

Figure 23.2 S1D13C00 WCSP-64 Package Diagram

24. Revision History

Rev. No.	Date	Page	Category	Contents
1.2	10/16/2019	20		• Section 4.2.4. Initial state of QSPID0 was corrected to be "O (L)" from "Hi-Z".
1.1	11/01/2019			• Added descriptions for support of EPD (electrophoretic deposition) panels interface.
1.0	04/09/2018			• Initial public release.

EPSON

International Sales Operations

For more information on Epson Display Controllers, visit the Epson Global website.

https://global.epson.com/products_and_drivers/semicon/products/display_controllers/

For Sales and Technical Support, contact the Epson representative for your region.

https://global.epson.com/products_and_drivers/semicon/information/support.html

Document Code: XB8A-A-001-01
First Issue May 2018
Revised October 2019