HS6621Cx

Bluetooth Low Energy Compliant System-on-Chip

V1.5

Datasheet

Table of Contents

Version History

Version	Revision	Date	Author	Reviewer
V1.1	Initial version	2019/12/24	YQ	
V1.2	Update pmu, pmu_hib	2020/01/02	YQ	
V1.3	delete pmu, pmu_hib etc.	2021/08/10	ZJ	
V1.4	Update the I2C content	2021/09/16	ZJ	
V1.5	Delete DMAC configuration register	2021/10/14	ZJ	

1 HS6621Cx Overview

1.1 Description

The HS6621Cx family is a single-mode, low-power Bluetooth 5.1 System-on-Chip (SOC). It can be configured as a Broadcaster, an Observer, a Central, and a Peripheral and supports the combination of all the above roles, making it an ideal choice for Internet of Things (IOT) and smart wearable devices.

Based on ARM® Cortex®-M4F CPU core, the HS6621C integrates Bluetooth 5.1 protocol stacks, a 2.4 GHz RF transceiver, programmable Flash memory, RAM, and multiple peripherals.

HS6621Cx series includes HS6621CG, HS6621CQ, and HS6621CM.

Parameter	HS6621CG	HS6621CQ	HS6621CM
CPU	Cortex-M4F	Cortex-M4F	Cortex-M4F
RAM	64KB	64KB	64KB
Flash	8Mb	4Mb	4Mb
Package	QFN48	QFN32	QFN32
Temperature range	-40~85℃	-40~85℃	-40~85℃

Table 1.1 HS6621Cx series

1.2 Features

- RF transceiver
  - -95dBm sensitivity @1Mbps
  - -93dBm sensitivity @2Mbps
  - -96dBm sensitivity @500Kbps
  - -100dBm sensitivity @125Kbps
  - TX power -20~7dBm
  - RSSI (1db resolution)
- CPU
  - ARM® Cortex™-M4F, max 64MHz
  - Serial Wire Debug (SWD)
- Memory
  - 64KB SRAM
  - 48KB ROM
  - SFLASH
    - HS6621CG: 8Mb
    - HS6621CQ: 4Mb
    - HS6621CM: 4Mb

- Clocks
  - HS6621CG: 32MHz crystal, 32MHz RC, 32.768KHz crystal, 32.768KHz RC
  - HS6621CQ: 32MHz crystal, 32MHz RC, 32.768KHz RC
  - HS6621CM: 32MHz crystal, 32MHz RC, 32.768KHz crystal, 32.768KHz RC
- Link Controller
  - BT 5.1 LE PHY, link controller
  - Proprietary 2.4-GHz link controller
- Power Management
  - Supply voltage range 1.8V~3.6V
  - 5.3mA peak current in RX
  - 5.2mA peak current in TX (0dBm)
  - 0.4uA in Hibernation mode without RTC clock
  - 0.8uA in Hibernation mode with RTC clock
  - 2.5uA in deep sleep mode
  - Integrated DCDC BUCK Converter
  - Integrated Charger
- Software
  - Full compliant with BLE version 5.1, complete power-optimized stack, including controller and host
  - Supports BT mesh network
  - Support BLE sample applications and profiles
  - Support HCI controller interface
  - Support 6 Low PAN
  - Support OTA
- Peripherals
  - 8 channels DMA
  - UART x 2
  - I2S interface
  - Flexible General Purpose IOs
    - HS6621CG: 32 general-purpose I/O GPIOs (max)
    - HS6621CM: 17 general-purpose I/O GPIOs (max)
    - HS6621CQ: 21 general-purpose I/O GPIOs (max)
  - I2C (master or slave) x 3
  - SPI (master or slave) x 2
  - QSPI LCD/AMOLED controller
  - QSPI Flash controller
  - Watchdog to prevent system dead lock
  - RTC
  - Timers(32bit) x 3
  - PWM 3 x 4
  - Keyboard controller
  - Three way QDEC
  - Flexible GP-ADC
    - HS6621CG/HS6621CQ:8 single-end or differential-end 12bits GP-ADC

- HS6621CM:6 single-end or differential-end 12bits GP-ADC
- Audio ADC
    - HS6621CG: Audio 16-bit ADC，SNR 93dB
    - HS6621CM: Audio 16-bit ADC，SNR 93dB
- Touch sensor controller
- Fully programmable pin assignment
- AES128 HW encryption
- HW Random Number Generator
- Digital signal processing (DSP) instructions

- Package
- HS6621CG: 48-pin 6x6mm QFN48
- HS6621CM: 32-pin 4x4mm QFN32
- HS6621CQ: 32-pin 4x4mm QFN32

1.3 Block Diagram

The following figure shows the block diagram of HS6621Cx.

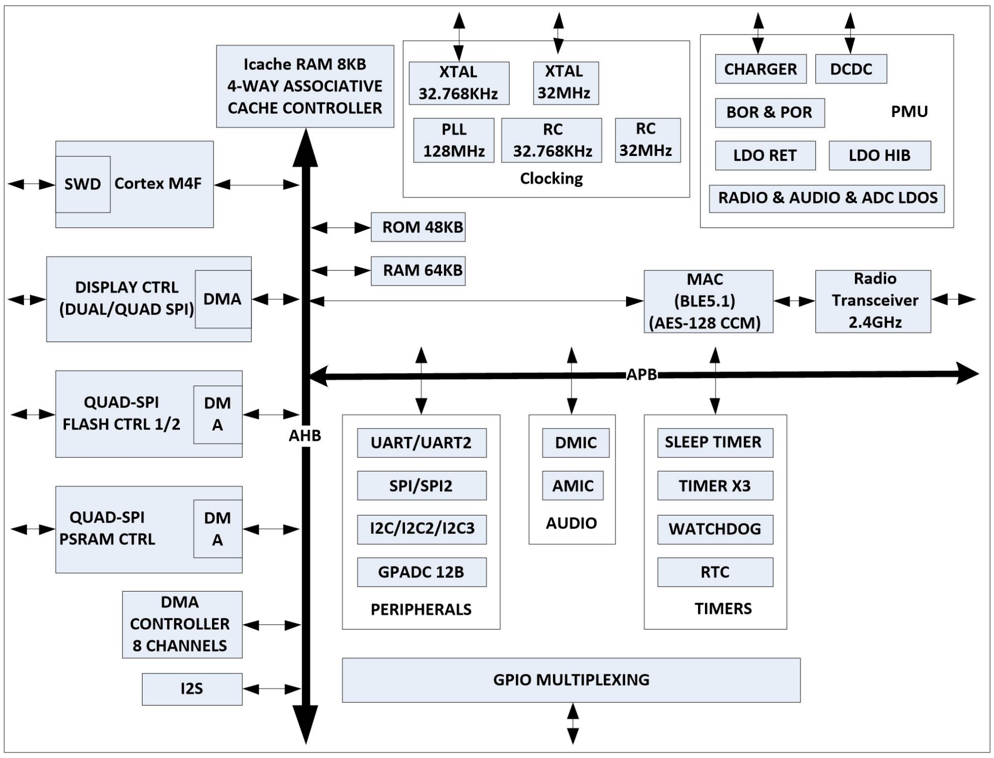


Figure 1.1 HS6621Cx block diagram

Bluetooth Subsystem

A 2.4GHz radio transceiver and a digital communication core (MAC) that supports Bluetooth LE 5.1.

MCU Subsystem

An ARM® Cortex®-M4F with all the required memories and peripherals.

Power Management Unit (PMU) Subsystem

All the required power management modules to supply sufficient power to both internal modules and external peripherals.

Modules required for ultra-low-power operation in standby mode. Sleep clock (XTAL32.768KHz/RC 32.768KHz), wakeup GPIOs (Wake up), sleep timer, and power state controller (Power Sequencer) are used to control the state of different modules.

1.4 Applications

HS6621Cx can be used for different applications. Examples of these applications include:

- Wearables
  - Bluetooth LE health sensor applications
  - Bluetooth LE sports sensor applications
- Bluetooth HID devices
  - Voice remote control
  - Bluetooth LE keyboard/mouse
  - Bluetooth LE game-pad
- Smart home and industrial applications
  - Smart lock
  - Lighting and Smart home
  - E-shelf label Beacon
  - Tire pressure monitoring system (TPMS)
  - Mesh applications

2 Pinout

2.1 QFN48

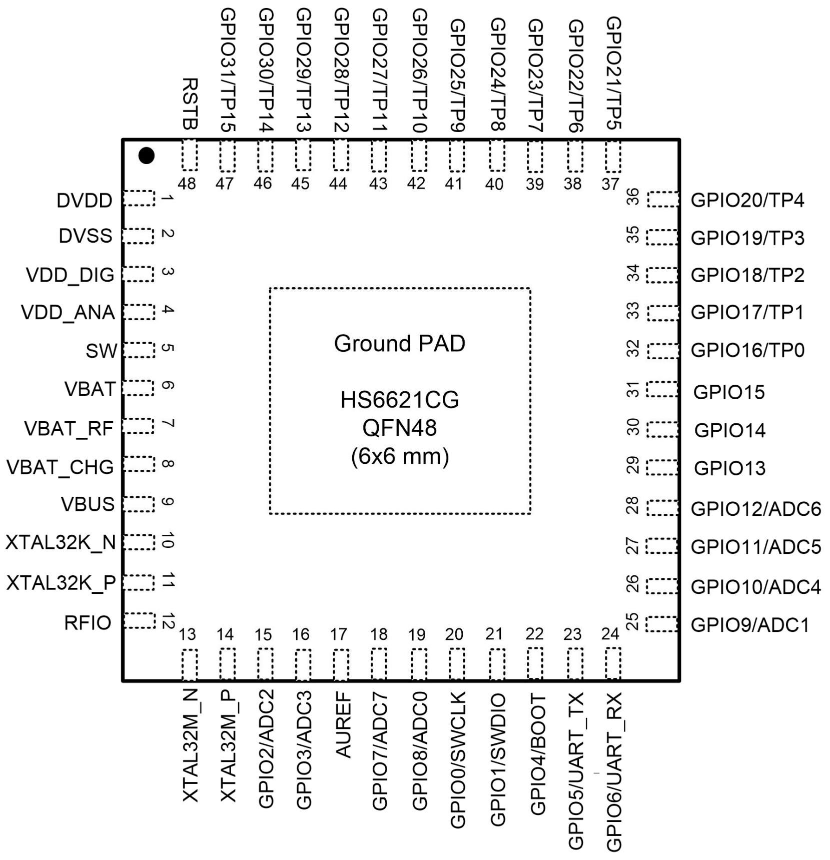


Figure 2.1 HS6621CG chip pin definition

2.2 QFN32

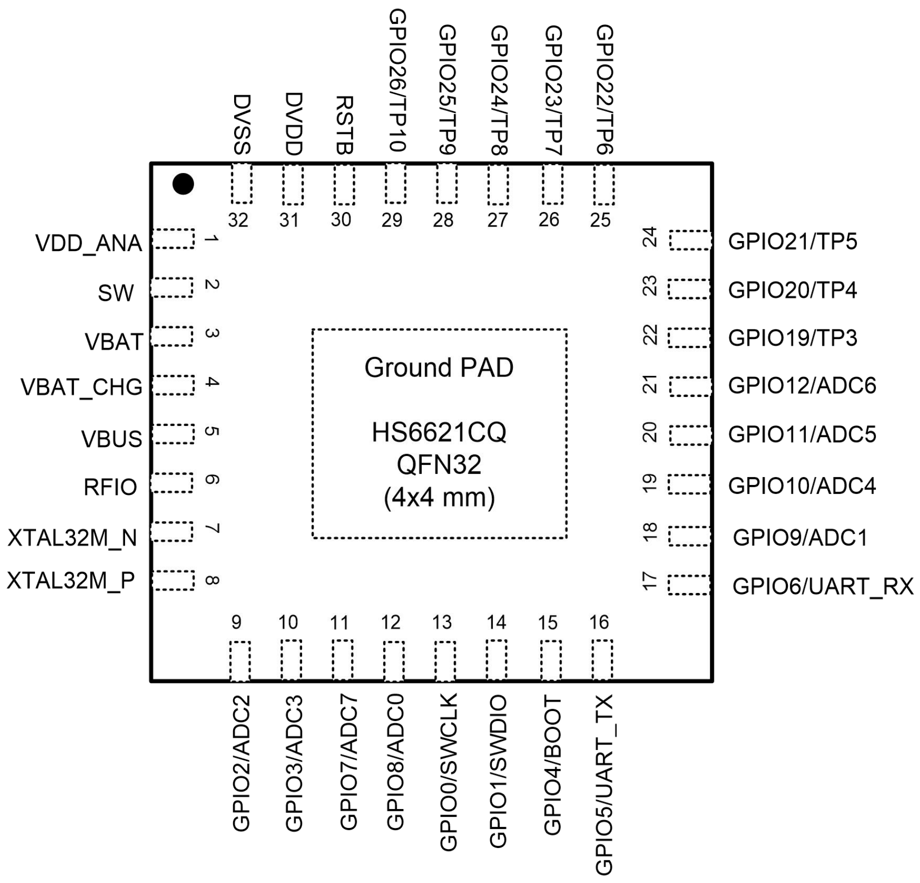


Figure 2.2 HS6621CQ chip pin definition

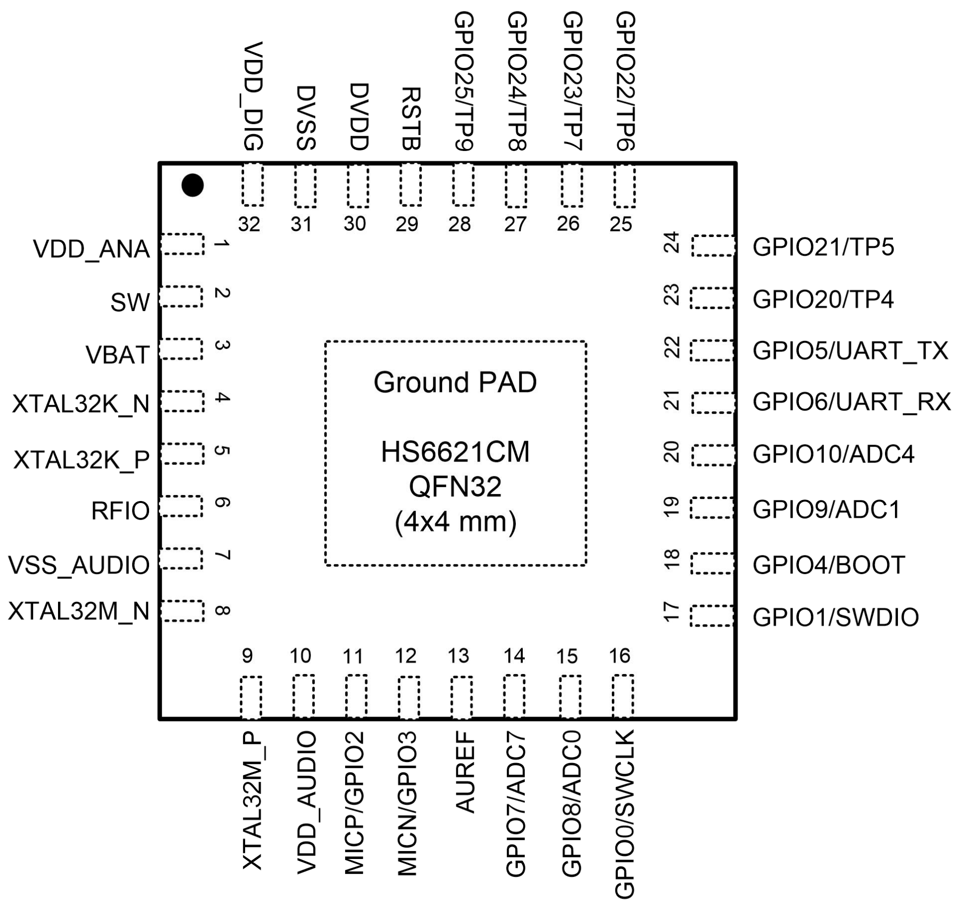


Figure 2.3 HS6621CM chip pin definition

3 MCU Subsystem

HS6621Cx has an MCU subsystem that contains an ARM Cortex-M4F processor, its corresponding buses and peripherals, including all the multiplexing options for the GPIOs, as illustrated in the following figure.

The processor has a floating-point unit and a 32-bit instruction set with Thumb-2 mode support to use a hybrid of 16-bit and 32-bit instructions to maximize the code performance and density.

MCU memories have a special retention voltage and its control to have the memories in different modes according to the application usage:

- OFF
- ON
- Retention

The following are the supported options for the Cortex-M4F:

- NVIC with 52 vectors
- System Tick Timer (SysTick)
- Floating-point unit (FPU)
- Flash Patch and Breakpoint Unit (FPB)

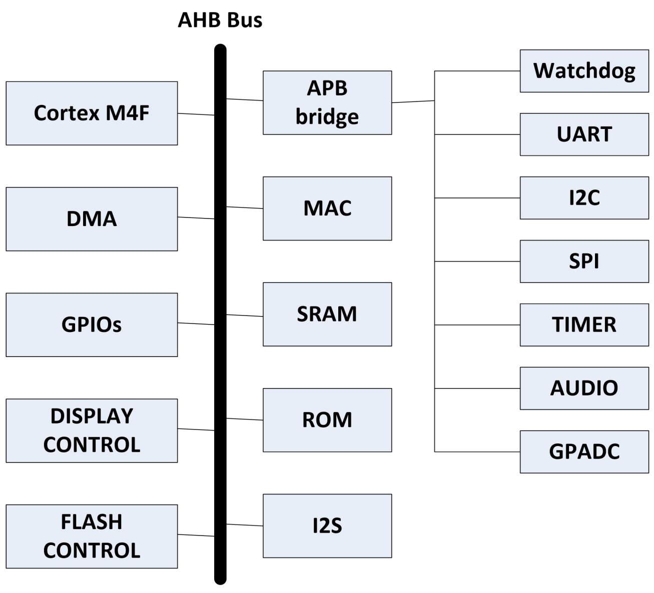


Figure 3.1 Microcontroller Subsystem

3.1 MCU Debug

Serial Wire Debug (SWD) is used for debug.

3.2 Interrupts Vector

The following table shows the NVIC interrupt vector table for HS6621Cx.

Number	Acronym	Description	Address
-	-	-	0x00000000
-15	Reset	Reset	0x00000004
-14	NMI	Non-maskable interrupt.	0x00000008
-13	HardFault	All class of fault	0x0000000c
-12	MemManage	Memory Management fault	0x00000010
-11	BusFault	Error response received from the bus system.	0x00000014
-10	UsageFault	Usage fault	0x00000018
-5	SVCall	System service call via SWI instruction	0x0000002c
-4	Debug	Debug monitor	0x00000030
-2	PendSV	Pendable request for System Service	0x00000038
-1	SysTick	System tick timer	0x0000003c
0	BT BB combo	BLE event	0x00000040
1	rw_native_int	BLE Timer	0x00000044
2	DMA combo	DMA interrupt	0x00000048
3	GPIO combo		
4	Timer combo		
5	KPP depress		
6	KPP release		
7	PMU_timer	System timer	0x0000005c
8	UART0 combo	UART0 global interrupt	0x00000060
9	UART1 combo	UART1 global interrupt	0x00000064
10	i2c1_int	I2C0 global interrupt	0x00000068
11	pin_wakeup_int	GPIO wakeup interrupt	0x0000006c
12	ADC	GPADC interrupt	0x00000070
13	SPI master 0 combo	SPI0 interrupt	0x00000074
14	SPI master 1 combo	SPI1 interrupt	0x00000078
15	gp_comp	analog comparator interrupt	
20	rtc_af_int	RTC alarm interrupt	0x00000090
21	rtc_1hz_int	RTC 1Hz interrupt	0x00000094
22	rtc_ble_int		
23	vtrack_int		
24	cry32m_dig_ready	XTAL 32M ready interrupt	0x000000a0
25	caps_intr		
26	GPIO	GPIO global interrupt	0x000000a8
27		FLASH control 1 global interrupt	0x000000ac

28-30	Timer0	TIMER0 global interrupt	0x000000b0
29	Timer1	TIMER1 global interrupt	0x000000b4
30	Timer2	TIMER2 global interrupt	0x000000b8
31	cpm_wakeup		
32	cry_32k_rdy	XTAL 32K ready interrupt	0x000000c0
33	SFLASH combo	FLASH control 0 global interrupt	0x000000c4
34	QDEC	QDEC global interrupt	0x000000c8
35	power_down_req		
36	i2s_tx_int	I2S tx global interrupt	0x000000d0
37	I2s_rx_int	I2S rx global interrupt	0x000000d4
38	6200_rf_irq	Proprietary 2.4G protocol interrupt	0x000000d8
39	6200_rf_spi_irq	Proprietary 2.4G protocol spi control interrupt	0x000000dc
40-47	soft_int		
48	audio_int	Audio global interrupt	0x00000100
49	i2c2_int	I2C1 global interrupt	0x00000104
50	i2c3_int	I2C2 global interrupt	0x00000108
51	Lcd_spi_int	Display control global interrupt	0x0000010c
52	cc_intr		

Table 3.1 the NVIC interrupt vector

3.3 Electrical Specifications

3.3.1 Absolute Maximum Ratings

HS6621CG/HS6621CM/HS6621CQ could be damaged by extra stress in excess of the absolute maximum ratings working conditions, please be sure the design follow this rule.

Rating		**Min**	**Max**	**Unit**
Storage Temperature		-40	120	ºC
ESD	Human Body Mode	±4000	-	V
	Machine Mode	±200	-	V
	Charge Device Mode	±1000	-	V

Table 3.2 HS6621CG/HS6621CM/HS6621CQ absolute maximum ratings

3.3.2 Ratings Recommend Operating Conditions

Rating	**Min**	**Typ**	**Max**	**Unit**
Operation Temperature	-40	-	85	ºC
Digital Core supply voltage (DVDD)	0.9	1.0	1.2	V
Internal analog LDO power supply (VDD_ANA)	1.1	1.25	1.4	V

Internal digital LDO power supply (VDD_DIG)	1.1	1.25	1.4	V
I/O voltage	Supply voltage	Supply voltage	Supply voltage	V
Supply voltage (VBAT and VBAT_RF)	1.8	3.3	3.6	V

Table 3.3 HS6621CG/HS6621CM recommend operating conditions

3.4 Module Address Mapping

Base Address	**Module**
0x41100000	DMA
0x50000000	SFLASH1
0x41200000	GPIO0
0x41300000	BB
0x41400000	I2S_TX
0x41500000	I2S_RX
0x60000000	SFLASH2
0x40000000	SYS_REG
0x40001000	CPM
0x40002000	QDEC1
0x40002100	QDEC2
0x40002200	QDEC3
0x40004000	RNG
0x40008000	ROM_PATCH
0x4000a000	AUDIO
0x4000b000	2.4G
0x4000c000	KPP
0x40020000	PHY
Base Address	**Module**
0x40030000	UART0
0x40040000	UART1
0x40050000	SPI0
0x40060000	SPI1
0x40070000	I2C1
0x40080000	PMU_HIB_SPI
0x400a0000	DA_IF
0x400b0000	I2C2
0x400c0000	TIMER
0x400d0000	I2C3
0x400e0000	PMU
0x400f0000	RTC

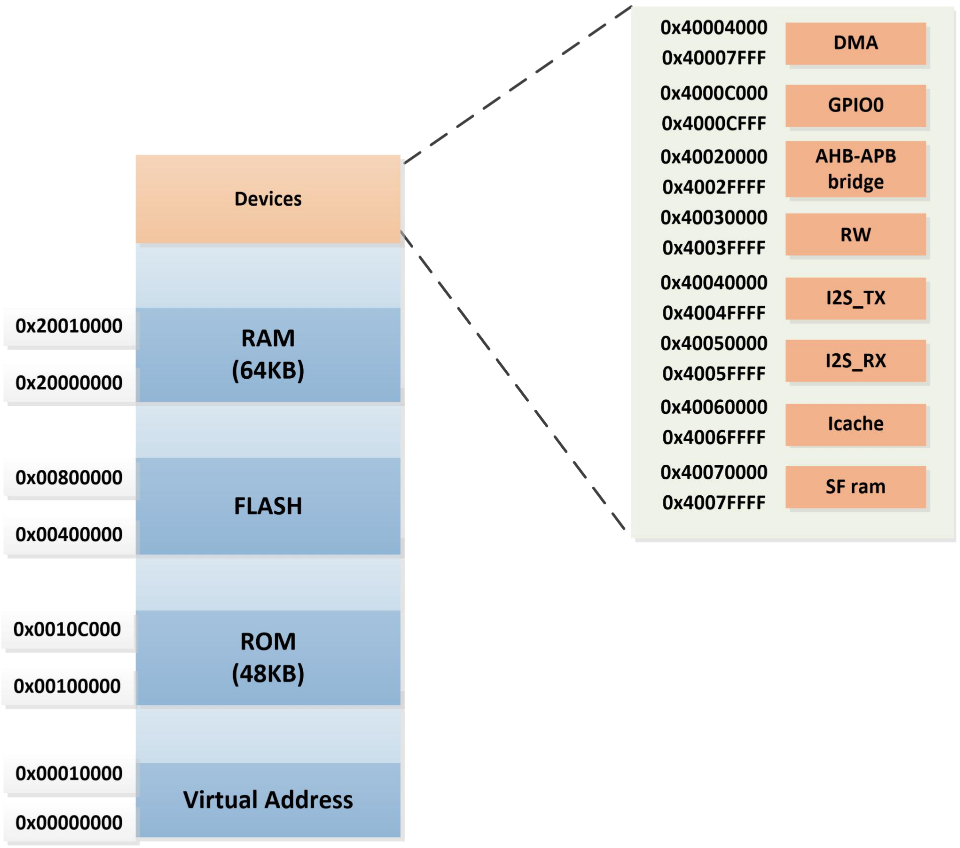
Table 3.4 Module Address Mapping

4 Memory

4.1 Memory Introduction

HS6621Cx SOC memory includes ROM, SRAM and stacked flash for code and data storage. The CPU and peripheral devices can access the memory. The CPU can access the peripherals as well. The address mapping of the memories and devices are explained in the following sections.

4.2 Memory Map



Devices
0x20010000
0x20000000
RAM (64KB)
0x00800000
0x00400000
FLASH
0x0010C000
0x00100000
ROM (48KB)
0x00010000
0x00000000
Virtual Address
0x40004000 0x40007FFF DMA
0x4000C000 0x4000CFFF GPIO0
0x40020000 0x4002FFFF AHB-APB bridge
0x40030000 0x4003FFFF RW
0x40040000 0x4004FFFF I2S_TX
0x40050000 0x4005FFFF I2S_RX
0x40060000 0x4006FFFF Icache
0x40070000 0x4007FFFF SF ram

Figure 4.1 Memery Map

4.3 APB Address Space

The following figure shows the details of Advanced Peripheral Bus (APB) portion of the memory map.

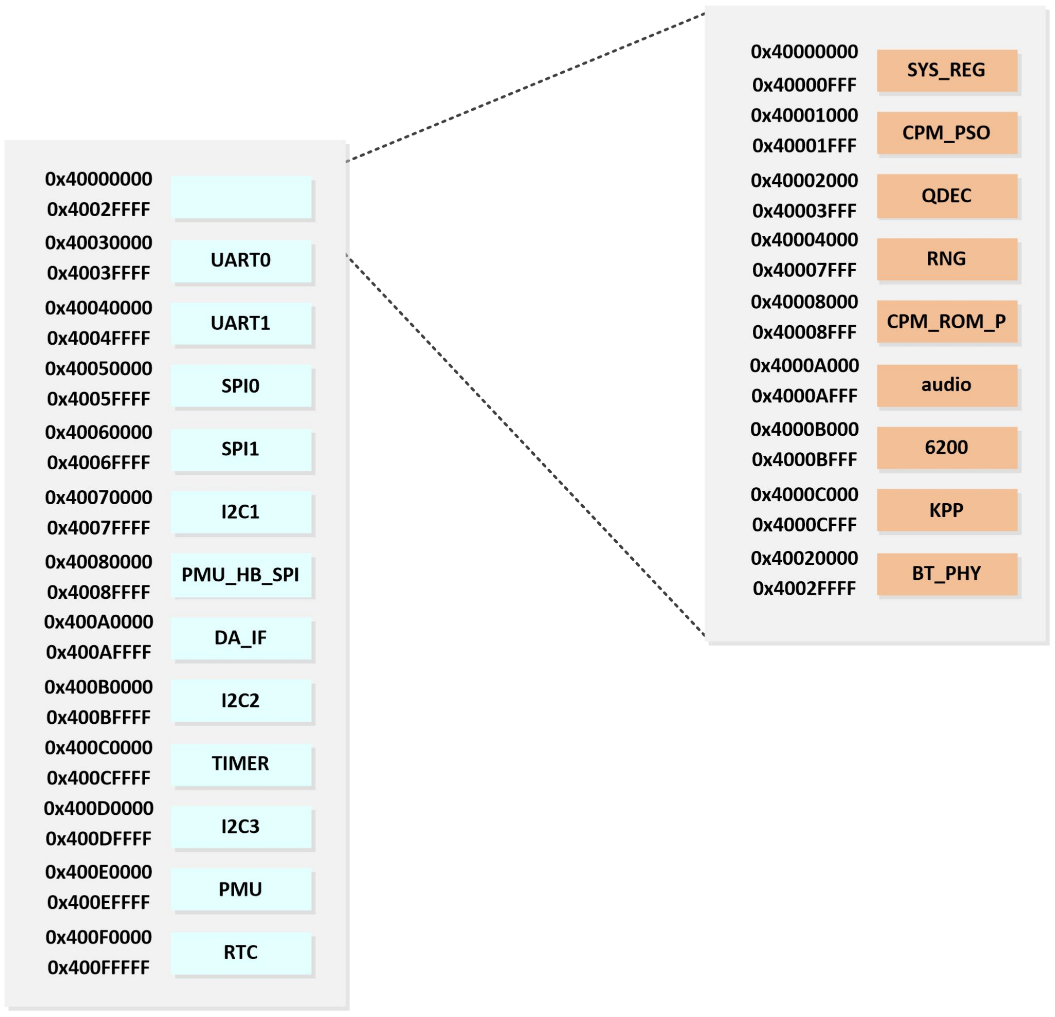


Figure 4.2 APB memory map

5 PMU

5.1 Power Management

Power management Unit(PMU) provide variable voltages and current bias for different blocks of HS6621C, and realize the power on/off function.HS6621C PMU provide several working mode, it mainly includes TX mode, RX mode, Active mode, sleep mode and hibernation mode.

For active mode and RX/TX mode, a DC-DC converter generate the voltage for transceiver modules and digital module efficiently, LDO_dig is used for generating a 1V-default voltage for digital module. Only charger and Wake-up module is always on at any time. HS6621C provide power-on key for power-on/off, and it can be shielded by tying PWR_EN pin to ground. Sleep mode only need turn on LDO_IO, RET_BG, LDO_RET, POR, it can be waked up by GPIO event, BOR and Low power Comparator is an option. If 32K XTAL or RC32K module is kept at sleep mode, the chip can be waked up by timer. Hibernation mode is a much more power-saving mode, only some necessary digital module is kept to waiting for chip activation, and there are partial GPIOs ready for work, 32K clock is also can be maintained at hibernation mode for timed wake up.

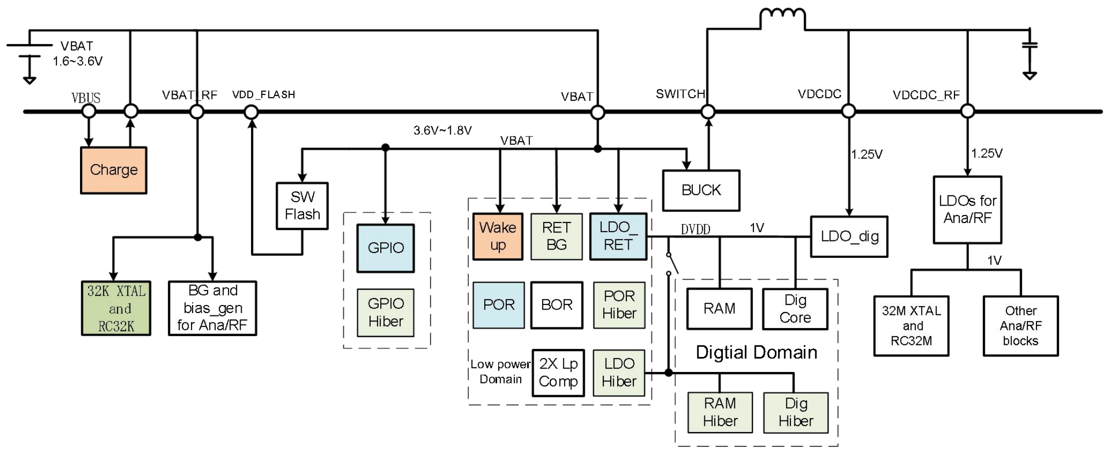


Figure 5.1 Power management Unit architecture

5.2 DC-DC Converter

The DC-DC buck converter provides efficient power supply to all the RF blocks, 32M clock block and digital blocks.

Parameter	Symbol	Min	Typ	Max	Unit	Comment
Output Voltage	Vdcdc	1.1	1.25	1.45	V	Programmable
Load Current	I_load	-	-	50	mA	-
Startup Time	Tstartup	-	-	300	us	-
Voltage Ripples[1]	Vripp	-	10	-	mV	-

Efficiency[2]	Eff	-	85	-	%	-

Table 5.1 DC-DC converter Specifications

Note1:Using 10uH inductor and 4.7uF Capacitor

Note2:Using 10uH inductor and 4.7uF Capacitor, loading current is 10mA.

5.3 Digital LDO

LDO_dig, LDO_RET and LDO_Hiber contains the digital LDO system. The LDO_dig is the main LDO, can provide up to 50mA current output. LDO_RET and LDO_Hiber consume sub-uW power, used to provide voltage of digital blocks at sleep mode or hibernation mode.

Parameter	Symbol	Min	Typ	Max	Unit	Comment
Output Voltage(LDO_dig)	Vdig	0.65	1	1.4	V	Programmable
Output Voltage(LDO_RET)	Vret	0.65	0.85	1	V	Programmable
External loading cap	Cload	0.1	1	10	uF	-

Table 5.2 Digital LDO specifications

5.4 POR/BOD

Power-on Reset (POR) circuit holds the system at reset while the digital voltage (Vdig) reaches the required voltage level. Brown-out detector (BOD) circuit puts the system into reset state when the supply falls below the Brown-out Threshold.

Parameter	Symbol	Min	Typ	Max	Unit	Comment
POR reset time	-	-	8.8	-	ms	-
BOR reset time	Trst	-	8.8	-	ms	-
Brown out threshold	TH_BOD	-	1.4	-	V	-

Table 5.3 POR/BOR specifications

5.5 Wakeup Comparator

HS6621C contains two low power comparators for wakeup. Each one can be turn on/off independently. Comp1 compares two voltages from GPIOs (programmable from GPIO0~8), Comp2 compares voltage from GPIO (programmable from GPIO0~8) and internal voltage reference (3-levels programmable).

Parameter	Symbol	Min	Typ	Max	Unit	Comment
Supply voltage	Vddio	1.8	3	3.3	V	-
Input mismatch(3-sigma)	Vos	-20	-	20	mV	-
VBAT based reference	Vref_VBAT	25	50	75	%	Ratio of VBAT

Table 5.4 Wakeup comparator specifications

6 Always-on Domain(AON)

6.1 WDT

6.1.1 Introduction

The window watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the down counter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit down counter value (in the control register) is refreshed before the down counter has reached the window register value. This implies that the counter must be refreshed in a limited window.

6.1.2 Main Features

- Programmable free-running down counter
- Conditional reset
    - Reset (if watchdog activated) when the down counter value becomes less than 0x40.
    - Reset (if watchdog activated) if the down counter is reloaded outside the window.
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

6.1.3 Function Description

6.1.3.1Block Diagram

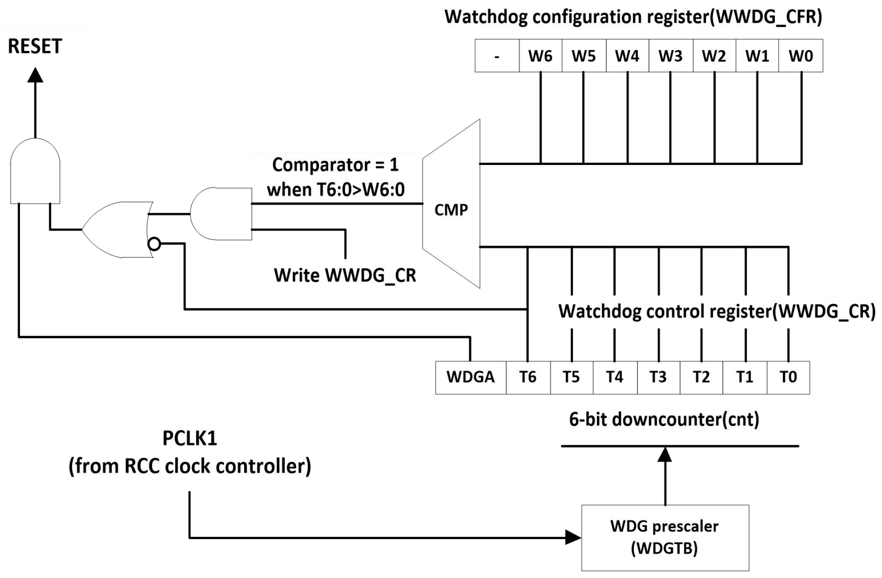


Figure 6.1 WDT Diagram

6.1.3.2Clock

【缺省】

6.1.3.3Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

6.1.3.4Alarm

【缺省】

6.1.4 WDT Register Map

Offset	Name	Description
0x0000	CR	WDT control register
0x0004	TORR	
0x0008	CCVR	second alarm register
0x000c	CRR	regulator register

0x0010	STAT	second counter
0x0014	EOI	watch dog control
0x0018	CLOCK_EN	watch dog trigger value

【寄存器详细说明缺省】

7 Clock

The following figure shows the clocking scheme of HS6621Cx. HS6621Cx clocks are generated using a crystal oscillator (XO) that requires a 32 MHz external crystal. The external crystal is needed to meet the offset requirements of the BLE specifications.

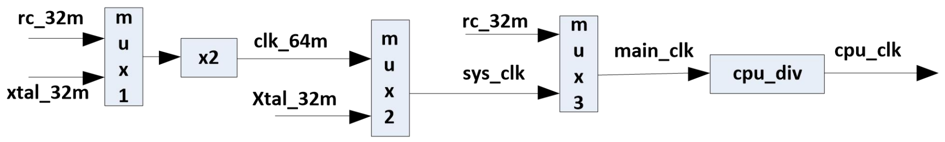


Figure 7.1 HS6621Cx clocking scheme

7.1 XO

XO clock is a crystal-based oscillator that connects to a 32-MHz external crystal. The startup time of the XO clock depends on the crystal type as well as the internal bias current. The bias current is programmable to reduce the current consumption when the XO clock settles.

Parameter	Symbol	Min	Typ	Max	Unit	Comment
XO Frequency					MHz	
Accuracy					ppm	

Table 7.1 XO clock specification

7.2 Ring Oscillator

The ring oscillator clock is a low-frequency (32.768 kHz), ultra-low power clock . This clock is used as to clock the power sequencers for boost, wakeup and retention of the chip.

Parameter	Symbol	Min	Typ	Max	Unit	Comment
Output Frequency					kHz	
Deviation across Temp					ppm/℃	

Table 7.2 Ring Oscillator specification

7.3 Sleep RTC

Real time clock is a crystal-based accurate clock (32.768 kHz) that is more stable than the ring oscillator across voltage and temperature. This clock is used for accurate sleep and wake up.

Parameter	Symbol	Min	Typ	Max	Unit	Comment
Output Frequency					kHz	

Table 7.3 Sleep RTC specification

7.4 CLK Register Map

Offset	Name	Description
0x00004	CPM_CPU_CFG	CPU clock config
0X00008	CPM_APB_CFG	APB clock config
0X0000C	CPM_REG_UPD	CPM update control
0X00010	CPM_SF_CFG	SFLASH clock config
0X00014	CPM_TIMER1_CFG	TIMER1 clock config
0X00018	CPM_TIMER2_CFG	TIMER2 clock config
0X0001C	CPM_TIMER3_CFG	TIMER3 clock config
0X00020	CPM_UART0_CFG	UART0 clock config
0X00024	CPM_UART1_CFG	UART1 clock config
0X00028	CPM_I2C_CFG	I2C clock config
0X0002C	CPM_I2C2_CFG	I2C2 clock config
0X00030	CPM_SPI0_CFG	SPI0 clock config
0X00034	CPM_SF1_CFG	SFLASH1 clock config
0X00038	CPM_I2C3_CFG	I2C3 clock config
0X0003C	CPM_KPP_CFG	KPP clock config
0X00040	CPM_BB_CFG	Baseband clock config
0X00044	CPM_CPU_TCLK_CFG	SysTick clock config
0X00048	CPM_AHB_CFG	AHB clock config
0X0004C	CPM_DMA_CFG	DMA clock config
0X00050	CPM_RAM_CFG	RAM clock config
0X00054	CPM_AUDIO_CFG	AUDIO clock config
0X00058	CPM_GPIO_CFG	GPIO clock config
0X0005C	CPM_QDEC_CFG	QDEC clock config
0X00060	CPM_SPI1_CFG	SPI1 clock config
0X00064	CPM_PHY_CFG	PHY clock config
0X00068	CPM_RNG_CFG	RNG clock config
0X0006C	CPM_I2S_CFG	I2S clock config
0X00070	CPM_STATUS_READ	CPM status
0X00074	CPM_ANA_IF_AHB_CFG	ANA_IF ahb clock config
0X00078	CPM_24G_CFG	2.4G clock config
0X0007C	CPM_ANA_IF_CFG	ANA_IF clock config
0x00080	CPM_SF2_CFG	Sflash2 clock config
0x00084	CPM_PMU_HIB_SPI_CFG	PMU hibernation clock config

CPM_CPU_CFG address offset: 0x0004

Bit	R/W	Reset	Name	Description
31:24	N/A	0x0	N/A	reserved
23	RW	0x1	ahb_clk_en_periph	Peripheral ahb clock enable 1: enable
22	RW	0x1	ahb_clk_en_ram	RAM ahb clock enable 1: enable
21:16	N/A	0x0	N/A	N/A
15:8	RW	0x2	cpu_div_coeff	cpu divider config
7:3	N/A	0x0	N/A	N/A
2	RW	0x0	cpu_div_sel	CPU clock select 1: select divided clock 0: select original clock
1	RW	0x1	cpu_div_en	CPU clock divider enable 1: enable
0	N/A	0x0	N/A	N/A

CPM_APB_CFG address offset: 0x0008

Bit	R/W	Reset	Name	Description
31:20	N/A	0x0	N/A	reserved
19	RW	0x0	rom_apb_soft_reset	ROM apb soft reset: 1: soft reset
18	RW	0x0	pso_apb_soft_reset	PSO apb soft reset: 1: soft reset
17	N/A	0x0	N/A	reserved
16	RW	0x0	pmu_apb_soft_reset	PMU apb soft reset: 1: soft reset
15:4	N/A	0x0	N/A	reserved
3	RW	0x1	rom_apb_gate_en	ROM apb clock gate 1: gate
2	RW	0x0	pso_apb_gate_en	PSO apb clock gate 1: gate
1	RW	0x1	rtc_apb_gate_en	RTC apb clock gate 1: gate
0	RW	0x0	pmu_apb_gate_en	PMU apb clock gate 1: gate

CPM_REG_UPD address offset: 0x000C

Bit	R/W	Reset	Name	Description
31:9	N/A	0x0	N/A	reserved
8	RW	0x0	reg_upd_rw_apb	Write 1 to update shadow reg of rw clock, self clear
7	R	0x0	reg_upd_rw_status	Bit 8 update status,

				1: update is done
6	R	0x0	reg_upd_32k_status	Bit2 update status, 1: update is done
5	R	0x0	reg_upd_xtal32m_status	Bit1 update status, 1: update is done
4	R	0x0	reg_upd_cpu_status	Bit0 update status, 1: update is done
3	RW	0x0	reg_upd_status_clr	Clear the status bits [6:4], write 1 to clear, self clear
2	RW	0x0	reg_upd_32k	Write 1 to update shadow reg of 32k clock, self clear
1	RW	0x0	reg_upd_xtal32m	Write 1 to update shadow reg of xtal32m clock, self clear
0	RW	0x0	reg_upd_cpu	Write 1 to update shadow reg of CPU clock, self clear

CPM_SF_CFG address offset: 0x0010

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:8	RW	0x0	sf_div_coeff	SFLASH divider config
7:5	N/A	0x0	N/A	reserved
4	RW	0x0	sf_soft_reset	SFLASH soft reset: 1: reset
3	N/A	0x0	N/A	reserved
2	RW	0x0	sf_div_sel	SFLASH divided clock select: 1: divided clock is selected
1	RW	0x0	sf_div_en	SFLASH divider enable: 1: enable 0: disable
0	RW	0x0	sf_gate_en	SFLASH clock gate: 1: gate 0: enable

CPM_TIMERx_CFG address offset: CPM_TIMER1_CFG: 0x0014, CPM_TIMER2_CFG: 0x0018, CPM_TIMER3_CFG: 0x001C

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:8	RW	0x0	timerx_div_coeff	TIMERx divider config
7:5	N/A	0x0	N/A	reserved
4	RW	0x0	timerx_soft_reset	TIMERx soft reset: 1: reset
3	N/A	0x0	N/A	reserved
2	RW	0x0	timerx_div_sel	TIMERx divided clock select:

				1: divided clock is selected
1	RW	0x0	timerx_div_en	TIMERx divider enable: 1: enable 0: disable
0	RW	0x1	timerx_gate_en	TIMERx clock gate: 1: gate 0: enable

CPM_UART0_CFG address offset: 0x0020

Bit	R/W	Reset	Name	Description
31:24	RW	0x40	uart0_div_coeff_frc	UART0 divider config (fraction part)
23:17	N/A	0x0	N/A	reserved
16:8	RW	0x9c	uart0_div_coeff_int	UART0 divider config (integer part)
7:5	N/A	0x0	N/A	reserved
4	RW	0x0	uart0_soft_reset	UART0 soft reset: 1: reset
3	N/A	0x0	N/A	reserved
2	RW	0x1	uart0_div_sel	UART0 divided clock select: 1: divided clock is selected
1	RW	0x1	uart0_div_en	UART0 divider enable: 1: enable 0: disable
0	RW	0x1	uart0_gate_en	UART0 clock gate: 1: gate 0: enable

CPM_UART1_CFG address offset: 0x0024

Bit	R/W	Reset	Name	Description
31:24	RW	0x40	uart1_div_coeff_frc	UART1 divider config (fraction part)
23:17	N/A	0x0	N/A	reserved
16:8	RW	0x9c	uart1_div_coeff_int	UART1 divider config (integer part)
7:5	N/A	0x0	N/A	reserved
4	RW	0x0	uart1_soft_reset	UART1 soft reset: 1: reset
3	N/A	0x0	N/A	reserved
2	RW	0x1	uart1_div_sel	UART1 divided clock select: 1: divided clock is selected
1	RW	0x1	uart1_div_en	UART1 divider enable: 1: enable 0: disable
0	RW	0x1	uart1_gate_en	UART1 clock gate: 1: gate 0: enable

CPM_I2C_CFG address offset: 0x0028

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:8	RW	0x0	i2c_div_coeff	I2C divider config
7:5	N/A	0x0	N/A	reserved
4	RW	0x0	i2c_soft_reset	I2C soft reset: 1: reset
3	N/A	0x0	N/A	reserved
2	RW	0x0	i2c_div_sel	I2C divided clock select: 1: divided clock is selected
1	RW	0x0	i2c_div_en	I2C divider enable: 1: enable 0: disable
0	RW	0x1	i2c_gate_en	I2C clock gate: 1: gate 0: enable

CPM_I2C2_CFG address offset: 0x002C

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:8	RW	0x0	i2c2_div_coeff	I2C2 divider config
7:5	N/A	0x0	N/A	reserved
4	RW	0x0	i2c2_soft_reset	I2C2 soft reset: 1: reset
3	N/A	0x0	N/A	reserved
2	RW	0x0	i2c2_div_sel	I2C2 divided clock select: 1: divided clock is selected
1	RW	0x0	i2c2_div_en	I2C2 divider enable: 1: enable 0: disable
0	RW	0x1	i2c2_gate_en	I2C2 clock gate: 1: gate 0: enable

CPM_SPI0_CFG address offset: 0x0030

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	RW	0x0	spi0_soft_reset	SPI0 soft reset: 1: reset
3:1	N/A	0x0	N/A	reserved
0	RW	0x1	spi0_gate_en	SPI0 clock gate: 1: gate

				0: enable

CPM_SF1_CFG address offset: 0x0034

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:8	RW	0x0	sf1_div_coeff	SFLASH1 divider config
7:5	N/A	0x0	N/A	reserved
4	RW	0x0	sf1_soft_reset	SFLASH1 soft reset: 1: reset
3	N/A	0x0	N/A	reserved
2	RW	0x0	sf1_div_sel	SFLASH1 divided clock select: 1: divided clock is selected
1	RW	0x0	sf1_div_en	SFLASH1 divider enable: 1: enable 0: disable
0	RW	0x0	sf1_gate_en	SFLASH1 clock gate: 1: gate 0: enable

CPM_I2C3_CFG address offset: 0x0038

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:8	RW	0x0	i2c3_div_coeff	I2C3 divider config
7:5	N/A	0x0	N/A	reserved
4	RW	0x0	i2c3_soft_reset	I2C3 soft reset: 1: reset
3	N/A	0x0	N/A	reserved
2	RW	0x0	i2c3_div_sel	I2C3 divided clock select: 1: divided clock is selected
1	RW	0x0	i2c3_div_en	I2C3 divider enable: 1: enable 0: disable
0	RW	0x1	i2c3_gate_en	I2C3 clock gate: 1: gate 0: enable

CPM_KPP_CFG address offset: 0x003C

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:8	RW	0x0	kpp_div_coeff	KPP divider config
7:5	N/A	0x0	N/A	reserved
4	RW	0x0	kpp_soft_reset	KPP soft reset: 1: reset

3	N/A	0x0	N/A	reserved
2	RW	0x0	kpp_div_sel	KPP divided clock select: 1: divided clock is selected
1	RW	0x0	kpp_div_en	KPP divider enable: 1: enable 0: disable
0	RW	0x1	kpp_gate_en	KPP clock gate: 1: gate 0: enable

CPM_BB_CFG address offset: 0x0040

Bit	R/W	Reset	Name	Description
31:9	N/A	0x0	N/A	reserved
8	RW	0x0	bb_ble_clk_sel	0: select xtal32m 1: select rc32m
7	RW	0x0	ahb3_soft_reset	Baseband ahb soft reset
6:4	RW	0x1	ahb3_clk_div	Baseband ahb divided cofig
3	RW	0x0	ahb3_div_en	Baseband ahb clock divider enable: 1: enable
2	RW	0x0	ahb3_clk_en	Baseband ahb clock enable: 1: enable
1	RW	0x0	bb_master_clk_en	Baseband master clock enable: 1: enable 0: disable
0	RW	0x1	bb_hclk_gate_en	Baseband hclk gate: 1: gate 0: enable

CPM_CPU_TCLK_CFG address offset: 0x0044

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:8	RW	0x0	cpu_tclk_div_coeff	SysTick divider config
7:3	N/A	0x0	N/A	reserved
2	RW	0x0	cpu_tclk_div_sel	SysTick divided clock select: 1: divided clock is selected
1	RW	0x0	cpu_tclk_div_en	SysTick divider enable: 1: enable 0: disable
0	RW	0x1	cpu_tclk_gate_en	SysTick clock gate: 1: gate 0: enable

CPM_AHB_CFG address offset: 0x0048

Bit	R/W	Reset	Name	Description
31:17	N/A	0x0	N/A	reserved
16	RW	0x0	ram_auto_gate_en	Ram clock auto gate enable 1: ram clock is auto gate while ram is not accessed
15:5	N/A	0x0	N/A	reserved
4	RW	0x0	ahb_soft_reset	ahb soft reset, Write 1 to soft reset, self clear
3:0	N/A	0x0	N/A	reserved

CPM_DMA_CFG address offset: 0x004C

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	RW	0x0	dma_soft_reset	DMA soft reset: 1: reset
3:1	N/A	0x0	N/A	reserved
0	RW	0x1	dma_gate_en	DMA clock gate: 1: gate 0: enable

CPM_RAM_CFG address offset: 0x0050

Bit	R/W	Reset	Name	Description
31:8	R	N/A	N/A	N/A
7:0	RW	0x7e	ram_gate_en	RAM clock gate: 1: gate 0: enable

CPM_AUDIO_CFG address offset: 0x0054

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	RW	0x0	audio_soft_reset	AUDIO soft reset: 1: reset
3:2	N/A	0x0	N/A	reserved
1	RW	0x1	audio_12m_gate_en	AUDIO 12Mhz clock gate: 1: gate 0: enable
0	RW	0x1	audio_gate_en	AUDIO clock gate: 1: gate 0: enable

CPM_GPIO_CFG address offset: 0x0058

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	RW	0x0	gpio_soft_reset	GPIO soft reset: Write 1 to soft reset, self clear
3:1	N/A	0x0	N/A	reserved
0	RW	0x1	gpio_gate_en	GPIO clock gate: 1: gate 0: enable

CPM_QDEC_CFG address offset: 0x005C

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:8	RW	0x0	qdec_div_coeff	QDEC divider config
7:5	N/A	0x0	N/A	reserved
4	RW	0x0	qdec_soft_reset	QDEC soft reset: 1: reset
3	N/A	0x0	N/A	reserved
2	RW	0x0	qdec_div_sel	QDEC divided clock select: 1: divided clock is selected
1	RW	0x0	qdec_div_en	QDEC divider enable: 1: enable 0: disable
0	RW	0x1	qdec_gate_en	QDEC clock gate: 1: gate 0: enable

CPM_SPI1_CFG address offset: 0x0060

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	RW	0x0	spi1_soft_reset	SPI1 soft reset: 1: reset
3:1	N/A	0x0	N/A	reserved
0	RW	0x1	spi1_gate_en	SPI1 clock gate: 1: gate 0: enable

CPM_PHY_CFG address offset: 0x0064

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	RW	0x0	phy_soft_reset	PHY soft reset: 1: reset
3:2	N/A	0x0	N/A	reserved

1	RW	0x1	phy_16m_gate_en	PHY 16mHz clock gate: 1: gate 0: enable
0	RW	0x1	phy_apb_gate_en	PHY apb clock gate: 1: gate 0: enable

CPM_RNG_CFG address offset: 0x0068

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	RW	0x0	rng_soft_reset	RNG soft reset: 1: reset
3:1	N/A	0x0	N/A	reserved
0	RW	0x1	rng_gate_en	RNG clock gate: 1: gate 0: enable

CPM_I2S_CFG address offset: 0x006C

Bit	R/W	Reset	Name	Description
31:7	N/A	0x0	N/A	reserved
6	RW	0x0	i2s_ext_inv	I2S external input clock inverted control in slave mode 1: external input clock is inverted
5	RW	0x0	i2s_tx_soft_reset	I2S TX soft reset: 1: reset
4	RW	0x0	i2s_rx_soft_reset	I2S RX soft reset: 1: reset
3:2	N/A	0x0	N/A	reserved
1	RW	0x0	i2s_tx_ahb_en	I2S TX ahb clock gate: 1: gate 0: enable
0	RW	0x0	i2s_rx_ahb_en	I2S RX ahb clock gate: 1: gate 0: enable

CPM_STATUS_READ address offset: 0x0070

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15	R	0x0	bb_ble_clk_sync_done	bb_ble clock sync done flag: 1: done
14	R	0x0	sf2_clk_sync_done	sf2 clock sync done flag: 1: done
13	R	0x0	sf1_clk_sync_done	sf1 clock sync done flag:

				1: done
12	R	0x0	cpu_tclk_sync_done	systick clock sync done flag: 1: done
11	R	0x0	qdec_clk_sync_done	qdec clock sync done flag: 1: done
10	R	0x0	kpp_clk_sync_done	kpp clock sync done flag: 1: done
9	R	0x0	i2c3_clk_sync_done	i2c3 clock sync done flag: 1: done
8	R	0x0	i2c2_clk_sync_done	i2c2 clock sync done flag: 1: done
7	R	0x0	i2c_clk_sync_done	i2c clock sync done flag: 1: done
6	R	0x0	uart1_clk_sync_done	uart1 clock sync done flag: 1: done
5	R	0x0	uart0_clk_sync_done	uart0 clock sync done flag: 1: done
4	R	0x0	timer3_clk_sync_done	timer3 clock sync done flag: 1: done
3	R	0x0	timer2_clk_sync_done	timer2 clock sync done flag: 1: done
2	R	0x0	timer1_clk_sync_done	timer1 clock sync done flag: 1: done
1	R	0x0	sf_clk_sync_done	sflash clock sync done flag: 1: done
0	R	0x0	main_clk_sync_done	main clock sync done flag: 1: done

CPM_ANA_IF_AHB_CFG address offset: 0x0074

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	RW	0x0	ana_if_ahb_soft_reset	ana_if ahb soft reset: 1: reset
3:1	N/A	0x0	N/A	reserved
0	RW	0x1	ana_if_ahb_gate_en	ana_if ahb clock gate: 1: gate 0: enable

CPM_24G_CFG address offset: 0x0078

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	RW	0x0	24G_soft_reset	24G soft reset: 1: reset

3:1	N/A	0x0	N/A	reserved
0	RW	0x1	24G_gate_en	24G clock gate: 1: gate 0: enable

CPM_ANA_IF_CFG address offset: 0x007C

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x1	ana_if_gate_en	ana_if clock gate: 1: gate 0: enable

CPM_SF2_CFG address offset: 0x0080

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:8	RW	0x0	sf2_div_coeff	SFLASH2 divider config
7:5	N/A	0x0	N/A	reserved
4	RW	0x0	sf2_soft_reset	SFLASH2 soft reset: 1: reset
3	N/A	0x0	N/A	reserved
2	RW	0x0	sf2_div_sel	SFLASH2 divided clock select: 1: divided clock is selected
1	RW	0x0	sf2_div_en	SFLASH2 divider enable: 1: enable 0: disable
0	RW	0x0	sf2_gate_en	SFLASH2 clock gate: 1: gate 0: enable

CPM_PMU_HIB_SPI_CFG address offset: 0x0084

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	RW	0x0	pmu_hib_soft_reset	pmu_hib soft reset: 1: reset
3:1	N/A	0x0	N/A	reserved
0	RW	0x1	pmu_hib_spi_gate_en	pmu_hib spi clock gate: 1: gate 0: enable

8 Peripherals

8.1 Pin Mux

8.1.1 Introduction

HS6621Cx has a configurable pin multiplexing module (Pin Mux) which can bring different peripherals on different GPIOs.

8.1.2 Main Features

Onmicro_pinmux has the following features:

- Peripheral pinmux can be config by REG_GPIOX_MUX
- There are 77 mux choices for the Peripheral pinmux.

8.1.3 Function Description

The pin multiplexing choices for all pads are shown in the following table.

There are 77 mux choices for the Pin Mux. When you pick a mux choice for an interface, make sure that all signal of the inteface should be configured to the picked mux choice. The following pin mux tables set the signals of an interface.

NAME	NUMBER
PINMUX_I2C_MST_SCK_CFG	1
PINMUX_I2C_MST_SDA_CFG	2
PINMUX_SPI0_MST_SDA_I_CFG	3
PINMUX_SPI0_MST_SDA_O_CFG	4
PINMUX_SPI0_MST_CSN_CFG	5
PINMUX_SPI0_MST_SCK_CFG	6
PINMUX_SPI1_MST_SDA_I_CFG	7
PINMUX_SPI1_MST_SDA_O_CFG	8
PINMUX_SPI1_MST_CSN_CFG	9
PINMUX_SPI1_MST_SCK_CFG	10
PINMUX_UART0_SDA_I_CFG	11
PINMUX_UART0_SDA_O_CFG	12
PINMUX_UART0_SIR_I_CFG	13
PINMUX_UART0_SIR_O_CFG	14
PINMUX_UART1_SDA_I_CFG	15

PINMUX_UART1_SDA_O_CFG	16
PINMUX_UART1_CTS_I_N_CFG	17
PINMUX_UART1_RTS_O_N_CFG	18
PINMUX_QDEC_XA_CFG	19
PINMUX_QDEC_XB_CFG	20
PINMUX_QDEC_YA_CFG	21
PINMUX_QDEC_YB_CFG	22
PINMUX_QDEC_ZA_CFG	23
PINMUX_SFLASH_CSN_1_CFG	24
PINMUX_SFLASH_SI_CFG	25
PINMUX_DMIC_IN_CFG	26
PINMUX_DMIC_CLK_CFG	27
PINMUX_GPIO_MODE_CFG	28
PINMUX_SFLASH_SO_CFG	29
PINMUX_SFLASH_HD_CFG	30
PINMUX_SFLASH_WP_CFG	31
PINMUX_SFLASH_CK_CFG	32
PINMUX_SFLASH_CSN_CFG	33
PINMUX_TIMER0_ETR_CFG	34
PINMUX_TIMER1_ETR_CFG	35
PINMUX_TIMER2_ETR_CFG	36
PINMUX_TIMER0_BKIN_CFG	37
PINMUX_TIMER1_BKIN_CFG	38
PINMUX_TIMER2_BKIN_CFG	39
PINMUX_TIMER0_IO_0_CFG	40
PINMUX_TIMER0_IO_1_CFG	41
PINMUX_TIMER0_IO_2_CFG	42
PINMUX_TIMER0_IO_3_CFG	43
PINMUX_TIMER0_TOGGLE_N_0_CFG	44
PINMUX_TIMER0_TOGGLE_N_1_CFG	45
PINMUX_TIMER0_TOGGLE_N_2_CFG	46
PINMUX_TIMER1_IO_0_CFG	47
PINMUX_TIMER1_IO_1_CFG	48
PINMUX_TIMER1_IO_2_CFG	49
PINMUX_TIMER1_IO_3_CFG	50
PINMUX_TIMER1_TOGGLE_N_0_CFG	51
PINMUX_TIMER1_TOGGLE_N_1_CFG	52
PINMUX_TIMER1_TOGGLE_N_2_CFG	53
PINMUX_TIMER2_IO_0_CFG	54
PINMUX_TIMER2_IO_1_CFG	55
PINMUX_TIMER2_IO_2_CFG	56

PINMUX_TIMER2_IO_3_CFG	57
PINMUX_TIMER2_TOGGLE_N_0_CFG	58
PINMUX_TIMER2_TOGGLE_N_1_CFG	59
PINMUX_TIMER2_TOGGLE_N_2_CFG	60
PINMUX_I2S_MST_SDI_CFG	61
PINMUX_I2S_MST_WS_CFG	62
PINMUX_I2S_MST_SCLK_CFG	63
PINMUX_I2S_SDO_0_CFG	64
PINMUX_I2S_SDO_1_CFG	65
PINMUX_I2S_RX_WS_CFG	66
PINMUX_I2S_RX_SCLK_CFG	67
PINMUX_I2C2_SCK_CFG	68
PINMUX_I2C2_SDA_CFG	69
PINMUX_I2C3_SCK_CFG	70
PINMUX_I2C3_SDA_CFG	71
PINMUX_SFLASH_1_SI_CFG	72
PINMUX_SFLASH_1_SO_CFG	73
PINMUX_SFLASH_1_HD_CFG	74
PINMUX_SFLASH_1_WP_CFG	75
PINMUX_SFLASH_1_CK_CFG	76
PINMUX_SFLASH_1_CSN_CFG	77

Table 8.1 Peripheral pinmux

8.1.4 Pin Mux Register Map

Adress	Name	Description
0X40000080	PIN_MUX_CTRL_1	Pinmux control
0X40000084	PIN_MUX_CTRL_2	Pinmux control
0X40000088	PIN_MUX_CTRL_3	Pinmux control
0X4000008C	PIN_MUX_CTRL_4	Pinmux control
0X40000090	PIN_MUX_CTRL_5	Pinmux control
0X40000094	PIN_MUX_CTRL_6	Pinmux control
0X40000098	PIN_MUX_CTRL_7	Pinmux control
0X4000009C	PIN_MUX_CTRL_8	Pinmux control

PIN_MUX_CTRL_1 address: 0x40000080

Bit	R/W	Reset	Name	Description
31	N/A	0x0	N/A	reserved
30:24	RW	0x0	gpio3_mux_reg	gpio3 mux config
22:16	RW	0x0	gpio2_mux_reg	gpio2 mux config
14:8	RW	0x0	gpio1_mux_reg	gpio1 mux config

6:0	RW	0x0	gpio0_mux_reg	gpio0 mux config

PIN_MUX_CTRL_2 address: 0x40000084

Bit	R/W	Reset	Name	Description
31	N/A	0x0	N/A	reserved
30:24	RW	0x0	gpio7_mux_reg	gpio7 mux config
23	N/A	0x0	N/A	reserved
22:16	RW	0x0	gpio6_mux_reg	gpio6 mux config
15	N/A	0x0	N/A	reserved
14:8	RW	0x0	gpio5_mux_reg	gpio5 mux config
7	N/A	0x0	N/A	reserved
6:0	RW	0x0	gpio4_mux_reg	gpio4 mux config

PIN_MUX_CTRL_3 address: 0x40000088

Bit	R/W	Reset	Name	Description
31	N/A	0x0	N/A	reserved
30:24	RW	0x0	gpio11_mux_reg	gpio11 mux config
23	N/A	0x0	N/A	reserved
22:16	RW	0x0	gpio10_mux_reg	gpio10 mux config
15	N/A	0x0	N/A	reserved
14:8	RW	0x0	gpio9_mux_reg	gpio9 mux config
7	N/A	0x0	N/A	reserved
6:0	RW	0x0	gpio8_mux_reg	gpio8 mux config

PIN_MUX_CTRL_4 address: 0x4000008C

Bit	R/W	Reset	Name	Description
31	N/A	0x0	N/A	reserved
30:24	RW	0x1c	gpio15_mux_reg	gpio15 mux config
23	N/A	0x0	N/A	reserved
22:16	RW	0x0	gpio14_mux_reg	gpio14 mux config
15	N/A	0x0	N/A	reserved
14:8	RW	0x0	gpio13_mux_reg	gpio13 mux config
7	N/A	0x0	N/A	reserved
6:0	RW	0x0	gpio12_mux_reg	gpio12 mux config

PIN_MUX_CTRL_5 address: 0x40000090

Bit	R/W	Reset	Name	Description
31	N/A	0x0	N/A	reserved
30:24	RW	0x1c	gpio19_mux_reg	gpio19 mux config
23	N/A	0x0	N/A	reserved
22:16	RW	0x1c	gpio18_mux_reg	gpio18 mux config
15	N/A	0x0	N/A	reserved
14:8	RW	0x1c	gpio17_mux_reg	gpio17 mux config

7	N/A	0x0	N/A	reserved
6:0	RW	0x1c	gpio16_mux_reg	gpio16 mux config

PIN_MUX_CTRL_6 address: 0x40000094

Bit	R/W	Reset	Name	Description
31	N/A	0x0	N/A	reserved
30:24	RW	0x1c	gpio23_mux_reg	gpio23 mux config
23	N/A	0x0	N/A	reserved
22:16	RW	0x1c	gpio22_mux_reg	gpio22 mux config
15	N/A	0x0	N/A	reserved
14:8	RW	0x1c	gpio21_mux_reg	gpio21 mux config
7	N/A	0x0	N/A	reserved
6:0	RW	0x1c	gpio20_mux_reg	gpio20 mux config

PIN_MUX_CTRL_7 address: 0x40000098

Bit	R/W	Reset	Name	Description
31	N/A	0x0	N/A	reserved
30:24	RW	0x1c	gpio27_mux_reg	gpio27 mux config
23	N/A	0x0	N/A	reserved
22:16	RW	0x1c	gpio26_mux_reg	gpio26 mux config
15	N/A	0x0	N/A	reserved
14:8	RW	0x1c	gpio25_mux_reg	gpio25 mux config
7	N/A	0x0	N/A	reserved
6:0	RW	0x1c	gpio24_mux_reg	gpio24 mux config

PIN_MUX_CTRL_8 address: 0x4000009C

Bit	R/W	Reset	Name	Description
31	N/A	0x0	N/A	reserved
30:24	RW	0x0	gpio31_mux_reg	gpio31 mux config
23	N/A	0x0	N/A	reserved
22:16	RW	0x1c	gpio30_mux_reg	gpio30 mux config
15	N/A	0x0	N/A	reserved
14:8	RW	0x1c	gpio29_mux_reg	gpio29 mux config
7	N/A	0x0	N/A	reserved
6:0	RW	0x1c	gpio28_mux_reg	gpio28 mux config

8.2 DMA

8.2.1 Introduction

Onmicro™ DMA is a direct memory access controller which transfers regions of data efficiently on bus.

8.2.2 Main Features

- Compliant with AMBA™ 2 AHB protocol specification
- Supports up to 8 DMA channels
- Supports up to 16 request/acknowledge pairs for hardware handshaking
- Provides the round-robin arbitration with 2 priority levels
- Supports 8/16/32-bit wide data transfer

8.2.3 Function Description

DMA supports up to 8 DMA channels. Each DMA channel provides a set of registers to describe the intended data transfers. Multiple DMA channels can be enabled concurrently, but the DMA controller services one channel at a time.

The following figure shows an illustration of data transfer timing for a channel. To prevent channels from being starved, the DMA controller services all ready-channels alternatively, performing at most SrcBurstSize data transfers each time. Consequently, the data transfers of a channel may be split into several chunks when the total transfer size (TranSize) is larger than the source burst size (SrcBurstSize). When the overall data transfers of a channel complete, the DMA controller will update the interrupt status register, IntStatus, and assert the dma_int interrupt signal if the terminal count interrupt is enabled.

The data transfers of a channel will be stopped when an error occurs. The data transfers of a channel can also be aborted by software. In either case, the DMA controller will disable the channel, and assert dma_int if the corresponding interrupt is enabled.

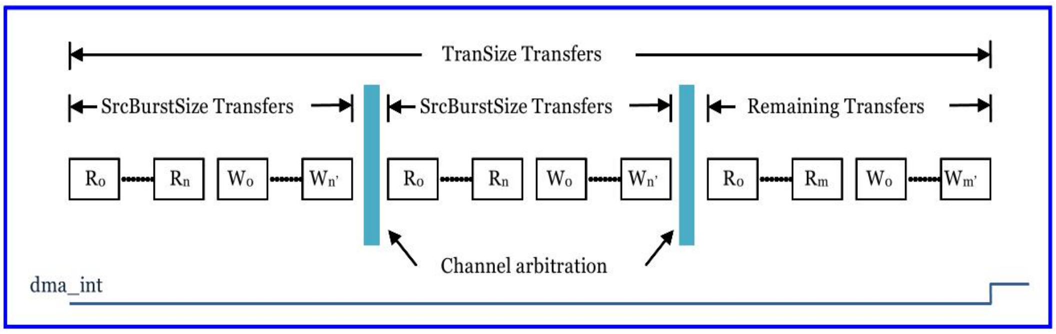


Figure 8.1 Example of DMA Data Transfers

8.2.3.1.Channel Arbitration

DMA provides two priority levels for channel arbitration. Every channel is associated with a priority level by the Priority field of the channel control register, ChnCtrl. During the channel arbitration, the DMA controller selects a high priority channel first. A low priority channel is only selected if there is no high priority channel. Channels of the same priority level will be selected by the round-robin scheme.

8.2.3.2.Hardware Handshak

DMA provides up to 16 pairs of hardware handshake signals (dma_req/dma_ack) for data transfers with low-speed devices.The following figure gives an example of hardware handshaking. The device should assert dma_req only when it prepares enough data to transfer or when it has enough empty space to receive the incoming data. The DMA controller only issues bus requests to read/write the data when it sees the dma_req asserted, avoiding holding the bus in the wait state indefinitely. The DMA controller asserts dma_ack when it completes SrcBurstSize data transfers from/to the device. The device should de-assert dma_req after detecting the assertion of dma_ack . The DMA controller should de-assert dma_ack after detecting the de-assertion of dma_req . If an error is encountered during the data transfers, the DMA controller will disable the channel without asserting dma_ack . The error handling software should reset both the source and destination of the DMA transfer to deassert dma_req.

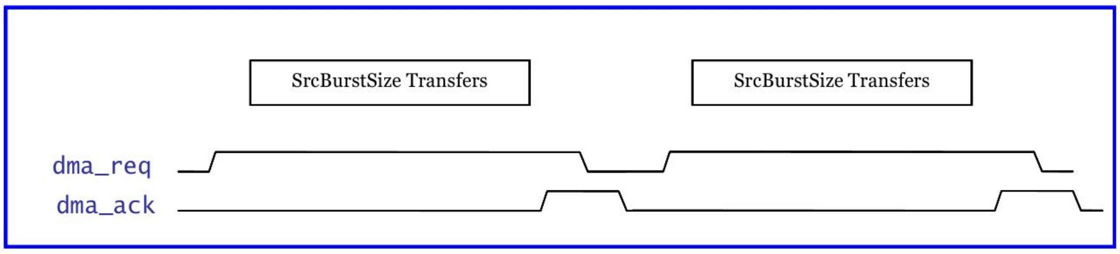


Figure 8.2 Example of Hardware Handshaking

8.2.3.3.Chain Transfer

DMA provides the chain transfer function, with which multiple blocks of data can be transferred consecutively without the intervention of the main processor.

Before a chain transfer is started, a linked list structure must be built to describe the data blocks to move and the associated control setups. The first element of the list (the head of the list) is described by the channel control registers. The rest of elements of the list are specified by the linked list descriptors stored in the memory, where the linked list descriptor holds the control values to load to the channel control registers to continue the data transfer.The following figure shows an example of the linked list structure.

When the channel is enabled, the DMA controller will first transfer data according to the channel control registers. After the data transfer completes, the DMA controller will continue the data transfer by following the ChnLLPointer. The content of the linked list descriptor pointed by ChnLLPointer will be loaded to the channel control registers if ChnLLPointer is not zero. The loaded descriptor becomes the new head of the list and this process repeats until the ChnLLPointer is zero.

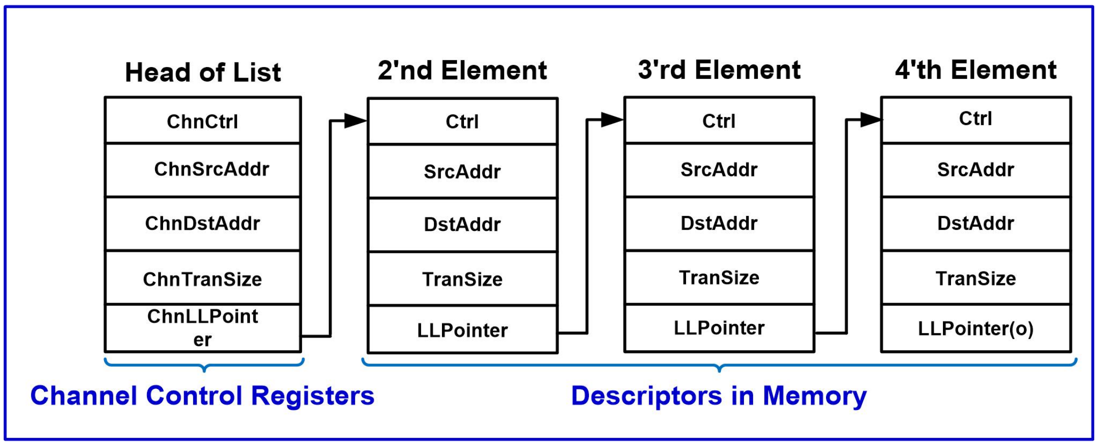


Figure 8.3 Linked List Structure for Chain Transfers

When the terminal count interrupt (IntTCMask) of a channel is enabled, the DMA controller will generate an interrupt and disable the channel when the data transfer for the head of the list is done. If the ChnLLPointer is not zero, the channel control registers will be preloaded with the next descriptor before the interrupt is generated. The interrupt handling software could resume the chain transfer by just re-enabling the channel.

The following table shows the format of the linked list descriptor. The bit field definition of each descriptor word is the same as the corresponding channel control register except the channel enable bit, which is reserved in the linked list descriptor.

Name	Offset	Description	Format
Ctrl	0x00	Channel control	See DMA Register Map
SrcAddr	0x04	Source address	See DMA Register Map
DstAddr	0x08	Destination address	See DMA Register Map
TranSize	0x0c	Total transfer size	See DMA Register Map
LLPointer	0x10	Linked list pointer	**See DMA Register Map**

Table 8.2 Format of Linked List Descriptor

8.2.3.4.Data Order

DMA provides three address control modes: increment mode, decrement mode, and fixed mode. At the increment mode, the address is increased after the DMA controller accesses a data of the source/destination. At the decrement mode, the address is decreased after the DMA controller accesses a data of the source/destination. At the fixed mode, the address remains unchanged after the DMA controller accesses a data of the source/destination.

When the address control mode of the source is the same as that of the destination, the DMA controller maintains the same byte order of the data between the source and the destination.When the address control mode of the source is opposite to that of the destination, the data written to the destination will be in the reverse byte order of that read from the source.The data order of the fixed mode is treated the same as that of the increment mode.The following figures illustrate the byte order of the data at the destination when the source address mode is increment,decrement,and fixed respectively.

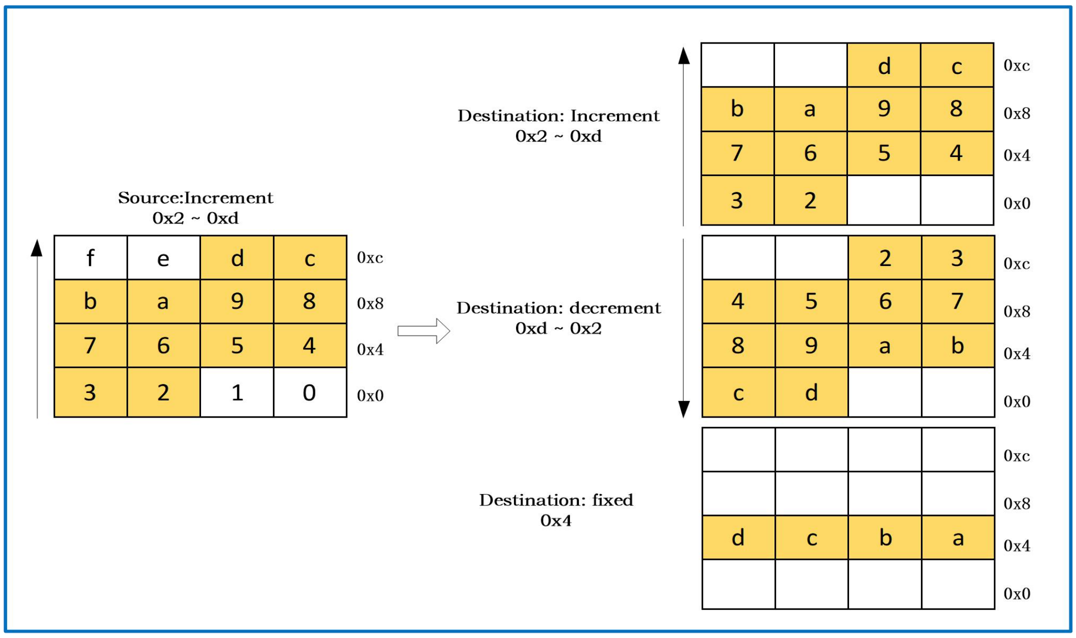


Figure 8.4 Data Order at the Destination when the Source Address Mode is the Increment Mode

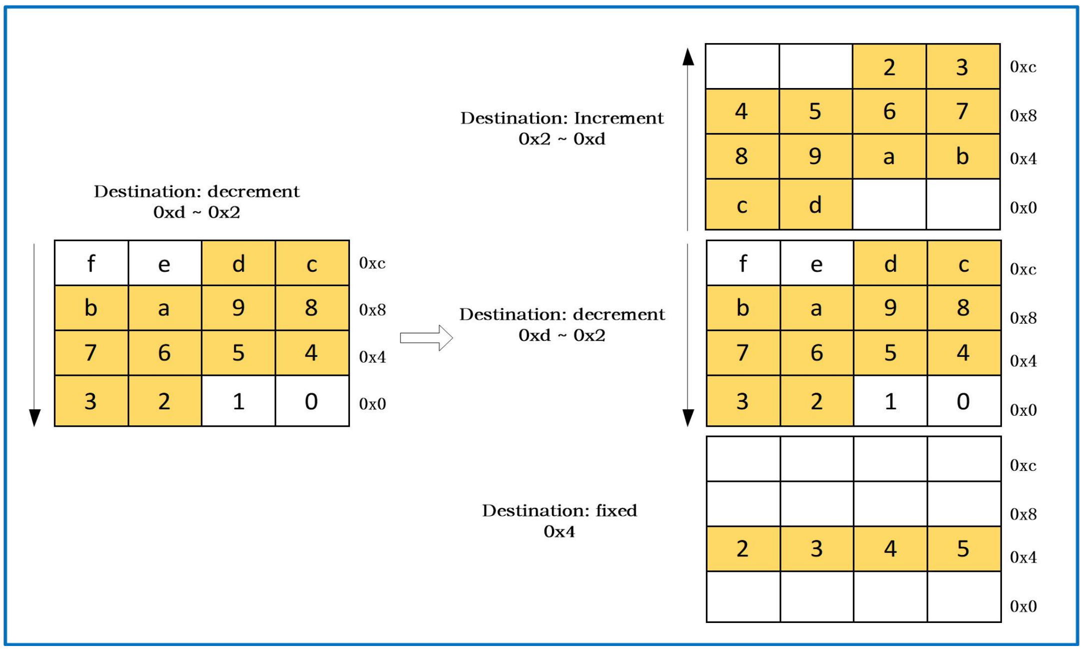


Figure 8.5 Data Order at the Destination when the Source Address Mode is the Decrement Mode

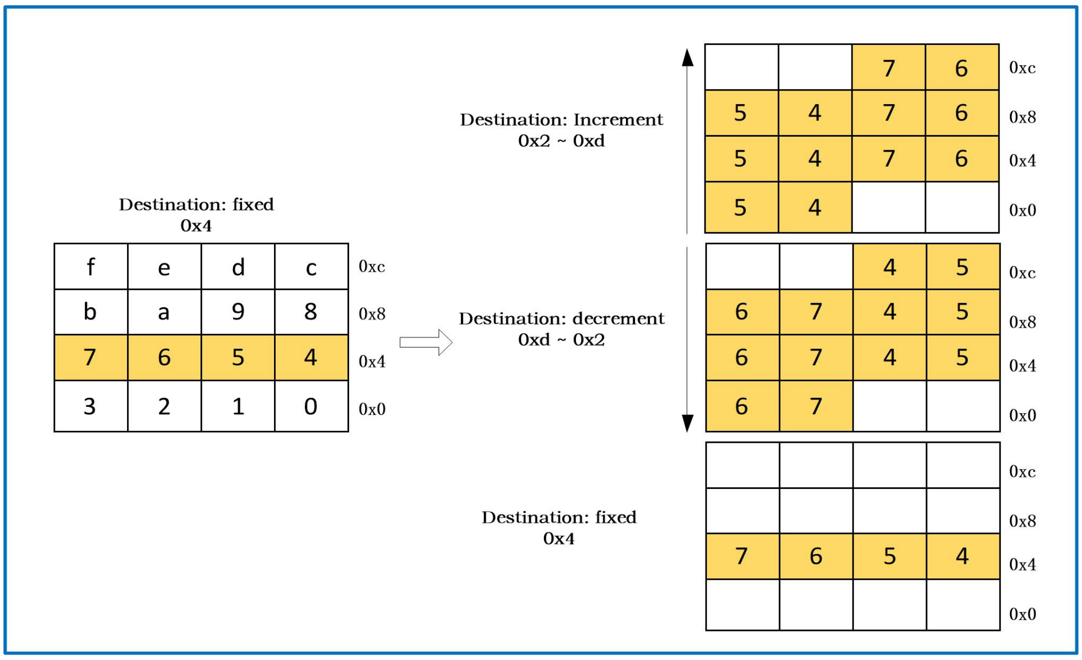


Figure 8.6 Data Order at the Destination

when the Source Address Mode is the Fixed Mode

8.2.4 DMA Register Map

offset	Name	Description
0x00	IdRev	ID and revision register
0x04	NA	Reserved
0x08	NA	Reserved
0x0C	NA	Reserved
0x14	NA	Reserved
0x18	NA	Reserved
0x1C	NA	Reserved
0x20	DMACtrl	DMAC control register
0x24	NA	Reserved
0x28	NA	Reserved
0x2C	NA	Reserved
0x30	IntStatus	Interrupt status register
0x34	ChEN	Channel enable register
0x38	NA	Reserved
0x3C	NA	Reserved
0x40	ChAbort	Channel abort register
0x44+n*0x14	ChnCtrl	Channel n control register
0x48+n*0x14	ChnSrcAddr	Channel n source address register
0x4C+n*0x14	ChnDstAddr	Channel n destination address register
0x50+n*0x14	ChnTranSize	Channel n transfer size register
0x54+n*0x14	ChnLLPointer	Channel n linked list pointer register

DMA Register Description

The following sections describe DMA registers in detail. The abbreviations for the Type column are summarized below:

RO: read only

WO: write only

R/W: readable and writable

R/W1C: readable and write 1 to clear

ID and Revision Register(offset 0x00)

This register holds the ID number and revision number. The reset values of the two revision fields are revision dependent.

Bit	R/W	Reset	Name	Description
31:12	R	0x01021	ID	ID number for DMAC
11:4	R	Revision dependent	RevMajor	Major revision number
3:0	R	Revision dependent	RevMinor	Minor revision number

DMAC Control Register (offset 0x20)

Bit	R/W	Reset	Name	Description
31:1	NA	NA	Reserved	NA
0	W	0x0	Reset	Software reset control. Set this bit to 1 to reset the DMA core and disable all channels.

Interrupt Status Register(offset 0x30)

This register contains the terminal count, error, and abort status. The terminal count status of a channel is asserted when the channel encounters the terminal counter event. The error/abort status of a channel is asserted when the channel encounters the error/abort event. There is one bit of status for each channel and the status bit is zero when the corresponding channel is not configured.

Bit	R/W	Reset	Name	Description
31:24	NA	NA	Reserved	NA
23:16	R/W$_1$C	0x0	TC	The terminal count status of DMA channels, one bit per channel. The terminal count status is asserted when a channel transfer finishes without abort or error event. 0=channel N has no terminal count status 1=channel N has terminal count status
15:8	R/W$_1$C	0x0	Abort	The abort status of channel, one bit per channel. The abort status is asserted when a channel transfer is aborted. 0=channel N has no abort status

				1=channel N has abort status
7:0	R/W$_1$C	0x0	Error	The error status, one bit per channel. The error status is asserted when a channel transfer encounters the following error events: **Bus error** **Unaligned address** **Unaligned transfer width** **Reserved configuration** 0=channel N has no error status 1=channel N has error status

Channel Enable Register (Offset 0x34)

The register shows the DMA channel enable status. The status fields only exist when the corresponding channels are configured. This register is an alias of the Enable fields of all ChnCtrl registers.

Bit	R/W	Reset	Name	Description
N:0	R	0x0	ChEN	Alias of the Enable field of all ChnCtrl registers

Channel Abort Register (Offset 0x40)

The register controls the abortion of the DMA channel transfers,one-bit per channel. Write 1 to stop the current transfer of the corresponding channel.The abort bit is automatically cleared by hardware after triggering the channel abort event.

Bit	R/W	Reset	Name	Description
N:0	W	0x0	ChAbort	Write 1 to this field to stop the channel transfer. The bits can only be set when the corresponding channels are enabled. Otherwise, the writes will be ignored for channels that are not enabled.

Channel n Control Register (Offset 0x44+n*0x14)

Bit	R/W	Reset	Name	Description
31:30	R/W	NA	Reserved	NA
29	R/W	0x0	Priority	Channel priority level. 0=lower priority 1=higher priority
28:25	NA	NA	Reserved	NA
24:22	R/W	0x0	SrcBurstSize	Source burst size. This field indicates the number of transfers before DMA channel re-arbitration. Total byte of a burst is SrcBurstSize * SrcWidth. 0x0: 1 transfer

				0x1: 2 transfers 0x2: 4 transfers 0x3: 8 transfers 0x4: 16 transfers 0x5: 32 transfers 0x6: 64 transfers 0x7: 128 transfers
21:20	R/W	0x2	SrcWidth	Source transfer width 0x0: byte transfer 0x1: half-word transfer 0x2: word transfer 0x3: reserved, setting the field with this value triggers error exception
19:18	R/W	0x2	DstWidth	Destination transfer width. Both the total transfer byte and the total burst bytes should be aligned to the destination transfer width; otherwise the error event will be triggered. For example, destination transfer width should be set as byte transfer if total transfer byte is not aligned to word or half-word. See SrcBurstSize field above for the definition of total burst byte and section 3.12 for the definition of the total transfer bytes. 0x0: byte transfer 0x1: half-word transfer 0x2: word transfer 0x3: reserved, set the field as this value triggers error exception
17	R/W	0x0	SrcMode	Source DMA handshake mode 0=normal mode 1=handshake mode
16	R/W	0x0	DstMode	Destination DMA handshake mode 0=normal mode 1=handshake mode
15:14	R/W	0x0	SrcAddrCtrl	Source address control 0x0: increment address 0x1: decrement address 0x2: fixed address 0x3: reserved, setting the field with this value triggers the error exception
13:12	R/W	0x0	DstAddrCtrl	Destination address control 0x0: increment address 0x1: decrement address

				0x2: fixed address 0x3: reserved, setting the field with this value triggers the error exception
11:8	R/W	0x0	SrcReqSel	Source DMA request select. Select the request/ack handshake pair that the source device is connected to.
7:4	R/W	0x0	DstReqSel	Destination DMA request select. Select the request/ack handshake pair that the destination device is connected to.
3	R/W	0x0	IntAbtMask	Channel abort interrupt mask. 0=allow the abort interrupt to be triggered 1=disable the abort interrupt
2	R/W	0x0	IntErrMask	Channel error interrupt mask. 0=allow the error interrupt to be triggered 1=disable the error interrupt
1	R/W	0x0	IntTCMask	Channel terminal count interrupt mask 0=allow the terminal count interrupt to be triggerd 1=disable the terminal count interrupt
0	R/W	0x0	Enable	Channel enable bit 0x0: disable 0x1: enable

Channel n Source Address Register (Offset 0x48+n*0x14)

Bit	R/W	Reset	Name	Description
31:0	R/W	0x0	SrcAddr	Source starting address. When a transfer completes, its value is updated to the ending address + sizeof(SrcWidth). This address must be aligned to the source transfer size; otherwise, an error event will be triggered.

Channel n Destination Address Register (Offset 0x4C+n*0x14)

Bit	R/W	Reset	Name	Description
31:0	R/W	0x0	DstAddr	Destination starting address. When a transfer completes, its value is updated to the ending address + sizeof(DstWidth). This address must be aligned to the destination transfer size; otherwise the error event will be triggered.

Channel n Transfer Size Register (Offset 0x50+n*0x14)

Bit	R/W	Reset	Name	Description
31:22	NA	NA	Reserved	NA
21:0	R/W	0x0	TranSize	Total transfer size from source. The total number of transferred bytes is TranSize *SrcWidth. The value is updated to zero when the DMA transfer is done. If a channel is enabled with zero total transfer size, the error event will be triggered and the transfer will be terminated.

Channel n Linked List Pointer Register (Offset 0x54+n*0x14)

Bit	R/W	Reset	Name	Description
31:2	R/W	0x0	LLPointer	Pointer to the next block descriptor. The pointer must be word aligned.
1:0	NA	NA	Reserved	NA

SSTATx address offset:
SSTAT0--0x020,SSTAT1--0x078,SSTAT2--0x0d0, SSTAT3--0x128

Bit	R/W	Reset	Name	Description
63:32	N/A	0x0	Reserved	Reserved
31:0	RW	0x0	SSTAT	Source status information retrieved by hardware from the address pointed to by the contents of the SSTATARx register.

DSTATx address offset:
DSTAT0--0x028, DSTAT1--0x080, DSTAT2--0x0d8, DSTAT3--0x130

Bit	R/W	Reset	Name	Description
63:32	N/A	0x0	Reserved	Reserved
31:0	RW	0x0	DSTAT	Destination status information retrieved by hardware from the address pointed to by the contents of the DSTATARx register.

SSTATARx address offset:
SSTATAR0--0x030, SSTATAR1--0x088, SSTATAR2--0x0e0, SSTATAR3--0x138

Bit	R/W	Reset	Name	Description
63:32	N/A	0x0	Reserved	Reserved
31:0	RW	0x0	SSTATAR	Pointer from where hardware can fetch the source status information, which is registered in the SSTATx register and written out to the SSTAT*x* register location of the LLI before the start of the next block.

DSTATARx address offset:

DSTATAR0--0x038,DSTATAR1--0x090,DSTATAR2--0x0e8,DSTATAR3--0x140

Bit	R/W	Reset	Name	Description
63:32	N/A	0x0	Reserved	Reserved
31:0	RW	0x0	DSTATAR	Pointer from where hardware can fetch the destination status information, which is registered in the DSTATx register and written out to the DSTAT*x* register location of the LLI before the start of the next block.

CFGx address offset:

CFG0--0x040, CFG1--0x098, CFG2--0x0f0, CFG3--0x148

Bit	R/W	Reset	Name	Description
63:47	N/A	0x0	Reserved	Reserved
46:43	RW	0x0	DEST_PER	Assigns a hardware handshaking interface (0 - DMAH_NUM_HS_INT-1) to the destination of channel *x* if the CFG*x*.HS_SEL_DST field is 0; otherwise, this field is ignored.
42:39	RW	0x0	SRC_PER	Assigns a hardware handshaking interface (0 - DMAH_NUM_HS_INT-1) to the source of channel *x* if the CFG*x*.HS_SEL_SRC field is 0; otherwise, this field is ignored.
38	RW	0x0	SS_UPD_EN	**Source Status Update Enable**. Source status information is fetched only from the location pointed to by the SSTATARx register, stored in the SSTATx register and written out to the SSTAT*x* location of the LLI if *SS_UPD_EN* is high.
37	RW	0x0	DS_UPD_EN	**Destination Status Update Enable**. Destination status information is fetched only from the location pointed to by the DSTATARx register, stored in the DSTATx register and written out to the DSTAT*x* location of the LLI if DS_UPD_EN is high.
36:34	RW	0x1	PROTCTL	**Protection Control** bits used to drive the AHB HPROT[3:1] bus.
33	RW	0x0	FIFO_MODE	**FIFO Mode Select**. Determines how much space or data

				needs to be available in the FIFO before a burst transaction request is serviced. 0 = Space/data available for single AHB transfer of the specified transfer width. 1 = Data available is greater than or equal to half the FIFO depth for destination transfers and space available is greater than half the fifo depth for source transfers.
32	RW	0x0	FCMODE	**Flow Control Mode**. Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller. 0 = Source transaction requests are serviced when they occur. Data pre-fetching is enabled. 1 = Source transaction requests are not serviced until a destination transaction request occurs.
31	RW	0x0	RELOAD_DST	**Automatic Destination Reload**. The DARx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated.
30	RW	0x0	RELOAD_SRC	**Automatic Source Reload**. The SARx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated.
29:20	RW	0x0	MAX_ABRST	**Maximum AMBA Burst Length**. Maximum AMBA burst length that is used for DMA transfers on this channel.
19	RW	0x0	SRC_HS_POL	**Source Handshaking Interface Polarity.** 0 = Active high 1 = Active low
18	RW	0x0	DST_HS_POL	**Destination Handshaking Interface Polarity.** 0 = Active high 1 = Active low
17:12	N/A	0x0	Reserved	Reserved

11	RW	0x1	HS_SEL_SRC	**Source Software or Hardware Handshaking Select**. This register selects which of the handshaking interfaces – hardware or software – is active for source requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware-initiated transaction requests are ignored. If the source peripheral is memory, then this bit is ignored.
10	RW	0x1	HS_SEL_DST	**Destination Software or Hardware Handshaking Select**. This register selects which of the handshaking interfaces – hardware or software – is active for destination requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware- initiated transaction requests are ignored. If the destination peripheral is memory, then this bit is ignored.
9	R	0x1	FIFO_EMPTY	Indicates if there is data left in the channel FIFO.
8	RW	0x0	CH_SUSP	**Channel Suspend**. Suspends all DMA data transfers from the source until this bit is cleared.
7:5	RW	Channnel number	CH_PRIOR	**Channel priority**. A priority of 3 is the highest priority, and 0 is the lowest.
4:0	N/A	0x0	Reserved	Reserved

SGRx address offset:
SGR0--0x048, SGR1--0x0a0, SGR2--0x0f8, SGR3--0x150

Bit	R/W	Reset	Name	Description
63:32	N/A	0x0	Reserved	Reserved
31:20	RW	0x0	SGC	Source gather count. Source contiguous transfer count between successive gather boundaries.

19:0	RW	0x0	SGI	Source gather interval.

DSRx address offset:
DSR0--0x050, DSR1--0x0a8, DSR2--0x100, DSR3--0x158

Bit	R/W	Reset	Name	Description
63:32	N/A	0x0	Reserved	Reserved
31:20	RW	0x0	DSC	Destination scatter count. Destination contiguous transfer count between successive scatter boundaries.
19:0	RW	0x0	DSI	Destination scatter interval.

RawTfr address offset: 0x2c0

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	RAW	Raw interrupt status.

RawBlock address offset: 0x2c8

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	RAW	Raw interrupt status.

RawSrcTran address offset: 0x2d0

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	RAW	Raw interrupt status.

RawDstTran address offset: 0x2d8

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	RAW	Raw interrupt status.

RawErr address offset: 0x2e0

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	RAW	Raw interrupt status.

StatusTfr address offset: 0x2e8

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	R	0x0	STATUS	Interrupt status.

StatusBlock address offset: 0x2f0

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	R	0x0	STATUS	Interrupt status.

StatusSrcTran address offset: 0x2f8

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	R	0x0	STATUS	Interrupt status.

StatusDstTran address offset: 0x300

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	R	0x0	STATUS	Interrupt status.

StatusErr address offset: 0x308

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	R	0x0	STATUS	Interrupt status.

MaskTfr address offset: 0x310

Bit	R/W	Reset	Name	Description
63:12	N/A	0x0	Reserved	Reserved
11:8	W	0x0	INT_MASK_WE	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
7:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	INT_MASK	Interrupt Mask 0 = masked 1 = unmasked

MaskBlock address offset: 0x318

Bit	R/W	Reset	Name	Description
63:12	N/A	0x0	Reserved	Reserved
11:8	W	0x0	INT_MASK_WE	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
7:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	INT_MASK	Interrupt Mask 0 = masked 1 = unmasked

MaskSrcTran address offset: 0x320

Bit	R/W	Reset	Name	Description
63:12	N/A	0x0	Reserved	Reserved
11:8	W	0x0	INT_MASK_WE	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
7:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	INT_MASK	Interrupt Mask 0 = masked 1 = unmasked

MaskDstTran address offset: 0x328

Bit	R/W	Reset	Name	Description
63:12	N/A	0x0	Reserved	Reserved
11:8	W	0x0	INT_MASK_WE	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
7:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	INT_MASK	Interrupt Mask 0 = masked 1 = unmasked

MaskErr address offset: 0x330

Bit	R/W	Reset	Name	Description
63:12	N/A	0x0	Reserved	Reserved
11:8	W	0x0	INT_MASK_WE	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
7:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	INT_MASK	Interrupt Mask 0 = masked 1 = unmasked

ClearTfr address offset: 0x338

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	W	0x0	CLEAR	Interrupt clear 0 = no effect 1 = clear interrupt

ClearBlock address offset: 0x340

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved

3:0	W	0x0	CLEAR	Interrupt clear 0 = no effect 1 = clear interrupt

ClearSrcTran address offset: 0x348

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	W	0x0	CLEAR	Interrupt clear 0 = no effect 1 = clear interrupt

ClearDstTran address offset: 0x350

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	W	0x0	CLEAR	Interrupt clear 0 = no effect 1 = clear interrupt

ClearErr address offset: 0x358

Bit	R/W	Reset	Name	Description
63:4	N/A	0x0	Reserved	Reserved
3:0	W	0x0	CLEAR	Interrupt clear 0 = no effect 1 = clear interrupt

StatusInt address offset: 0x360

Bit	R/W	Reset	Name	Description
63:5	N/A	0x0	Reserved	Reserved
4	R	0x0	ERR	OR of the contents of StatusErr register.
3	R	0x0	DSTT	OR of the contents of StatusDst register.
2	R	0x0	SRCT	OR of the contents of StatusSrcTran register.
1	R	0x0	BLOCK	OR of the contents of StatusBlock register.
0	R	0x0	TFR	OR of the contents of StatusTfr register.

ReqSrcReg address offset: 0x368

Bit	R/W	Reset	Name	Description
63:12	N/A	0x0	Reserved	Reserved
11:8	W	0x0	SRC_REQ_WE	Source request write enable 0 = write disabled 1 = write enabled
7:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	SRC_REQ	Source request

ReqDstReg address offset: 0x370

Bit	R/W	Reset	Name	Description
63:12	N/A	0x0	Reserved	Reserved
11:8	W	0x0	DST_REQ_WE	Destination request write enable 0 = write disabled 1 = write enabled
7:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	DST_REQ	Destination request

SglReqSrcReg address offset: 0x378

Bit	R/W	Reset	Name	Description
63:12	N/A	0x0	Reserved	Reserved
11:8	W	0x0	SRC_SGLREQ_WE	Single write enable 0 = write disabled 1 = write enabled
7:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	SRC_SGLREQ	Source single request

SglReqDstReg address offset: 0x380

Bit	R/W	Reset	Name	Description
63:12	N/A	0x0	Reserved	Reserved
11:8	W	0x0	DST_SGLREQ_WE	Destination write enable 0 = write disabled 1 = write enabled
7:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	DST_SGLREQ	Destination single or burst request

LstSrcReg address offset: 0x388

Bit	R/W	Reset	Name	Description
63:12	N/A	0x0	Reserved	Reserved
11:8	W	0x0	LSTSRC_WE	Source last transaction request write enable 0 = write disabled 1 = write enabled
7:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	LSTSRC	Source last transaction request 0 = Not last transaction in current block 1 = Last transaction in current block

LstDstReg address offset: 0x390

Bit	R/W	Reset	Name	Description
63:12	N/A	0x0	Reserved	Reserved
11:8	W	0x0	LSTDST_WE	Destination last transaction request

				write enable 0 = write disabled 1 = write enabled
7:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	LSTDST	Destination last transaction request 0 = Not last transaction in current block 1 = Last transaction in current block

DmaCfgReg address offset: 0x398

Bit	R/W	Reset	Name	Description
63:1	N/A	0x0	Reserved	Reserved
0	RW	0x0	DMA_EN	**HS_ahb_dmac Enable bit**. 0 = HS_ahb_dmac Disabled 1 = HS_ahb_dmac Enabled.

ChEnReg address offset: 0x3a0

Bit	R/W	Reset	Name	Description
63:12	N/A	0x0	Reserved	Reserved
11:8	W	0x0	CH_EN_WE	Channel enable write enable.
7:4	N/A	0x0	Reserved	Reserved
3:0	RW	0x0	CH_EN	Enables/Disables the channel. Setting this bit enables a channel; clearing this bit disables the channel. 0 = Disable the Channel 1 = Enable the Channel

8.3 SFLASH

8.3.1 Introduction

The Serial Peripheral Interface (SPI) is used primarily for a synchronous serial communication of host processor and peripherals.The SPI controller is Motorola SPI-compatible interface.Up to two devices can be connected using four chip selects, data-in (SPI_DO), data-out (SPI_DO) and clock (SPI_CLK) signals are common for all the four devices. SPI interface can also be used to connect to SPI flash devices; commands such as read/write are user configurable.

8.3.2 Main Features

Following features are supported:

- Motorola SPI compatible 4 wire interface up to 133MHz
- SPI Master Mode support only
- Four chip select support with software configurable settings
- DMA read and write support. DMA reads can be maximum size of 65535 Bytes (64KB -1)
- Command based read and writes
- Operating speed is software configurable
- Transparent read support for flash device

8.3.3 Function Description

The AHB Master I/F transfers data from the SPI FIFO to system memory for SPI reads or from system memory to the SPI FIFO for SPI writes. The CPU uses the AHB slave interface to setup a SPI read or write transactions.The single buffered FIFO is used for data transport through the AHB Master Interface.The same FIFO is used for both SPI reads and SPI writes. When a transaction is completed,an interrupt can be generated,if enabled, to signal the CPU that the requested transaction has completed.

For SPI devices that are less than 32 bits wide,the endianness of the SPI device needs to be considered.Since data is sent MSB-first,when the SPI device is little-endian,this module will internally swap the data bytes (for 8-bit devices) or half-words (for 16-bit devices) before sending and after receiving data from the SPI device.SPI device endianness is register configurable and programmed with the WIDTH field in the SPI Configuration Register.Frame work below indicates the bit order sent for a specific SPI device configuration:

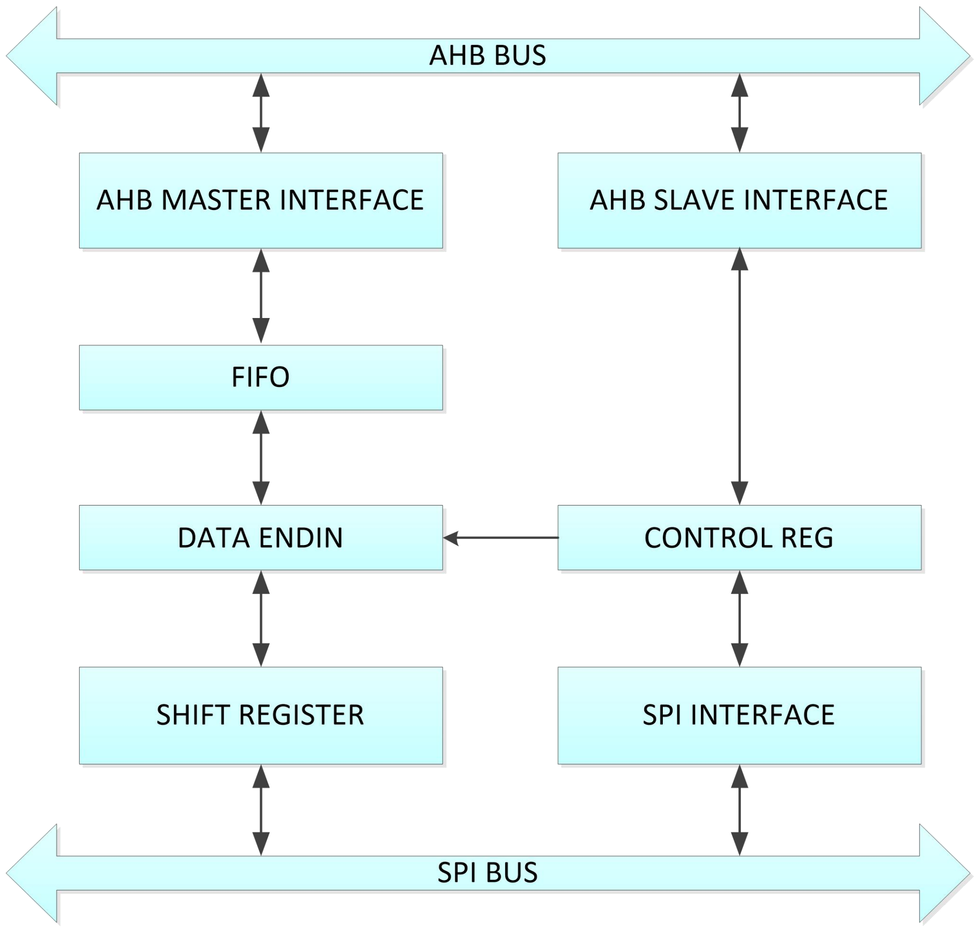


Figure 8.7 Specific SPI Device Configuration

8.3.4 SFLASH address map

SPI address is mapped to transparent read space. Below is the SPI address map:

Address Range	Description
0x5000 0000 to 0x50FF FFFF (un-cacheable)	Transparent read space (physical address)
0x1200 0000 to 0x12FF FFFF (cacheable/un-cacheable)	Transparent read space
0x0000 0000 to 0x00FF FFFF (cacheable/un-cacheable)	Remap address space
0x5100 0000 to 0x51FF FFFF	SPI control register space

8.3.5 SFLASH Register Map

Offset	Name	Description
0x00000	spi_intr_status	SPI Interrupt Status Register
0X00004	spi_raw_intr_status	SPI Raw Interrupt Status Register

0X00008	spi_intr_mask	SPI Interrupt Mask register
0X0000C	spi_command	SPI Command Register
0X00010	spi_command_data0_reg	SPI command data0 register
0X00014	spi_command_data1_reg	SPI command data1 register
0X00018	spi_read0_reg	SPI Read0 Register
0X0001C	spi_read1_reg	SPI Read1 Register
0X00020	spi_address_reg	SPI Address Register
0X00024	spi_read_opcode_reg	SPI Read Opcode Register
0X00028	spi_configuration_0	SPI Configuration Register 0
0X0002C	spi_cs_configuration_0	SPI CS Configuration Register 0
0X00030	spi_configuration_1	SPI Configuration Register 1
0X00034	spi_cs_configuration_1	SPI CS Configuration Register 1
0X00038	Transparent_remap	Transparent remap register
0X0003C	Reg_wp_hold	Reg_wp_hold register
0X00040	Reg_spi_cfg0	Reg_spi_cfg0 register
0X00044	Reg_spi_cfg1	Reg_spi_cfg1 register

Spi_intr_status address offset: 0x000

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	reserved	reserved
0	R	0x0	spi_cmd_done	SPI command done

spi_raw_intr_status address offset: 0x004

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	reserved	reserved
0	RW	0x0	spi_raw_intr_status	SPI command done interruption (internal interrupt value before mask). Set when the SPI command is complete. Writing 1 to clear the interrupt Status.

spi_intr_mask address offset: 0x008

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	reserved	reserved
0	RW	0x0	spi_cmd_done_mask	SPI command done interrupt mask. When 1, allows the interrupt to assert the interrupt.

spi_command address offset: 0x00c

Bit	R/W	Reset	Name	Description
31:12	RW	0x0	data_bytes	This field specifies the number of data bytes to transfer after the Command Data bits have been sent. Valid values

				are 0-65535. For a Read command, if this field is not a multiple of 4, zeros will be padded before data is written to system memory.This value is decremented during SPI the transaction until it reaches 0.
11:5	RW	0x0	cmd_bits	This field specifies the number of bits of the Command Data to send. Valid values are 0-64.This value is decremented during the SPI transaction until it reaches 0. Note that the 64 command data bits are defined in 2 32-bitregisters. The first 32 command data bits are sent from the COMMAND_DATA0 register, the next 32 bits from the COMMAND_DATA1 register.
4	RW	0x0	keep_cs	When this bit is set, the CS will remain asserted after the command is finished.
3:2	RW	0x0	chip_select	Chip Select. 0 = CS0 1 = CS1
1:0	RW	0x0	command	Command.This field is self-clearing after the SPI transition has completed. 0x0: NOP 0x1:Read.Data is transferred to Memory after the Command Data bits are sent 0x2: Write. Data is transferred to the SPI device after the Command Data bits are sent.

spi_command_data0_reg address offset: 0x010

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	command_data	This is the command data which is sent for the first32 SPI clock cycles, depending on the CMD_BITS field. It is sent MSB first (data is left-shifted out of bit31).This value is maintained during the SPI transaction.

spi_command_data1_reg address offset: 0x014

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	command_data	This is the command data which is sent

				for the first32 SPI clock cycles, depending on the CMD_BITS field. It is sent MSB first (data is left-shifted out of bit31).This value is maintained during the SPI transaction.

spi_read_data0_reg address offset: 0x018

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	read_data0	This register holds data that is captured during the first 32 SPI clock cycles. It is captured MSB first(data is left-shifted in from bit 0). Unused leading bits will be 0.

spi_read_data1_reg address offset: 0x01c

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	read_data1	This register holds data that is captured during the first 32 SPI clock cycles. It is captured MSB first(data is left-shifted in from bit 0). Unused leading bits will be 0.

spi_address_reg address offset: 0x020

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	address	This register holds the system memory address for data transfer. It is incremented by 4 as data is read from system memory (in the case of a Write command), or as data is written to system memory(in the case of a Read command). The lower two bits of this register are always 0, in order to force word alignment.

spi_read_opcode_reg address offset: 0x024

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	reserved	reserved
15:8	RW	0x3b	cs1_opcode	This register holds the OPCODE which is used to read from a Serial Flash device when a Transparent Read occurs in the CS1 address space
7:0	RW	0x3b	cs0_opcode	This register holds the OPCODE which is used to read from a Serial Flash device when a Transparent Read occurs in the CS0 address space

spi_configuration_0 address offset: 0x028

Bit	R/W	Reset	Name	Description
31:24	N/A	0x0	reserved	reserved
23	RW	0x0	lcd_rd_en	0x0: flash read and write, lcd write 0x1: lcd read
22:21	RW	0x0	rgb_mode	0x0:flash, RGB565 1-wire/2-wire data-lane 0x2: RGB888 1-wire/2-wire data-lane
20:18	RW	0x0	lcd_spi_ctrl	0x0: flash-mode; 0x1: RGB565 3-wire, 1-wire data-lane 0x2: RGB565 4-wire, 1-wire data-lane 0x3: RGB565 2-wire data-lane 0x4:RGB666/RGB888 3-wire, 1-wire data-lane 0x5:RGB666/RGB888 4-wire, 1-wire data-lane 0x6:RGB666 2-wire data-lane 0x7:RGB888 2-wire data-lane
17:16	RW	0x0	width	Width of the data reads/writes. This field causes data to be properly written to devices which are less than 32-bits wide in little-endian system. Big-endian systems should use the 32-bit data setting. 00: 8-bit data 01: 16-bit data 1X: 32-bit data Note: 0x1 for RGB565; 0x2 for RGB666 and RGB888
15	N/A	0x0	reserved	reserved
14	RW	0x0	fe_dly_sample	Falling edge delayed sampling. 1: sample SPI_DI on falling edge of delayed sampling clock (SPI_CLK_DLY). 0: sample SPI_DI on falling edge of internal sampling clock (SPI_CLK). Enabling this bit will allow higher frequency operation. If set, the dly_sample[13:12] must beset to 1 or greater. This is only valid for SPI modes 0 and3. For modes 1 and 2, these bits must be set to 0.

13:12	RW	0x0	dly_sample	Delayed Sampling, the number of REF_CLKs after the falling edge of SPI_CLK to sample SPI_DI. Setting these bits will allow higher frequency operation. If fe_dly_sample[14] is set, this field must be set to 1 or greater. This is only valid for SPI modes 0 and 3. For modes 1 and 2, these bits must be set to 0.
11	N/A	0x0	reserved	reserved
10	RW	0x0	bp_ clock_div	Bypass clock divider
9	RW	0x0	cpol	Clock polarity. 0: The clock is low during idle times, and each clock pulse consists of a rising edge followed by a falling edge. 1: The clock is high during idle times, and each clock pulse consists of a falling edge followed by a rising edge.
8	RW	0x0	cpha	Clock phase. 0: Input data is clocked on first edge of each clock pulse. 1: Input data is clocked on the second edge of each clock pulse.
7:0	RW	0x2	clock_div	This register is the divider for the system clock and to generate the SPI clock. Only even values should be programmed in order to keep a 50% duty cycle clock. The minimum value of this register is 2.

spi_cs_configuration_0 address offset: 0x02c

Bit	R/W	Reset	Name	Description
31:24	RW	0x0a	cs_recover	Chip Select Recover time. This is the number of system cycles that must pass after the chip select is de_asserted before the same or any other chip select can be asserted.
23:16	RW	0x0a	cs_hold	Chip Select Hold time. This is the number of system clock cycles between the last clock and the de-assertion of the chip select.
15:8	RW	0x0a	cs_setup	Chip Select Setup time. This is the number of system clock cycles between the assertion of the

				chip select and the first clock pulse.
7:1	N/A	0x0	reserved	reserved
0	RW	0x0	cs_pol	Chip Select Polarity. 0: Chip select is active low. 1: Chip select is active high

spi_configuration_1 address offset: 0x030

Bit	R/W	Reset	Name	Description
31:24	N/A	0x0	reserved	reserved
23	RW	0x0	lcd_rd_en	0x0: flash read and write, lcd write 0x1: lcd read
22:21	RW	0x0	rgb_mode	0x0:flash,RGB565 1-wire/2-wire data-lane 0x2: RGB888 1-wire/2-wire data-lane
20:18	RW	0x0	lcd_spi_ctrl	0x0: flash-mode; 0x1: RGB565 3-wire, 1-wire data-lane 0x2: RGB565 4-wire, 1-wire data-lane 0x3: RGB565 2-wire data-lane 0x4:RGB666/RGB888 3-wire, 1-wire data-lane 0x5:RGB666/RGB888 4-wire, 1-wire data-lane 0x6:RGB666 2-wire data-lane 0x7:RGB888 2-wire data-lane
17:16	RW	0x0	width	Width of the data reads/writes. This field causes data to be properly written to devices which are less than 32-bits wide in little-endian system.Big-endian systems should use the 32-bit data setting. 00: 8-bit data 01: 16-bit data 1X: 32-bit data Note: 0x1 for RGB565; 0x2 for RGB666 and RGB888
15	N/A	0x0	reserved	reserved
14	RW	0x0	fe_dly_sample	Falling edge delayed sampling. 1: sample SPI_DI on falling edge of delayed sampling clock (SPI_CLK_DLY). 0: sample SPI_DI on falling edge of internal sampling clock (SPI_CLK). Enabling this bit will allow higher frequency operation. If set, the dly_sample[13:12] must beset to 1 or greater. This is only valid for SPI modes 0 and3. For modes 1 and 2, these

				bits must be set to 0.
13:12	RW	0x0	dly_sample	Delayed Sampling, the number of REF_CLKs after the falling edge of SPI_CLK to sample SPI_DI. Setting these bits will allow higher frequency operation. If fe_dly_sample[14] is set, this field must be set to 1 or greater. This is only valid for SPI modes 0 and 3. For modes 1 and 2, these bits must be set to 0.
11	N/A	0x0	reserved	reserved
10	RW	0x0	bp_ clock_div	Bypass clock divider
9	RW	0x0	cpol	Clock polarity. 0: The clock is low during idle times, and each clock pulse consists of a rising edge followed by a falling edge. 1: The clock is high during idle times, and each clock pulse consists of a falling edge follow edby a rising edge.
8	RW	0x0	cpha	Clock phase. 0: Input data is clocked on first edge of each clock pulse. 1: Input data is clocked on the second edge of each clock pulse.
7:0	RW	0x2	clock_div	This register is the divider for the system clock and to generate the SPI clock. Only even values should be programmed in order to keep a 50% duty cycle clock. The minimum value of this register is 2.

spi_cs_configuration_1 address offset: 0x034

Bit	R/W	Reset	Name	Description
31:24	RW	0x0a	cs_recover	Chip Select Recover time. This is the number of system cycles that must pass after the chip select is de-asserted before the same or any other chip select can be asserted.
23:16	RW	0x0a	cs_hold	Chip Select Hold time. This is the number of system clock cycles between the last clock and the de-assertion of the chip select.
15:8	RW	0x0a	cs_setup	Chip Select Setup time. This is the number of system clock cycles between the assertion of the

				chip select and the first clock pulse.
7:1	N/A	0x0	reserved	reserved
0	RW	0x0	cs_pol	Chip Select Polarity. 0: Chip select is active low. 1: Chip select is active high

spi_Transparent_remap address offset: 0x038

Bit	R/W	Reset	Name	Description
31:25	N/A	0x0	reserved	reserved
24:0	RW	0x0	Remap_base	When controller is working on transparent access mode the address to spi device will be address on ahb bus ORed remap_base configuration in this field.

WP_HOLD address offset: 0x03C

Bit	R/W	Reset	Name	Description
31:2	N/A	0x0	reserved	reserved
1	RW	0x0	HOLD	Hold Bit output to spi device.
0	RW	0x0	WP	Write Protect Bit output to spi device.

SW_SPI_CFG0 address offset: 0x040

Bit	R/W	Reset	Name	Description
30:24	RW	0x0	Sw_dummy_cycle_cnt	dummy cycle before data phase.
22:16	RW	0x0	Cmd_p1_bit_cnt	Spi cmd phase1 bit count – the cmd phase1 data will come from reg_cmd_data1.

13:12	RW	0x0	Cmd_p1_bus_width	Spi cmd phase1 bus width
				0: 1bit;
				1: 2bit;
				2: 4bit

9:8	RW	0x0	Cmd_p0_bus_width	Spi cmd phase0 bus width
				0: 1bit;
				1: 2bit;
				2: 4bit

6:0	RW	0x0	Cmd_p0_bit_cnt	Spi cmd phas0 bit count. The cmd phase1 data will come from reg_cmd_data0.

SW_SPI_CFG1 address offset: 0x044

Bit	R/W	Reset	Name	Description
31	RW	0x0	Sw_cfg_en	1:the controller drives spi sequence to spi device according to SW configuration in SW_SPI_CFG0/

				SW_SPI_CFG1 register. 0: the controller decode sequence info from spi cmd.
30:27	N/A	0x0	reserved	reserved
26:24	RW	0x0	Buf_width_bytes	Buffer width count by bytes.

21:20	RW	0x0	Sdata_bus_width	Spi data phase bus width
				0: 1bit;
				1: 2bit;
				2: 4bit

19:0	RW	0x0	Sdata_byte_cnt	In/Out data byte count.

8.3.6 SFLASH software program guide

8.3.6.1.Read ID

In order to read the ID of the flash device, the SPI Master sends the 8- bit Read-ID command; then the flash sends back the 24-bit ID value (total of 32 bits are transferred). Hence the number of command bits for this transaction will be 32; the number of data bytes will be 0. The ID value will be available on ReadData0 register.

- Write the SPI Command Data0 register with the Read-ID command (0x9f); please remember the command is transmitted MSB first; hence write 0x9f000000 (SPI Command Data1 register is not used for this transaction).
- Write the SPI Command Register with the following details: Number of command bits = 32, Number of data bytes = 0, Keep_CS = 0, Chip_select = Required Chip select number (example 0), Command = Read i.e. 1.
- Wait until the command completes. One way to do this is by polling the spi_raw_intr_status bit in the SPI Raw Interrupt Status Register. Another option is to use the interrupt.
- Read the ID Value from SPI Read0 register (bits [23:0]). The data is captured MSB first i.e., left shifted from bit 0, so the LSB will always be at 0.

8.3.6.2.Read Status Register

The Read-Status-Register command can be issued after initiating an erase or write operation to check the status of that operation. The SPI Master sends the 8-bit command, and then the flash sends back the 8-bit status value. Thus a total of 16 bits are transferred. Hence the number of command bits for this transaction will be 16; the number of data bytes will be 0. The Status value will be available on the lower 8-bits of the Read Data0 register.

- Write the SPI Command Data0 register with the Read-Status command (0x05); please remember the command is transmitted MSB first; hence write 0x05000000 (SPI Command Data1 register is not used for this transaction).
- Write the SPI Command Register with the following details: Number of command

bits = 16, Number of data bytes = 0, Keep_CS = 0, Chip_select = Required Chip select number (example 0), Command = Read i.e. 1.

- Wait until the command completes. One way to do this is by polling the spi_raw_intr_status bit in the SPI Raw Interrupt Status Register.Another option is to use the interrupt.
- Read the Status Value from SPI Read0 register (bits [7:0]). The data is captured MSB first i.e., left shifted from bit 0, so the LSB will always be at 0.

8.3.6.3.DMA Write Operation

For DMA write operation, only a maximum of Page Size number of bytes supported by the flash can be written at a time. The Page Write or Page Program command (8-bits) is sent along with the 24-bit start flash address; this is followed by the required number of data bytes. Thus the number of command bits should be set to 32 and the number of data bytes is set to the required number.

- Perform a Sector Erase operation if the flash supports only Page Program command; if it supports Page Write then there is no need to do a separate Erase command.
- Send Write enable command to the flash.
- Write the SPI Command Data0 register with the Page Program (Or Page Write) command and the 24-bit start address in flash; please remember the command is transmitted MSB first; hence for example if the start address in flash is 0, then write 0x02000000 to do a page program. (SPI Command Data1register is not used for this transaction).
- Set the DMA Source Address in the SPI Address Register.
- Write the SPI Command Register with the following details: Number of command bits=32, Number of data bytes=N (required number), Keep_CS=0, Chip_select=Required Chip select number (example0), Command = Write i.e. 2.
- Wait until the command completes. One way to do this is by polling the spi_raw_intr_status bit in the SPI Raw Interrupt Status Register. Another option is to use the interrupt.
- Use the Read-Status-Register command to find out when the Write operation is completed by flash.

8.3.6.4.DMA Read Operation

For DMA Read operation, the whole flash can be read using a single command. Normal Read command (0x03) can be used only up-to a certain frequency (example 20 MHz for M25P128); above that frequency the Fast Read command (0x0b) should be used. When the Normal Read command is used the SPI Master sends the 8-bit read command along with a 24-bit start flash address; thus the number of command bits will be 32 for normal read command. When the Fast Read command is used the SPI Master sends the 8-bitFast-read command along with a 24-bit start flash address; the flash device returns a dummy byte before starting to return valid data. Hence the number of command bits for

Fast-read command will be 40 bits.

- Write the SPI Command Data0 register with the Normal Read command (0x03) or the Fast-Read command (0x0b) based on the SPI clock frequency, along with the 24-bit start flash address; please remember the command is transmitted MSB first; hence for example if the start address in flash is 0, then write0x0b000000 to do a fast read. Write 0x00000000 to SPI Command Data1 register for fast read to make sure the MOSI line does not toggle during the dummy byte interval; this is safer.
- Set the DMA Destination Address in the SPI Address Register.
- Write the SPI Command Register with the following details: Number of command bits=32 for normal read OR 40 for Fast read, Number of data bytes=N (required number), Keep_CS=0, Chip_select =Required Chip select number (example 0), Command = Read i.e. 1.
- Wait until the command completes. One way to do this is by polling the spi_raw_intr_status bit in the SPI Raw Interrupt Status Register. Another option is to use the interrupt.
- The data will be available in the system memory starting from address DMA Destination Address.

8.4 GPIO

8.4.1 Introduction

HS6621Cx has GPIOs which can connect to various signal interfaces.The following figure shows the basic structure of an I/O Port bit.

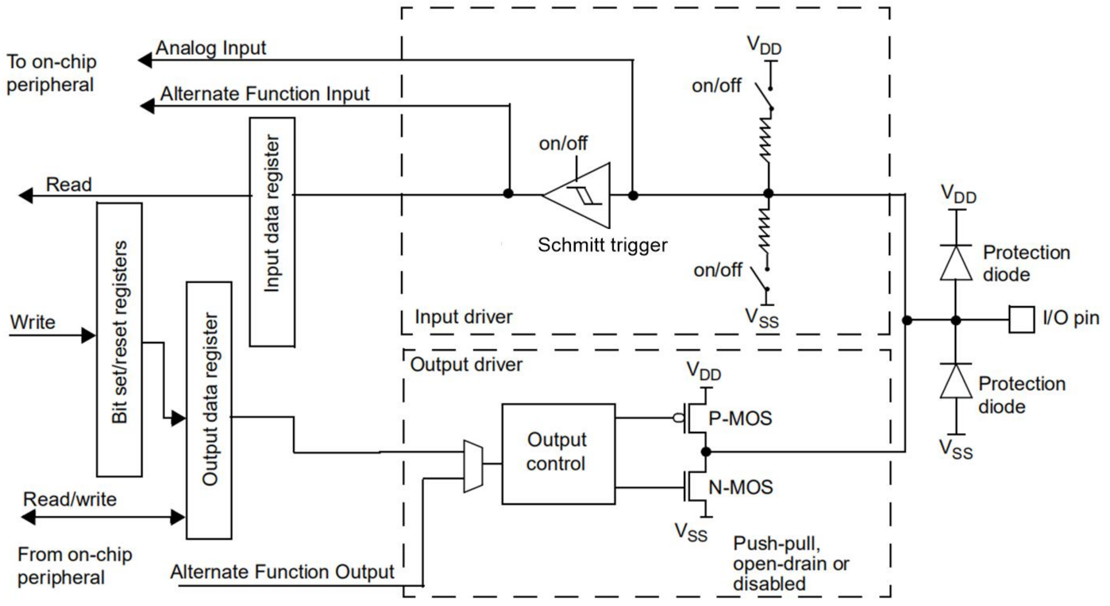


Figure 8.8 Basic structure of a standard I/O port bit

8.4.2 Main Features

- HS6621CB: 32 general-purpose I/O GPIOs (max)
- HS6621CG: 32 general-purpose I/O GPIOs (max)
- HS6621CM: 17 general-purpose I/O GPIOs (max)
- HS6621CQ: 21 general-purpose I/O GPIOs (max)

8.4.3 Function Description

The digital GPIO pads can be configured to set direction, enable pull-up/pull-down resistors and enable output retention. GPIOs can also be read or written by firmware for applications that need direct access, by using the GPIOx registers.

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the General Purpose IO (GPIO) Ports, can be individually configured by software in several modes:

- Input floating
- Input pull-up

- Input-pull-down
- Output open-drain
- Output push-pull
- Open-drain,pull up

The default mode of GPIO4,GPIO10~GPIO30 is pull-up.

TheGPIO1~GPIO32 are normal GPIO,and the GPIO33~GPIO38 can be used by flash.

8.4.3.1.External interrupt

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode.And it can be set to trigger in a variety of ways through the software configuration register.

- Falling edge
- Rising edge
- Both edge
- High level
- Low level
- Disable trigger

8.4.3.2.Input configuration

When the I/O Port is programmed as Input:

- The Output Buffer is disabled.
- The Schmitt Trigger Input is activated.
- The weak pull-up and pull-down resistors are activated or not depending on input configuration (pull-up, pull-down or floating).
- The data present on the I/O pin is sampled into the Input Data register.
- A read access to the Input Data register obtains the I/O State.

The following figure shows the Input Configuration of the I/O Port bit.

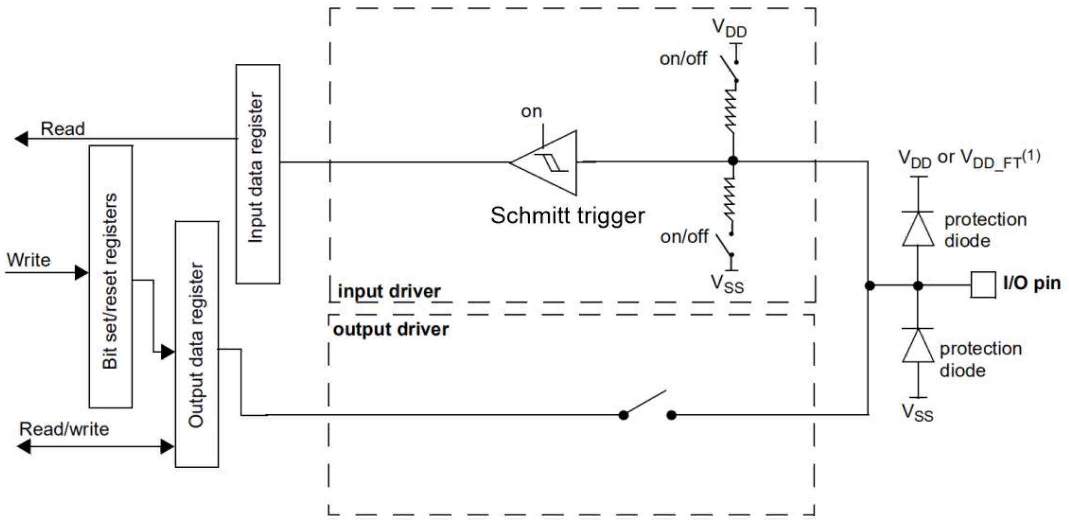


Figure 8.9 Input floating/pull up/pull down configurations

8.4.3.3.Output configuration

When the I/O Port is programmed as Output:

- The Output Buffer is enabled:
  - Open Drain Mode: A “0” in the Output register activates the N-MOS while a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated).
  - Push-Pull Mode: A “0” in the Output register activates the N-MOS while a “1” in the Output register activates the P-MOS.
- The Schmitt Trigger Input is activated.
- The weak pull-up and pull-down resistors are disabled.
- The data present on the I/O pin is sampled into the Input Data register
- A read access to the Input Data register gets the I/O state in open drain mode.
- A read access to the Output Data register gets the last written value in Push-Pull mode.

The following figure shows the Output configuration of the I/O Port bit.

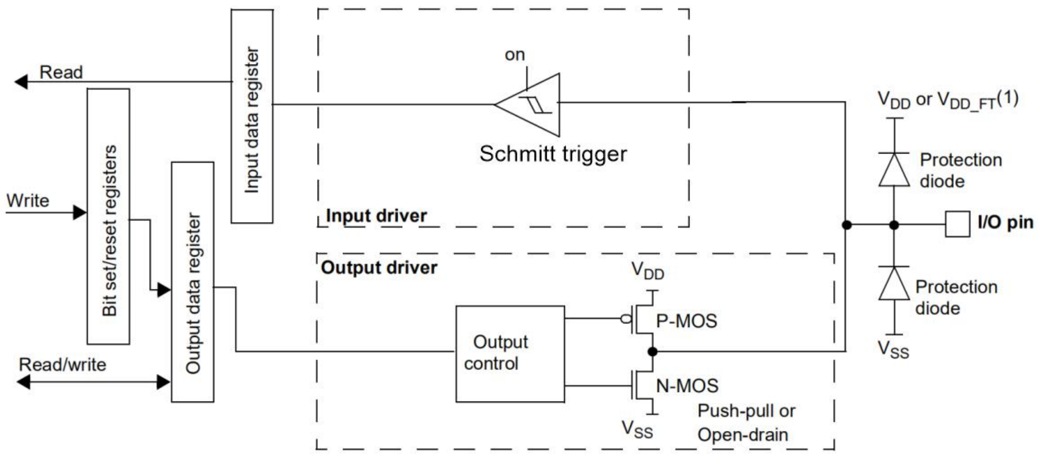


Figure 8.10 Output configuration

8.4.3.4.Analog configuration

When the I/O Port is programmed as Analog configuration:

- The Output Buffer is disabled.
- The Schmitt Trigger Input is de-activated providing zero consumption for every analog value of the I/O pin.The output of the Schmitt Trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled.
- Read access to the Input Data register gets the value “0”.

The following figure shows the high impedance-analog configuration of the I/O Port bit.

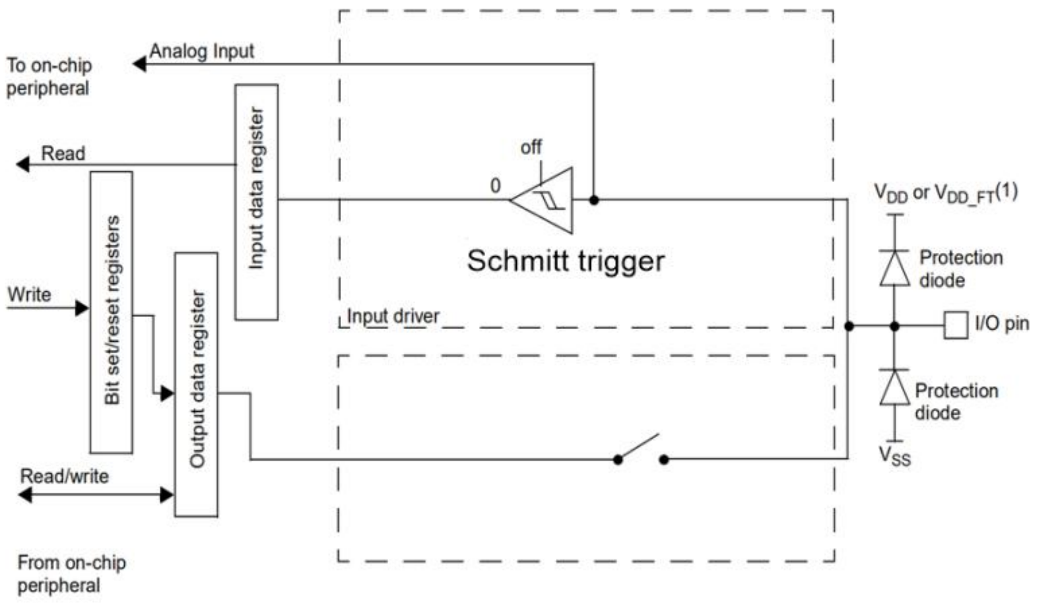


Figure 8.11 High impedance-analog configuration

8.4.4 GPIO Register Map

Offset	Name	Description
0x00000	DATA	Data
0X00004	DATAOUT	Data output latch
0X00010	OUTENSET	Output enable set
0X00014	OUTENCLR	Output enable clear
0X00018	ALTFUNCSET	Alternative function set
0X0001c	ALTFUNCCLR	Alternative function set
0X00020	INTENSET	Interrupt enable set
0X00024	INTENCLR	Interrupt enable clear
0X00028	INTTYPESET	Interrupt type set
0X0002c	INTTYPECLR	Interrupt type clear
0X00030	INTPOLSET	Interrupt polarity set
0X00034	INTPOLCLR	Interrupt polarity clear
0X00038	INTSTATUS	Interrupt status
0x00040	INTTYPE1SET	Interrupt type1 set
0x00044	INTTYPE1CLR	Interrupt type1 clear
0x01000-0x13FC	MASKBYTE0	Byte 0 masked access
0x01400-0x17FC	MASKBYTE1	Byte 1 masked access
0x01800-0x1BFC	MASKBYTE2	Byte 2 masked access
0x01c00-0x1FFC	MASKBYTE3	Byte 3 masked access
0x400E002C	GPIO_OE_CTRL	GPIO output enable control
0x400E0030	GPIO_OE_CTRL_1	GPIO output enable control
0x400E0034	GPIO_PU_CTRL	GPIO pullup control

0x400E0040	GPIO_ODA_CTRL	GPIO output
0x400E0054	GPIO_IE_CTRL	GPIO input enable control
0x400E00A0	GPIO_ODE_CTRL	GPIO open drain control
0x400E00A8	GPIO_PD_CTRL	GPIO pull down control
0x400E00C4	GPIO_DRV_CTRL_0	GPIO driver control register
0x400E00C8	GPIO_DRV_CTRL_1	GPIO driver control register
0x400E00CC	GPIO_DRV_CTRL_2	GPIO driver control register
0x400E00D0	GPIO_DRV_CTRL_3	GPIO driver control register

DATA address offset: 0x00000

Bit	R/W	Reset	Name	Description
31:0	RW	N/A	DATA	**Read** Sampled at pin. **Write** To data output register. Note: for GPIO1, only the low 8bit is valid

DATAOUT address offset: 0x00004

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	DATAOUT	Data output Register value: **Read** Current value of data output register. **Write** To data output register. Note: for GPIO1, only the low 8bit is valid

OUTENSET address offset: 0x00010

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	OUTENSET	Output enable set: **Write** **1** Set the output enable bit. **0** No effect. **Read back** **0** Indicates the signal direction as input. **1** Indicates the signal direction as output Note:for GPIO1, only the low 8bit is valid

OUTENCLR address offset: 0x00014

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	OUTENCLR	Output enable clear: **Write** **1** Clears the output enable bit. **0** No effect. **Read back** **0** Indicates the signal direction as input. **1** Indicates the signal direction as output Note: for GPIO1, only the low 8bit is valid

ALTFUNCSET address offset: 0x00018

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	ALTFUNCSET	Alternative function set: **Write** **1** Sets the ALTFUNC bit. **0** No effect. **Read back** **0** For I/O. **1** For an alternate function. Note: for GPIO1, only the low 8bit is valid

ALTFUNCCLR address offset: 0x0001C

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	ALTFUNCCLR	Alternative function clear: **Write** **1** Clears the ALTFUNC bit. **0** No effect. **Read back** **0** For I/O. **1** For an alternate function Note: for GPIO1, only the low 8bit is valid

INTENSET address offset: 0x00020

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	INTENSET	Interrupt enable set: **Write** **1** Sets the enable bit. **0** No effect. **Read back** **0** Interrupt disabled. **1** Interrupt enabled Note: for GPIO1, only the low 8bit is valid

INTENCLR address offset: 0x00024

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	INTENCLR	Interrupt enable clear: **Write** **1** Clear the enable bit. **0** No effect. **Read back** **0** Interrupt disabled. **1** Interrupt enabled Note:for GPIO1, only the low 8bit is valid

INTTYPESET address offset: 0x00028

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	INTTYPESET	Interrupt type set: **Write** **1** Sets the interrupt type bit. **0** No effect. **Read back** **0** For LOW or HIGH level. **1** For falling edge or rising edge Note: for GPIO1, only the low 8bit is valid

INTTYPECLR address offset: 0x0002c

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	INTTYPECLR	Interrupt type clear: **Write** **1** Clears the interrupt type bit. **0** No effect. **Read back** **0** For LOW or HIGH level. **1** For falling edge or rising edge. Note: for GPIO1, only the low 8bit is valid

INTPOLSET address offset: 0x00030

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	INTPOLSET	Polarity-level, edge IRQ configuration: **Write** **1** Sets the interrupt polarity bit. **0** No effect. **Read back** **0** For LOW level or falling edge. **1** For HIGH level or rising edge. Note: for GPIO1, only the low 8bit is valid

INTPOLCLR address offset: 0x00034

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	INTPOLCLR	Polarity-level, edge IRQ configuration: **Write** **1** Clears the interrupt polarity bit. **0** No effect. **Read back** **0** For LOW level or falling edge. **1** For HIGH level or rising edge. Note: for GPIO1, only the low 8bit is valid

INTSTATUS address offset: 0x00038

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	INTSTATUS	Write one to clear interrupt request: **Write** IRQ status clear Register. Write: **1** To clear the interrupt request. **0** No effect. **Read back** IRQ status Register Note: for GPIO1, only the low 8bit is valid

INTTYPE1SET address offset: 0x00040

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	INTTYPE1SET	Interrupt type1 set: **Write** **1** if the corresponding bit of INTTYPE SET is also 1, both falling edge and rising edge triggers interrupt; else no effect **0** No effect. **Read back** **0** double edge detect disable **1** double edge detect enable Note: for GPIO1, only the low 8bit is valid

INTTYPE1CLR address offset: 0x00044

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	INTTYPE1CLR	Interrupt type1 clear: **Write** **1** Clears the interrupt type1 bit. **0** No effect. **Read back** **0** double edge detect disable. **1** double edge detect enable. Note: for GPIO1, only the low 8bit is valid

MASKBYTE0 address offset: 0x01000-0x013FC

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
7:0	RW	0x0	MASKBYTE0	BYTE0 masked access. Bits[9:2] of the address value are used as enable bit mask for the access: **[7:0]** Data for BYTE0 access, with bits[9:2] of address value used as

				enable mask for each bit

MASKBYTE1 address offset: 0x01400-0x017FC

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
7:0	RW	0x0	MASKBYTE1	BYTE1 masked access. Bits[9:2] of the address value are used as enable bit mask for the access: **[7:0]** Data for BYTE1 access, with bits[9:2] of address value used as enable mask for each bit NOTE: this is only valid for GPIO0

MASKBYTE2 address offset: 0x01800-0x01BFC

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
7:0	RW	0x0	MASKBYTE2	BYTE2 masked access. Bits[9:2] of the address value are used as enable bit mask for the access: **[7:0]** Data for BYTE2 access, with bits[9:2] of address value used as enable mask for each bit NOTE: this is only valid for GPIO0

MASKBYTE3 address offset: 0x01C00-0x01FFC

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
7:0	RW	0x0	MASKBYTE3	BYTE3 masked access. Bits[9:2] of the address value are used as enable bit mask for the access: **[7:0]** Data for BYTE3 access, with bits[9:2] of address value used as enable mask for each bit. NOTE: this is only valid for GPIO0

GPIO_OE_CTRL address offset: 0x002c

Bit	R/W	Reset	Name	Description
31:0	RW	0xffffffff	gpio_oeb_reg[31:0]	Only valid when gpio_oeb_sel=1 GPIO[31:0] output enable control 0: output enable

GPIO_OE_CTRL_1 address offset: 0x0030

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved

0	RW	0x0	gpio_oeb_sel	1: gpio_oeb and gpio_out is controlled by reg, 0: gpio_oeb and gpio_out is controlled by hw or gpio_auto_latch_ctrl

GPIO_PU_CTRL address offset: 0x0034

Bit	R/W	Reset	Name	Description
31:0	RW	0x010	gpio_pu_reg[31:0]	GPIO[31:0] pull up control, high active

GPIO_ODA_CTRL address offset: 0x0040

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	gpio_oda_ctrl[31:0]	GPIO[31:0] output data (Only valid when gpio_oeb_sel=1)

GPIO_IE_CTRL address offset: 0x0054

Bit	R/W	Reset	Name	Description
31:0	RW	0x1f	gpio_ie_ctrl[31:0]	GPIO[31:0] input enable, high active

GPIO_ODE_CTRL address offset: 0x00a0

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	gpio_ode_reg[31:0]	gpio[31:0] open drain control, high active

GPIO_PD_CTRL address offset: 0x00a8

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	gpio_pd_reg[31:0]	gpio[31:0] pull down control, high active

GPIO_DRV_CTRL_0 address offset: 0x00c4

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	gpio_drv_ctrl_0[31:0]	GPIO driver strength

GPIO_DRV_CTRL_1 address offset: 0x00c8

Bit	R/W	Reset	Name	Description
31:6	N/A	0x0	N/A	reserved
5:0	RW	0x0	gpio_drv_ctrl_0[37:32]	GPIO driver strength

GPIO_DRV_CTRL_2 address offset: 0x00cc

Bit	R/W	Reset	Name	Description
31:0	RW	0xffffffff	gpio_drv_ctrl_1[31:0]	GPIO driver strength

GPIO_DRV_CTRL_3 address offset: 0x00d0

Bit	R/W	Reset	Name	Description
31:6	N/A	0x0	N/A	reserved
5:0	RW	0x3f	gpio_drv_ctrl_1[37:32]	GPIO driver strength

8.5 I2S

8.5.1 Introduction

The Onmicro_i2s is a configurable, synthesizable, and programmable component designed to be used in systems that process digital audio signals, such as:

- A/D and D/A converters
- digital signal processors
- error correction for compact disc and digital recording
- digital filters
- digital input/output interfaces

The Inter-IC Sound (I2S) Bus is a simple three-wire serial bus protocol developed by Philips to transfer stereo audio data. The bus only handles the transfer of audio data; hence control and subcoding signals need to be transferred separately using a different bus protocol (such as I2C).

8.5.2 Main Features

Onmicro_i2s has the following features:

- APB data bus widths of 8, 16, and 32 bits
- I2S transmitter and/or receiver based on the Philips I2S serial protocol
- Configurable number of stereo channels (up to 4) for both transmitter and receiver
- Full duplex communication due to the independence of transmitter and receiver
- Asynchronous clocking of APB bus and I2S sclk
- Master or slave mode of operation
- Audio data resolutions of 12, 16, 20, 24, and 32 bits
- External sclk gating and enable signals
- Configurable FIFO depth of 2, 4, 8, 16 bits
- Configurable support for programmable DMA registers
- Programmable FIFO thresholds
- Component parameters for configurable software driver support

8.5.3 Function Description

The I2S bus can only handle audio data transmissions; subcoding and controls are handled by another device, such as an I2C. The I2S protocol requires a minimum of three wires—data (sd), word select (ws), and serial clock (sclk)—keeping the design simple and the pin count minimal. However, Onmicro_i2s can be configured to have up to four channels for transmit and receive operations, making the maximum configured wire count

10.

The component also can be configured to operate as either a master or a slave (the default mode). When configured as a master, Onmicro_i2s initializes the word select (ws_out signal) and supplies the clock gating(sclk_gate) and clock enabling (sclk_en) signals. When operating as a slave, Onmicro_i2s responds to externally generated sclk and ws signals.

Whether configured as a master or slave, an external sclk and an inverted version of sclk need to besupplied to the device via input signals sclk and sclk_n.Onmicro_i2s supports the standard I2S frame format for transmitting and receiving data —the MSB of aword is sent one sclk cycle after a word select change.

8.5.3.1.Onmicro_i2s Enable

You must enable the Onmicro_i2s component before any data can be received or transmitted into the FIFOs.

To enable the component, set the I2S Enable (IEN) bit of the I2S Enable Register (IER) to 1. When you disable.The device, it acts as a global disable. To disable Onmicro_i2s, set IER[0] to 0.

After disable, the following events occur:

- TX and RX FIFOs are cleared, and read/write pointers are reset;
- Any data in the process of being transmitted or received is lost;
- All other programmable enables (such as transmitter/receiver block enables and individual TX/RX channel enables) in the component are overridden;
- Generation of master mode clock signals sclk_en, ws_out and sclk_gate are disabled (for instance,they are held low).

When Onmicro_i2s is enabled and configured as a master, the device always starts in the left stereo data cycle (ws = 0), and one sclk cycle later transitions to the right stereo data cycle (ws = 1). This allows for half a frame of sclks to write data to the TX FIFOs and to ensure that any connected slave receivers do not miss the start of the data frame (for instance, the ws 1-to-0 transition) once the sclk restarts. (When Onmicro_i2s is configured as a slave, ws is externally supplied.) On reset, the IER[0] is set to 0 (disable).

8.5.3.2.Onmicro_i2s as Transmitter

The Onmicro_i2s component can be configured to support up to four stereo I2S transmit (TX) channels. These channels can operate in either master or slave mode. By default, Onmicro_i2s is configured in slave mode. Stereo data pairs (such as, left and right audio data) written to a TX channel via the APB bus are shifted out serially on the appropriate serial data out line (sdo0, sdo1, sdo2, sdo3). The shifting is timed with respect to the serial clock (sclk) and the word select line (ws).

The instantiation of the I2S transmitter block and the number of TX channels is determined by the two configuration parameters: Transmitter Block Enabled (I2S_TRANSMITTER_BLOCK) and Number of Transmit Channels (I2S_TX_CHANNELS), respectively. By default,Onmicro_i2s is configured with one transmit channel.The following figure illustrates the basic usage flow for Onmicro_apb_i2s when it acts as a transmitter.

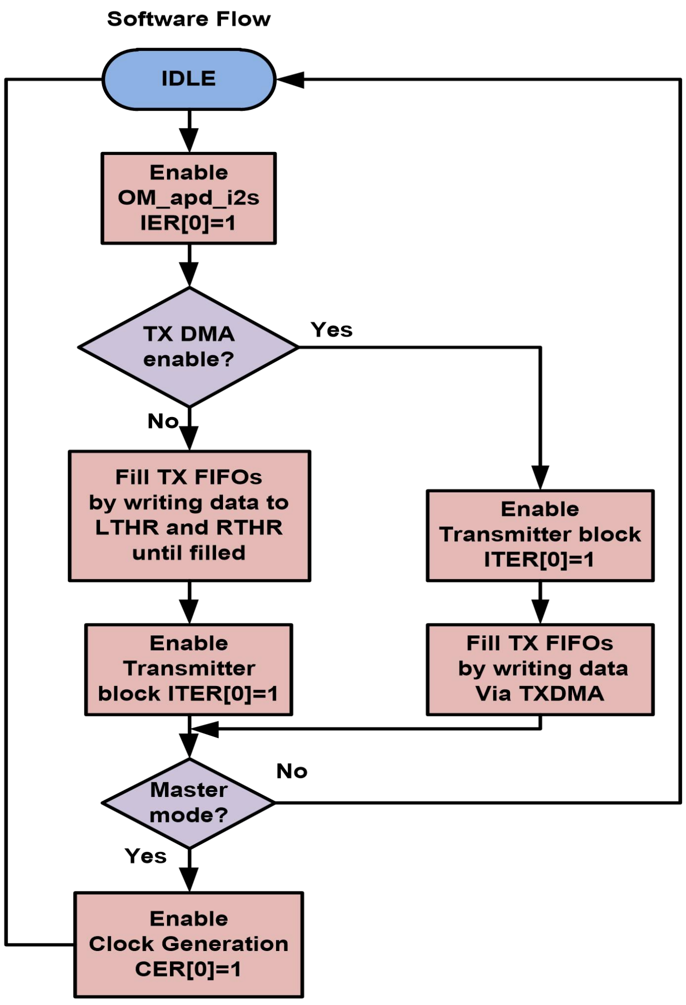

Software Flow
IDLE
Enable OM_apd_i2s IER[0]=1
TX DMA enable?
Yes
No
Fill TX FIFOs by writing data to LTHR and RTHR until filled
Enable Transmitter block ITER[0]=1
Enable Transmitter block ITER[0]=1
Fill TX FIFOs by writing data Via TXDMA
Master mode?
No
Yes
Enable Clock Generation CER[0]=1

Figure 8.12 Basic Usage Flow –Onmicro_apb_i2s as Transmitter

8.5.3.3.Transmitter Block Enable

The Transmitter Block Enable (TXEN) bit of the I2S Transmitter Enable Register (ITER) globally turns on and off all of the configured TX channels.To enable the transmitter block, set ITER[0] to 1.To disable the block, set ITER[0] to 0.

When the transmitter block is disabled, the following events occur:

- Outgoing data is lost and the channel outputs are held low;
- Data in the TX FIFOs are preserved and the FIFOs can be written to;
- Any previous programming (like changes in word size, threshold levels, and so on) of the TX channels is preserved;
- Any individual TX channel enables are overridden.

When the transmitter block is enabled, if there is data in the TX FIFOs, the channel resumes transmission on the next left stereo data cycle (such as when the ws line goes low).

When the block is disabled, you can perform any of the following procedures:

- Program (or further program) TX channel registers
- Flush the TX FIFOs by programming the Transmitter FIFOs Reset bit of the Transmitter FIFO Flush Register (TXFFR[0] = 1)
- Flush an individual channel's TX FIFO by programming the Transmit Channel FIFO Reset(TXCHFR) bit of the Transmit FIFO Flush Register (TFFx[0] = 1, where x is the channel number)

On reset, the ITER[0] is set to 0 (disable) .

8.5.3.4.Transmit Channel Enable

Each transmit channel has its own enable/disable that can be set independently of the other channels to allow the reprogramming of a channel and to flush the channel's TX FIFOs while other TX channels are transmitting. This enable/disable is controlled by bit 0 of the Transmitter Enable Register (TERx, where x is the channel number). For example, to enable TX Channel 1, write a 1 to TER1[0]. To disable this channel, write a 0 to TER1[0].

When a TX channel is disabled, the following occurs:

- Outgoing stereo data is lost;
- Channel output is held low;
- Data in the TX FIFO is preserved, and the FIFO can be written to;
- Any previous programming of the TX channel's registers is preserved, and the registers can be further reprogrammed.

When a TX channel is disabled, you can flush the channel's TX FIFO by programming the Transmit Channel FIFO Reset (TXCHFR) bit of the Transmit FIFO Flush (TFFx[0] = 1, where x is the channel number). When the TX channel is enabled, if there is data in the TX FIFO, the channel resumes transmission on the next left stereo data cycle (such as, when the ws line goes low).On reset, the TFFx[0] is set to 1 (enable).

8.5.3.5.Transmit Channel Audio Data Resolution

Each TX channel is initially configured with a maximum audio data resolution as set by the Maximum Audio Resolution parameter (I2S_TX_WORDSIZE_x, where x is the channel number). A TX channel can be reprogrammed during operation to any supported audio data resolution that is less than I2S_TX_WORDSIZE_x.

For example, if the TX Channel 2 is initially configured with a 32-bit audio resolution, it can be programmed to support a resolution of 12, 16, 20, 24, or 32 bits. However, if TX Channel 3 is initially configured with a 20-bit audio resolution, it can only be programmed to support resolution of 12, 16, or 20 bits. Any other resolution values are considered invalid. Furthermore, if the channel is programmed with an invalid audio resolution, the TX channel defaults to I2S_TX_WORDSIZE_x.

Reprogramming of the audio resolution ensures that the MSB of the data is still transmitted first if the resolution of the data to be sent is reduced. Changes to the resolution are programmed via the Word Length (WLEN) bits of the Transmitter

Configuration Registers (TCRx[2:0], where x is the channel number). The channel must be disabled prior to any resolution changes.On reset or if an invalid resolution is selected, the TX channel's audio data resolution defaults back to the initial parameter setting of I2S_TX_WORDSIZE_x.

8.5.3.6.Transmit Channel FIFOs

Each Transmit Channel has two FIFO banks for left and right stereo data. The FIFOs can be configured with depths of 2, 4, 8, and 16 bits as determined by the I2S_FIFO_DEPTH_GLOBAL. The FIFO width is determined by the "Maximum Audio Resolution – Transmit Channel X" parameter (I2S_TX_WORDSIZE_x, where x is the channel number).

There are several ways to clear the TX FIFOs and reset the read/write pointers as described as follows;

- on reset
- by disabling Onmicro_i2s (IER[0] = 0)
- by flushing the transmitter block (TXFFR[0] = 1)
- by flushing an individual TX channel (TFFx[0] = 1, where x is the channel number)

You must disable the transmitter block/channel before the transmitter block and individual channel FIFO can be flushed.

The TX FIFO Empty Threshold Trigger Level parameter (I2S_TX_FIFO_THRE_x, where x is the channel number) sets the default trigger threshold level for the TX FIFO. The trigger level can be set to any value in the range of 0 to I2S_TX_FIFO_x – 1. When this level is reached, a transmit channel empty interrupt is generated. This level can be reprogrammed during operation by writing to the Transmit Channel Empty Trigger (TXCHET) bits of the Transmit FIFO Configuration Register (TFCRx[3:0], where x is the channel number).

You must disable the TX channel prior to changing the trigger level.

8.5.3.7.Transmit Channel Interrupts

All interrupts in Onmicro_i2s can be configured as active low or active high by setting the "Polarity of Interrupt Signals is Active High?" parameter. Each TX channel generates two interrupts: TX FIFO Empty and Data Overrun.

- TX FIFO Empty interrupt – This interrupt is asserted when the empty trigger threshold level for the TX FIFO is reached. When this interrupt is included on the I/O, it appears on the outputs tx_emp_x_intr (where x is the channel number). A TX FIFO Empty interrupt is cleared by writing data to the TX FIFO to bring its level above the empty trigger threshold level for the channel.
- Data Overrun interrupt –This interrupt is asserted when an attempt is made to write to a full TX FIFO (any data being written is lost while data in the FIFO is preserved). When this interrupt is included on the I/O, it appears on the outputs tx_or_x_intr (where x is the channel number). A Data Overrun interrupt is cleared

by reading the Transmit Channel Overrun (TXCHO) bit [0] of the Transmit Overrun Register (TORx, where x is the channel number).

The interrupt status of any TX channel can be determined by polling the Interrupt Status Register (ISRx, where x is the channel number). The TXFE bit [4] indicates the status of the TX FIFO Empty interrupt, while the TXFO bit [5] indicates the status of the Data Overrun interrupt.

Both the TX FIFO Empty and Data Overrun interrupts can be masked off by writing a 1 in the Transmit Empty Mask (TXFEM) and Transmit Overrun Mask (TXFOM) bits of the Interrupt Mask Register (IMRx, where x is the channel number), respectively. This prevents the interrupts from driving their output lines, however, the ISRx always shows the current status of the interrupts regardless of any masking.

The setting of the “Multiple Interrupt Output Ports Present?” configuration parameter (I2S_INTERRUPT_SIGNALS) affects whether these two interrupts are included on Onmicro_i2s’s interface. The following table shows the specific interrupt signal to appear on the I/O according to the setting of I2S_INTERRUPT_SIGNALS.

Interrupt	Signal(s) on I/O
I2S_INTERRUPT_SIGNALS = True (multiple interrupts on I/O)	
TX FIFO Empty	tx_emp_x_intr
Data Overrun	tx_or_x_intr
I2S_INTERRUPT_SIGNALS = False (shared single interrupt on I/O)	
combined interrupt signal	intr

Table 8.3 TX Channel Interrupt Signals on I/O

8.5.3.8.Writing to a Transmit Channel

The stereo data pairs to be transmitted by a TX channel are written to the TX FIFOs via the Left Transmit Holding Register (LTHRx, where x is the channel number) and the Right Transmit Holding Register (RTHRx, where x is the channel number). All stereo data pairs must be written using the following twostage process:

- Write left stereo data to LTHRx
- Write right stereo data to RTHRx.

Note: You must write stereo data to the device in this order, otherwise, the interrupt and status lines values will be invalid, and the left/right stereo pairs might be transmitted out of sync.

When TX DMA is enabled (I2S_TX_DMA = 1), data to be transmitted by TX channels are written to the TX FIFOs via the TXDMA register rather than through LTHRx and RTHRx. Data is written cyclically through all enabled TX channels starting from the lowest-numbered enabled channel. After a stereo data pair is transmitted, the component will point to the next enabled channel.

The following example describes the behavior of the TXDMA register for a component that has been configured with four Transmit channels, where Channels 0 and 2 are enabled. Order of transmitted data:

- Ch0 — Left Data

- Ch0 — Right Data
- Ch2 — Left Data
- Ch2 — Right Data
- Ch0 — Left Data
- Ch0 — Right Data, and so on

The RTXDMA register resets TXMDA to the lowest-enabled Channel. The RTXDMA register can be written to at any stage of the TXDMA transmit cycle; however, it has no effect when the component is in the middle of a stereo pair transmit.

The following example describes the operation of this register for a system with four Transmit channels, where all the channels are enabled.

Order of transmitted data

- Ch0 — Left Data
- Ch0 — Right Data
- RTXDMA Reset
- Ch0 — Left Data
- Ch0 — Right Data
- Ch1 — Left Data
- RTXDMA Reset — No effect (read not complete)
- Ch1 — Right Data, etc.
- Ch2 — Left Data
- Ch2 — Right Data
- RTXDMA Reset
- Ch0 — Left Data
- Ch0 — Right Dat

When Onmicro_i2s is enabled, if the TX FIFO is empty and data is not written to the FIFOs before the next left cycle, the channel outputs zeros for a full frame (left and right cycle). Transmission only commences if there is data in the TX FIFO prior to the transition to the left data cycle. In other words, if the start of the frame is missed, the channel output idles until the next available frame.

If the APB bus width is less than the configured or programmed audio resolution, multiple writes are required to write each stereo word. For example,if the Maximum Audio Resolution for Transmit Channel 1 is 20 (I2S_TX_WORDSIZE_1 = 20) and APB_DATA_WIDTH = 8, then three writes per register are required to write data to LTHR1 and RTHR1. However, if the audio resolution of the channel is reprogrammed to be 12 bits, then only two writes per register are required (the third write is ignored by the device). Thus, if the audio resolution is reduced, there is no need to perform the extra write to pad the data with leading zeros.

Note: Data should only be written to the FIFO when it is not full. Any attempt to write to a full FIFO results in that data being lost and a Data Overrun interrupt being generated.

8.5.3.9.Onmicro_i2s as Receiver

Onmicro_i2s can be configured to support up to four stereo I2S receive (RX) channels. These channels can operate in either master or slave mode. By default, Onmicro_i2s is configured in slave mode. Stereo data pairs (such as, left and right audio data) are received serially from a data input line (sdi0, sdi1, sdi2, sdi3). These data words are stored in RX FIFOs until they are read via the APB bus. The receiving is timed with respect to the serial clock (sclk) and the word select line (ws). The instantiation of the receiver block and the number of RX channels is determined by two configuration parameters: Receiver Block Enabled (I2S_RECEIVER_BLOCK) and Number of Receive Channels (I2S_RX_CHANNELS), respectively. By default, Onmicro_i2s is configured with one receive channel.The following figure illustrates the basic usage flow for Onmicro_apb_i2s when it acts as a receiver.

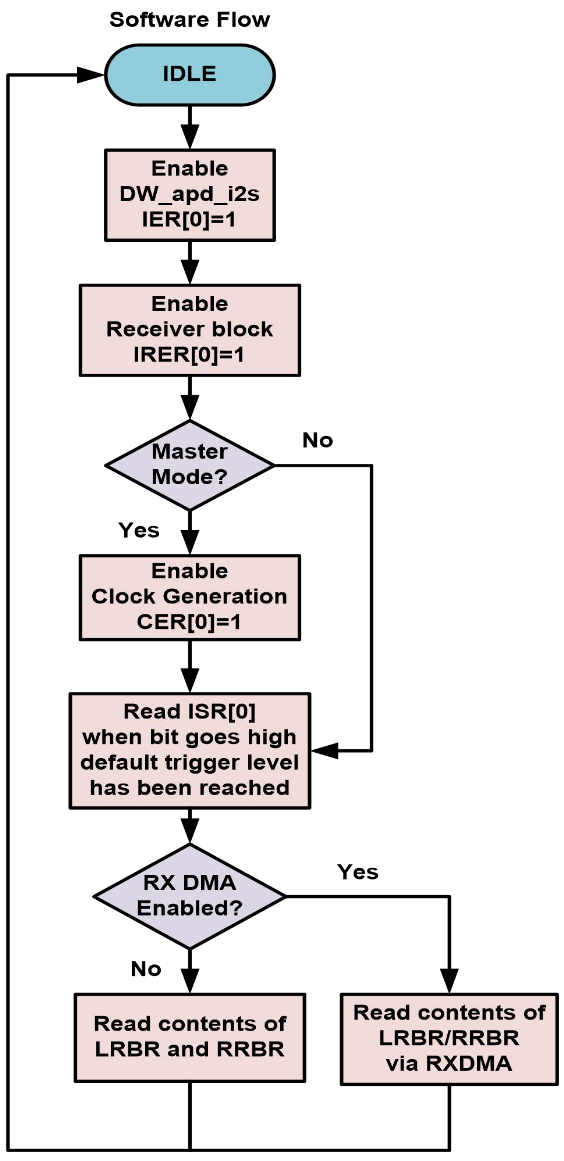

Software Flow
IDLE
Enable DW_apd_i2s IER[0]=1
Enable Receiver block IRER[0]=1
Master Mode?
No
Yes
Enable Clock Generation CER[0]=1
Read ISR[0] when bit goes high default trigger level has been reached
RX DMA Enabled?
Yes
No
Read contents of LRBR and RRBR
Read contents of LRBR/RRBR via RXDMA

Figure 8.13 Basic Usage Flow – Onmicro_apb_i2s as Receiver

Receiver Block Enable

The Receiver Block Enable (RXEN) bit of the I2S Receiver Enable Register (IRER) enables/disables all configured RX channels. To enable the receiver block, set IRER[0] to '1'. To disable the block, set this bit to '0'.

When the receiver block is disabled, the following events occur:

- Incoming data is lost;
- Data in the RX FIFOs is preserved and the FIFOs can be read;
- Any previous programming (such as changes in word size, threshold levels, and so on) of the RX channels is preserved;
- Any individual RX channel enable is overridden. Enabling the channel resumes receiving on the next left stereo data cycle (for instance, when ws goes low).

When the block is disabled, you can perform any of the following procedures:

- Program (or further program) the RX channel registers;
- Flush the RX FIFOs by programming the Receiver FIFOs Reset (RXFR) bit of the Receiver FIFO Flush Register (RXFFR[0] = 1);
- Flush an individual channel's RX FIFO by programming the Receive Channel FIFO Reset (RXCHFR) bit of the Receive FIFO Flush Register (RFFx [0] = 1, where x is the channel number).

On reset, IRER[0] is set to 0 (disable).

Receive Channel Enable

Each RX channel has its own enable/disable that can be set independently of the other channels to allow programming of the channel and to clear the channel's RX FIFO while other RX channels are still receiving data. This enable/disable is controlled by bit 0 of the Receiver Enable Register (RERx[0], where x is the channel number). For example, to enable RX Channel 1, write a 1 to RER1[0]. To disable this channel, write a 0 to RER1[0].

When the RX channel is disabled, the following occurs:

- Incoming data is lost;
- Data in the RX FIFO is preserved;
- FIFO can be read;
- Previous programming of the RX channel is preserved;
- RX channel can be further programmed.

When the RX channel or block is disabled, you can flush the channel's RX FIFO by writing 1 in bit 0 of the Receive FIFO Flush Register (RFFx, where x is the channel number). When the channel is enabled, it resumes receiving on the next left stereo data cycle (for instance. when ws line goes low).

On reset, the RFFx[0] is set to 1 (enable).

Receive Channel FIFOs

Each Receive Channel has two FIFO banks for left and right stereo data. The FIFOs can be configured with depths of 2, 4, 8, or 16 bits as determined by the "FIFO Depth for RX and TX Channels?" parameter (I2S_FIFO_DEPTH_GLOBAL, see page 44). The FIFO width is determined by the configured maximum data resolution for the channel (I2S_RX_WORDSIZE_x, where x is the channel number).

The RX FIFOs can be cleared and the read/write pointers reset in a number ways, as described as follows:

- on reset
- by disabling Onmicro_i2s (IER[0] = 0)
- by flushing the receiver block (RXFFR[0] = 1)
- by flushing an individual RX channel (RFFx[0] = 1, where x is the channel number)

Before you flush the receiver block or individual channels, you must disable the receiver block or channel.

The RX FIFO Data Available Level parameter (I2S_RX_FIFO_THRE_x, where x is the channel number) sets the default data available trigger level for the RX FIFO. When this level is reached, a RX channel data available interrupt is generated. The valid values are 0 to FIFO_DEPTH–1, which correspond to trigger levels of 1 to FIFO_DEPTH (for example, Trigger Level = Configured Value + 1). This level can be reprogrammed during operation via the Receive Channel Data Trigger (RXCHDT) bits of the Receive FIFO Configuration Register (RFCRx[3:0], where x is the channel number). The RX channel needs to be disabled prior to any changes in the trigger level.

Receive Channel Interrupts

All interrupts in Onmicro_i2s can be configured as active low or active high via the “Polarity of Interrupts Signals is Active High?” Each RX channel generates two interrupts: RX FIFO Data Available and Data Overrun.

- RX FIFO Data Available interrupt – This interrupt is asserted when the trigger level for the RX FIFO is reached. When this interrupt is included on the I/O, it appears on the outputs rx_da_x_intr (where x is the channel number). This interrupt is cleared by reading data from the RX FIFO until its level drops below the data available trigger level for the channel.
- Data Overrun interrupt – This interrupt is asserted when an attempt is made to write received data to a full RX FIFO (any data being written is lost while data in the FIFO is preserved). When this interrupt is included on the I/O, it appears on the outputs rx_or_x_intr (where x is the channel number). This interrupt is cleared by reading the Receive Channel Overrun (RXCHO) bit [0] of the Receive Overrun Register (RORx, where x is the channel number).

The interrupt status of any RX channel can be determined by polling the Interrupt Status Register (ISRx, where x is the channel number). The RXDA bit [0] indicates the status of the RX FIFO Data Available interrupt; the RXFO bit [1] indicates the status of the RX FIFO Data Overrun interrupt.

Both the Receive Empty Threshold and Data Overrun interrupts can be masked by writing a 1 in the Receive Empty Threshold Mask (RDM) and Receive Overrun Mask (ROM) bits of the Interrupt Mask Register (IMRx, where x is the channel number), respectively. This prevents the interrupts from driving their output lines, however, the ISRx always shows the current status of the interrupts regardless of any masking.

The setting of the “Multiple Interrupt Output Ports Present?” configuration parameter (I2S_INTERRUPT_SIGNALS) affects whether these two interrupts are included on Onmicro_i2s’s interface.The following table shows the specific interrupt signal to appear on the I/O according to the setting ofI2S_INTERRUPT_SIGNALS.

Interrupt	Signal(s) on I/O
I2S_INTERRUPT_SIGNALS = True (multiple interrupts on I/O)	
RX FIFO Data Available	tx_da_x_intr
Data Overrun	tx_or_x_intr
I2S_INTERRUPT_SIGNALS = False (shared single interrupt on I/O)	

combined interrupt signal	intr

Table 8.4 RX Channel Interrupt Signals on I/O

Reading from a Receive Channel

The stereo data pairs received by a RX channel are written to the left and right RX FIFOs. These FIFOs can be read via the Left Receive Buffer Register (LRBRx, where x is the channel number) and the Right Receive Buffer Register (RRBRx, where x is the channel number). All stereo data pairs must be read using the following two-stage process:

- Read the left stereo data from LRBRx.
- Read the right stereo data from RRBRx.

When RX DMA is enabled (I2S_RX_DMA = 1), data can be read from RX FIFOs via the RXDMA register rather than through LRBRx and RRBRx. The RXDMA register cyclically accesses the RX FIFOs of all enabled RX channels similarly to the TXDMA register.

The RRXDMA register resets the RXDMA read cycle. This register provides the same functionality as the RTXDMA register, but targets RXDMA instead.

Receive Channel Audio Data Resolution

Each RX channel is initially configured with a maximum audio data resolution as set by the “Max Audio Resolution – Receive Channel X” parameter (I2S_RX_WORDSIZE_x, where x is the channel number). An RX channel can be programmed during operation to any supported audio data resolution that is less than I2S_RX_WORDSIZE_x.

For example, if the RX Channel 2 is initially configured with a 32-bit audio resolution, it can be programmed to support resolutions of 12 16, 20, 24, or 32 bits. However, if RX Channel 3 is initially configured with a 20- bit audio resolution, it can only be programmed to support resolutions of 12, 16, or 20 bits. Any other resolution values are considered invalid. Additionally, if the channel is programmed with an invalid audio resolution, the RX channel defaults to I2S_RX_WORDSIZE_x.

This programming ensures that the LSB of the received data is placed in the LSB position of the RX FIFO if the resolution of the data being received is reduced. Changes to the resolution are programmed via the Word Length (WLEN) bits of the Receive Configuration registers (RCRx[3:0], where x is the channel number). The channel must be disabled prior to any resolution changes.

The RX channel also supports unknown data resolutions. If the received word is greater than the configured channel resolution, the least significant bits are ignored. If the received word is less than the configured/programmed channel resolution, the least significant bits are padded with zeros.

On reset or if an invalid resolution is selected, the RX channel’s audio data resolution defaults back to the initial parameter setting of I2S_RX_WORDSIZE_x.

8.5.3.10.Clock Generation (Master Mode)

The clock generation block is only instantiated when the “Is an I2S Master?” parameter (I2S_MODE_EN) is checked (“True”). When Onmicro_i2s is a master, it initializes the word select signal (ws_out) and supplies the clock gating (sclk_gate) and clock enabling (sclk_en) signals. Additionally, the sclk and sclk_n inputs are included on the I/O regardless of whether the device is a master or a slave.

Clock Generation Enable

The Clock Generation Enable (CLKEN) bit of the Clock Enable Register (CER) enables and disables the master mode clock signals: ws_out, sclk_en, and sclk_gate. To enable these signals, set CER[0] to 1; to disable them, set this bit to 0, in which case ws_out is held low (ws_out = 0).

When the CLKEN bit is disabled, any incoming or outgoing data is lost. However, data already in the RX and TX FIFOs are preserved. After this bit is enabled, transmission recommences at the start of the next stereo frame.

On enabling CER[0], ws_out always starts in the left stereo data cycle (ws_out = 0). One sclk cycle later, it transitions to the right stereo data cycle (ws_out = 1); hence—a 0-to-1 transition. This allows for half a frame of sclks to write data to the TX FIFOs and ensures that any connected slave receivers do not miss the start of the data frame (the ws 1-to-0 transition) once the sclk restarts.

To further explain this behavior, the ws transitions—0-to-1 and 1-to-0—are used by the device to clock the start of the right or left stereo data cycle. The transition 1-to-0 indicates the start of a new stereo data pair. Because Onmicro_i2s is simple in terms of control, for every 1-to-0 transition, the device sends the next entry in its FIFO (similarly, the Receiver assumes new data is being sent and starts receiving). On enabling CER[0], the device starts with ws = 0 for one cycle and then transitions 0-to-1. This allows connected devices to clock off the transition and determine which cycle they are in. However, because it's not a 1-to-0 transition, the devices do not have to start TX or RX with potential garbage (assuming FIFOs are empty prior to enable). Additionally, the transmission ensures that after a clk_en, there is ample time to configure TX or RX and to input some data for transmission before the start of the next frame.

The signal sclk_en is provided that can be AND'd with sclk to disable the clock when the master device is disabled.

Word Select Generation

When Onmicro_i2s is configured as a master, the number of sclk cycles during which ws_out is held high or low is determined by the "Has a Word Select Length of?" parameter. However, this setting can be reprogrammed during operation of the component by setting the Word Select Size (WSS) bits [4:3] of the Clock Configuration Register (CCR). You must disable the Clock Generation block (CER[0] = 0) before you can change the word select size .The Onmicro_apb_i2s supports 16, 24, or 32 sclk cycles per left/right ws cycle as illustrated in the following figure.

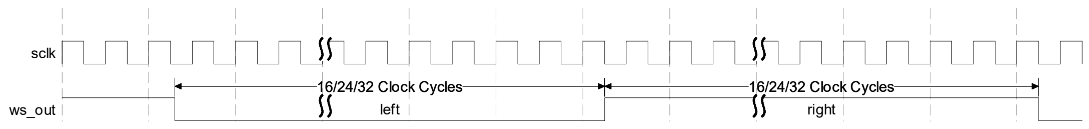


Figure 8.14 Number of SCLK Cycles Per Left/Right WS Cycle

SCLK Gating

When Onmicro_i2s is configured as a master and the audio data resolution of the receive and transmit channels is less than the current word select size, the sclk can be gated off for the remainder of the left/right cycle, as illustrated in the following figure. This gating is determined by the “Has a Serial Clock Gating of?” parameter). However, this can be reprogrammed during operation of the component by setting the SCLK Gating (SCLKG) bits [2:0] of the Clock Configuration Register (CCR).

The Clock Generation Block must be disabled prior to any changes to sclk gating value. Since sclk_gate is 1 during the cycles that are to be gated off, the actual gating of sclk needs to be done externally by AND'ing the inverse of the generated sclk_gate signal with sclk.

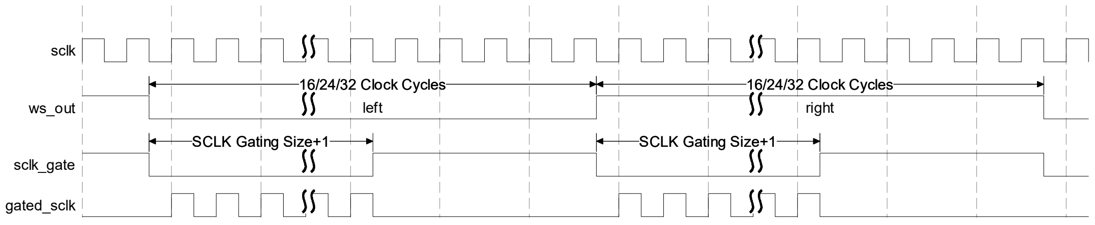


Figure 8.15 Gating of SCLK

8.5.4 I2S_TX Register Map

Offset	Name	Description
0x00000	IER	Onmicro_i2s Enable Register Reset Value: 0x0
0X00008	ITER	I2S Transmitter Block Enable Register Reset Value: 0x0
0x0000c	CER	Clock Enable Register Reset Value: 0x0
0x00010	CCR	Clock Configuration Register Reset Value: Configuration dependent
0X00018	TXFFR	Transmitter Block FIFO Register Reset Value: 0x0
0X00020	LTHR0	Left Transmit Holding Register 0 Reset Value: 0x0
0X00024	RTHR0	Right Transmit Holding Register 0 Reset Value: 0x0
0X0002C	TER0	Transmit Enable Register 0 Reset Value: 0x1
0X00034	TCR0	Transmit Configuration Register 0 Reset Value: Configuration Dependent
0X00038	ISR0	Interrupt Status Register 0 Reset Value: 0x10
0X0003C	IMR0	Interrupt Mask Register 0 Reset Value: 0x33
0X00044	TOR0	Transmit Overrun Register 0 Reset Value: 0x0

0X0004C	TFCR0	Transmit FIFO Configuration Register 0 Reset Value: Configuration Dependent
0X00054	TFF0	Transmit FIFO Flush 0 Reset Value: 0x0
0X00060	LTHR1	Left Transmit Holding Register 1 Reset Value: 0x0
0X00064	RTHR1	Right Transmit Holding Register 1 Reset Value: 0x0
0X0006C	TER1	Transmit Enable Register 1 Reset Value: 0x1
0X00074	TCR1	Transmit Configuration Register1 Reset Value: Configuration Dependent
0X00078	ISR1	Interrupt Status Register 1 Reset Value: 0x10
0X0007C	IMR1	Interrupt Mask Register 1 Reset Value: 0x33
0X00084	TOR1	Transmit Overrun Register 1 Reset Value: 0x0
0X0008C	TFCR1	Transmit FIFO Configuration Register 1 Reset Value: Configuration Dependent
0X00094	TFF1	Transmit FIFO Flush 1 Reset Value: 0x0
0X000A0	LTHR	Left Transmit Holding Register 2 Reset Value: 0x0
0X000A4	RTHR2	Right Transmit Holding Register 2 Reset Value: 0x0
0X000AC	TER2	Transmit Enable Register 2 Reset Value: 0x1
0X000B4	TCR2	Transmit Configuration Register 2 Reset Value: Configuration Dependent
0X000B8	ISR2	Interrupt Status Register 2 Reset Value: 0x10
0X000BC	IMR2	Interrupt Mask Register 2 Reset Value: 0x33
0X000C4	TOR2	Transmit Overrun Register 2 Reset Value: 0x0
0X000CC	TFCR2	Transmit FIFO Configuration Register 2 Reset Value: Configuration Dependent
0X000D4	TFF2	Transmit FIFO Flush 2 Reset Value: 0x0
0X000E0	LTHR3	Left Transmit Holding Register 3 Reset Value: 0x0
0X000E4	RTHR3	Right Transmit Holding Register 3

		Reset Value: 0x0
0X000EC	TER3	Transmit Enable Register 3 Reset Value: 0x1
0X000F4	TCR3	Transmit Configuration Register 3 Reset Value: Configuration Dependent
0X000F8	ISR3	Interrupt Status Register 3 Reset Value: 0x10
0X000FC	IMR3	Interrupt Mask Register 3 Reset Value: 0x33
0X00104	TOR3	Transmit Overrun Register 3 Reset Value: 0x0
0X0010C	TFCR3	Transmit FIFO Configuration Register 3 Reset Value: Configuration Dependent
0X00114	TFF3	Transmit FIFO Flush 3 Reset Value: 0x0
0X001C8	TXDMA	Transmitter Block DMA Register Reset Value: 0x00
0X001CC	RTXDMA	Reset Transmitter Block DMA Register Reset Value: 0x00
0x01F0	I2S_COMP_PARAM_2	Component Parameter 2 Register Reset Value: Reset value depends on user configuration. For more information, refer to
0x01F4	I2S_COMP_PARAM_1	Component Parameter 1 Register Reset Value: Reset value depends on user configuration. For more information, refer to
0x01F8	I2S_COMP_VERSION	Component Version ID Reset Value: See the releases table in the
0x01FC	I2S_COMP_TYPE	Design Ware Component Type Reset Value: 0x445701a0

IER address offset: 0x0000

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	RW	0x0	SINGLE_EN	Single track enable 1: enable 0: diable
3:1	N/A	0x0	N/A	reserved
0	RW	0x0	IEN	I2S global enable, a disable on this bit overrides any other block enables and flushes all FIFOs. 1: enable 0: disable

ITER address offset: 0x0008

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	TXEN	I2S Transmitter enable 1: enable transmitter 0: disable transmitter

TXFFR address offset: 0x0018

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	W	0x0	TXFFR	Transmitter FIFO Reset. Writing a 1 to this register flushes all the TX FIFOs (this is a self clearing bit). The Transmitter Block must be disabled prior to writing this bit.

LTHR address offset: 0x0020

Bit	R/W	Reset	Name	Description
31:0	W	0x0	LTHR	The left stereo data to be transmitted serially through the transmit output is written through this register.

RTHR address offset: 0x0024

Bit	R/W	Reset	Name	Description
31:0	W	0x0	RTHR	The right stereo data to be transmitted serially through the transmit output is written through this register.

TER address offset: 0x002C

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	TXCHEN	Transmit channel enable. On enable, the channel begins transmitting on the next left stereo cycle. 0: Disable 1: Enable

TCR address offset: 0x0034

Bit	R/W	Reset	Name	Description
31:3	N/A	0x0	N/A	reserved
2:0	RW	0x0	WLEN	These bits are used to program the data resolution of the transmitter and ensures the MSB of the data is

				transmitted first. 000 = Ignore word length 001 = 12 bit resolution 010 = 16 bit resolution 011 = 20 bit resolution 100 = 24 bit resolution 101 = 32 bit resolution

ISR address offset: 0x0038

Bit	R/W	Reset	Name	Description
31:6	N/A	0x0	N/A	reserved
5	R	0x0	TXFO	Status of Data Overrun interrupt for the TX channel. Attempt to write to full TX FIFO. 0: TX FIFO write valid 1: TX FIFO write overrun
4	R	0x0	TXFE	Status of Transmit Empty Trigger interrupt. TX FIFO is empty. 1: trigger level reached 0: trigger level not reached
3:0	N/A	0x0	N/A	reserved

IMR address offset: 0x003C

Bit	R/W	Reset	Name	Description
31:6	N/A	0x0	N/A	reserved
5	RW	0x1	TXFOM	Masks TX FIFO Overrun interrupt. 1: masks interrupt 0: unmasks interrupt
4	RW	0x1	TXFEM	Masks TX FIFO Empty interrupt. 1: masks interrupt 0: unmasks interrupt
3:0	N/A	0x0	N/A	reserved

TOR address offset: 0x0044

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	TXCHO	Read this bit to clear the TX FIFO Data Overrun interrupt. 0: TX FIFO write valid 1: TX FIFO write overrun

TFCR address offset: 0x004C

Bit	R/W	Reset	Name	Description
31:4	N/A	0x0	N/A	reserved

3:0	RW	0x0	TXCHET	Transmit Channel Empty Trigger. These bits program the trigger level in the TX FIFO at which the Empty Threshold Reached Interrupt is generated. Trigger Level = TXCHET

TFF address offset: 0x0054

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	W	0x0	TXCHFR	Transmit Channel FIFO Reset. Writing a 1 to this register flushes TX FIFO. (This is a self clearing bit.)

TXDMA address offset: 0x01C8

Bit	R/W	Reset	Name	Description
31:0	W	0x0	TXDMA	Transmitter Block DMA Register. This register can be used to cycle repeatedly through the enabled Transmit channels to allow writing of stereo data pairs.

RTXDMA address offset: 0x01CC

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	W	0x0	RTXDMA	Reset Transmitter Block DMA Register. Writing a 1 to this self-clearing register resets the TXDMA register.

8.5.5 I2S_RX Register Map

Offset	Name	Description
0x00000	IER	Onmicro_i2s Enable Register Reset Value: 0x0
0X00004	IRER	I2S Receiver Block Enable Register Reset Value: 0x0
0x0000c	CER	Clock Enable Register Reset Value: 0x0
0x00010	CCR	Clock Configuration Register Rest Value:Configuration dependent
0X00014	RXFFR	Receiver Block FIFO Register Reset Value: 0x0
0X00020	LRBR0	Left Receiver Buffer 0 Reset Value: 0x0
0X00024	RRBR0	Right Receiver Buffer 0

		Reset Value: 0x0
0X00028	RER0	Receiver Enable Register 0 Reset Value: 0x1
0X00030	RCR0	Receiver Configuration Register 0 Reset Value: Configuration Dependent
0X00038	ISR0	Interrupt Status Register 0 Reset Value: 0x10
0X0003C	IMR0	Interrupt Mask Register 0 Reset Value: 0x33
0X00040	ROR0	Receive Overrun Register 0 Reset Value: 0x0
0X00048	RFCR0	Receive FIFO Configuration Register 0 Reset Value:Configuration Dependent
0X00050	RFF0	Receive FIFO Flush 0 Reset Value: 0x0
0X00060	LRBR1	Left Receiver Buffer 1 Reset Value: 0x0
0X00064	RRBR1	Right Receiver Buffer 1 Reset Value: 0x0
0X00068	RER1	Receiver Enable Register 1 Reset Value: 0x1
0X00070	RCR1	Receiver Configuration Register 1 Reset Value: Configuration Dependent
0X00078	ISR1	Interrupt Status Register 1 Reset Value: 0x10
0X0007C	IMR1	Interrupt Mask Register 1 Reset Value: 0x33
0X00080	ROR1	Receive Overrun Register 1 Reset Value: 0x0
0X00088	RFCR1	Receive FIFO Configuration Register 1 Reset Value: Configuration Dependent
0X00090	RFF1	Receive FIFO Flush 1 Reset Value: 0x0
0X000A0	LRBR2	Left Receiver Buffer 2 Reset Value: 0x0
0X000A4	RRBR2	Right Receiver Buffer 2 Reset Value: 0x0
0X000A8	RER2	Receiver Enable Register 2 Reset Value: 0x1
0X000B0	RCR2	Receiver Configuration Register 2 Reset Value: Configuration Dependent
0X000B8	ISR2	Interrupt Status Register 2 Reset Value: 0x10

0X000BC	IMR2	Interrupt Mask Register 2 Reset Value: 0x33
0X000C0	ROR2	Receive Overrun Register 2 Reset Value: 0x0
0X000C8	RFCR2	Receive FIFO Configuration Register 2 Reset Value: Configuration Dependent
0X000D0	RFF2	Receive FIFO Flush 2 Reset Value: 0x0
0X000E0	LRBR3	Left Receiver Buffer 3 Reset Value: 0x0
0X000E4	RRBR3	Right Receiver Buffer 3 Reset Value: 0x0
0X000E8	RER3	Receiver Enable Register 3 Reset Value: 0x1
0X000F0	RCR3	Receiver Configuration Register 3 Reset Value: Configuration Dependent
0X000F8	ISR3	Interrupt Status Register 3 Reset Value: 0x10
0X000FC	IMR3	Interrupt Mask Register 3 Reset Value: 0x33
0X00100	ROR3	Receive Overrun Register 3 Reset Value: 0x0
0X00108	RFCR3	Receive FIFO Configuration Register 3 Reset Value: Configuration Dependent
0X00110	RFF3	Receive FIFO Flush 3 Reset Value: 0x0
0X001C0	RXDMA	Receiver Block DMA Register Reset Value: 0x00
0X001C4	RRXDMA	Reset Receiver Block DMA Register Reset Value: 0x00
0x01F0	I2S_COMP_PARAM_2	Component Parameter 2 Register Reset Value: Reset value depends on user configuration. For more information, refer to
0x01F4	I2S_COMP_PARAM_1	Component Parameter 1 Register Reset Value: Reset value depends on user configuration. For more information, refer to
0x01F8	I2S_COMP_VERSION	Component Version ID Reset Value: See the releases table in the
0x01FC	I2S_COMP_TYPE	Design Ware Component Type Reset Value: 0x445701a0

IER address offset: 0x0000

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	RW	0x0	SINGLE_EN	Single track enable 1: enable 0: disable
3:1	N/A	0x0	N/A	reserved
0	RW	0x0	IEN	I2S global enable, a disable on this bit overrides any other block enables and flushes all FIFOs. 1: enable 0: disable

IRER address offset: 0x0004

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	RXEN	I2S Receiver enable 1: enable receiver 0: disable receiver

RXFFR address offset: 0x0014

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	W	0x0	RXFFR	Receiver FIFO Reset. Writing a 1 to this register flushes all the RX FIFOs (this is a self clearing bit). The Receiver Block must be disabled prior to writing this bit.

LRBR address offset: 0x0020

Bit	R/W	Reset	Name	Description
31:0	R	0x0	LRBR	The left stereo data received serially from the receive channel input is read through this register

RRBR address offset: 0x0024

Bit	R/W	Reset	Name	Description
31:0	R	0x0	RRBR	The right stereo data received serially from the receive channel input is read through this register.

RER address offset: 0x0028

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x1	RXCHEN	Receive channel enable. On enable, the channel begins receiving on the next left stereo cycle 0: Disable 1: Enable

RCR address offset: 0x0030

Bit	R/W	Reset	Name	Description
31:3	N/A	0x0	N/A	reserved
2:0	RW	0x0	WLEN	These bits are used to program the desired data resolution of the receiver and enables the LSB of the incoming left (or right) word to be placed in the LSB of the LRBR (or RRBR) register. 000 = Ignore word length 001 = 12 bit resolution 010 = 16 bit resolution 011 = 20 bit resolution 100 = 24 bit resolution 101 = 32 bit resolution

ISR address offset: 0x0038

Bit	R/W	Reset	Name	Description
31:2	N/A	0x0	N/A	reserved
1	R	0x0	RXFO	Status of Data Overrun interrupt for the RX channel. Incoming data lost due to a full RX FIFO. 0: RX FIFO write valid 1: RX FIFO write overrun
0	R	0x0	RXDA	Status of Receive Data Available interrupt. RX FIFO data available. 1: trigger level reached 0: trigger level not reached

IMR address offset: 0x003C

Bit	R/W	Reset	Name	Description
31:2	N/A	0x0	N/A	reserved
1	RW	0x1	RXFOM	Masks RX FIFO Overrun interrupt. 1: masks interrupt 0: unmasks interrupt

0	RW	0x1	RXDAM	Masks RX FIFO Data Available interrupt. 1: masks interrupt 0: unmasks interrupt

ROR address offset: 0x0040

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	RXCHO	Read this bit to clear the RX FIFO Data Overrun interrupt. 0: RX FIFO write valid 1: RX FIFO write overrun

RFCR address offset: 0x0048

Bit	R/W	Reset	Name	Description
31:4	N/A	0x0	N/A	reserved
3:0	RW	0x0	RXCHDT	These bits program the trigger level in the RX FIFO at which the Received Data Available interrupt is generated.

RFF address offset: 0x0050

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	W	0x0	RXCHFR	Receive Channel FIFO Reset. Writing a 1 to this register flushes an individual RX FIFO. (This is a self clearing bit.)

RXDMA address offset: 0x01C0

Bit	R/W	Reset	Name	Description
31:0	R	0x0	RXDMA	Receiver Block DMA Register. Used to cycle repeatedly through the enabled receive channels, reading stereo data pairs.

RRXDMA address offset: 0x01C4

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	W	0x0	RRXDMA	Reset Receiver Block DMA Register. Writing a 1 to this self-clearing register resets the RXDMA register.

8.6 QDECx

8.6.1 Introduction

The QDEC functional test content: Rotating encoder function.

8.6.2 Main Features

- 32-bit up,down,up/down auto-reload counter.
- 32-bit programmable prescaler.(allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 2^{32}-1).
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- Interrupt/DMA generation on the following events:
  - Update:counter overflow or underflow,counter initialization (by software or internal or external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input

8.6.3 Function Description

The main block of the programmable advanced-control timer is a 32-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software.This is true even when the counter is running.

8.6.4 QDECx Register Map

Offset	Name	Description
0x0000	ENCODER_EN	Encoder enable
0x0004	ENCODER_ARR	Encoder auto reload control
0x0008	ENCODER_CNT	Encoder count
0x000c	ENCODER_MODE	Encoder mode control
0x0010	ENCODER_SAM	Encoder sample control
0x0014	ENCODER_OF	Encoder overflow, underflow

0x0018	ENCODER_PMUX	Encoder pinmux control

ENCODER_EN address offset: 0x0000

Bit	R/W	Reset	Name	Description
31:3	N/A	0x0	N/A	reserved
2	RW	0x0	ien_move	counter move interrupt enable: 1: enable 0: disable
1	RW	0x0	ien	interrupt enable: 1: enable 0: disable
0	RW	0x0	cen	counter enable: 1: enable 0: disable

ENCODER_ARR address offset: 0x0004

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	arr	auto reload register. when counter counts up and meet this value, counter returns to 0; when counter is 0 and is sheduled to count down,arr value is loaded into counter.

ENCODER_CNT address offset: 0x0008

Bit	R/W	Reset	Name	Description
31:0	R	0x0	cnt	internal counter value. read only.

ENCODER_MODE address offset: 0x000C

Bit	R/W	Reset	Name	Description
31:4	N/A	0x0	N/A	reserved
3	RW	0x0	cc2p	ti2 polarity. 0: ti2 not inverted; 1: ti2 inverted.
2	RW	0x0	cc1p	ti1 polarity. 0: ti1 not inverted; 1: ti1 inverted.
1:0	RW	0x0	sms	01: Encoder mode 1 - Counter counts up/down on ti2 edge depending on ti1 level. 10: Encoder mode 2 - Counter counts up/down on ti1 edge depending on ti2 level. 00: Encoder mode 3 - Counter counts

				up/down on both inputs edges depending on the level of the other input.

ENCODER_SAM address offset: 0x0010

Bit	R/W	Reset	Name	Description
31:4	N/A	0x0	N/A	reserved

3:0	RW	0x0	icf	input signal is considered to be valid only if "n_event" consecutive samples are the same.
				icf=0: n_event = 1;
				icf=1: n_event = 2;
				icf=2: n_event = 4;
				icf=3: n_event = 8;
				icf=4: n_event = 48;
				icf=5: n_event = 64;
				icf=6: n_event = 96;
				icf=7: n_event = 128;
				icf=8: n_event = 192;
				icf=9: n_event = 256;
				icf=10: n_event = 320;
				icf=11: n_event = 384;
				icf=12: n_event = 512;
				icf=13: n_event = 640;
				icf=14: n_event = 768;
				icf=15: n_event = 1024;

ENCODER_OF address offset: 0x0014

Bit	R/W	Reset	Name	Description
31:10	N/A	0x0	N/A	reserved
9	R	0x0	cnt_mv_clr	cnt move interrupt clear signal flag
8	R	0x0	cnt_mv_clr_apb	cnt move interrupt clear signal flag
7	R	0x0	cnt_uf_clr	cnt underflow interrupt clear signal flag
6	R	0x0	cnt_uf_clr_apb	cnt underflow interrupt clear signal flag
5	R	0x0	cnt_of_clr	cnt overflow interrupt clear signal flag
4	R	0x0	cnt_of_clr_apb	cnt overflow interrupt clear signal flag
3	RW	0x0	cnt_mv_flag	count move interrupt Wirte 1 to clear, self clear

2	R	0x0	dir	counter direction
				0: up counting;
				1: down counting;
				this bit is read only.

1	RW	0x0	overflow	counter overflow interrupt Wirte 1 to clear, self clear
0	RW	0x0	underflow	counter underflow interrupt

				Wirte 1 to clear, self clear

ENCODER_PMUX address offset: 0x0018

Bit	R/W	Reset	Name	Description
31:14	N/A	0x0	N/A	reserved
13:8	RW	0x0	ti2_sel	config which gpio as ti2 input
7:6	N/A	0x0	N/A	reserved
5:0	RW	0x0	ti1_sel	config which gpio as ti1 input

8.7 AUDIO

8.7.1 Introduction

The HS6621Cx has an audio PGA and ADC inside. The structure is as shown below, the PGA has a single input, which can be switched from GPIO[2,3]. The PGA gain varies from PGA GAIN -3 to +30db(3db step). The ADC gain varies from -6 to 12db (6db step).The PGA cascades ADC，dynamic range is 95dB, SNDR is 88dB, and SNR is 93 dB.

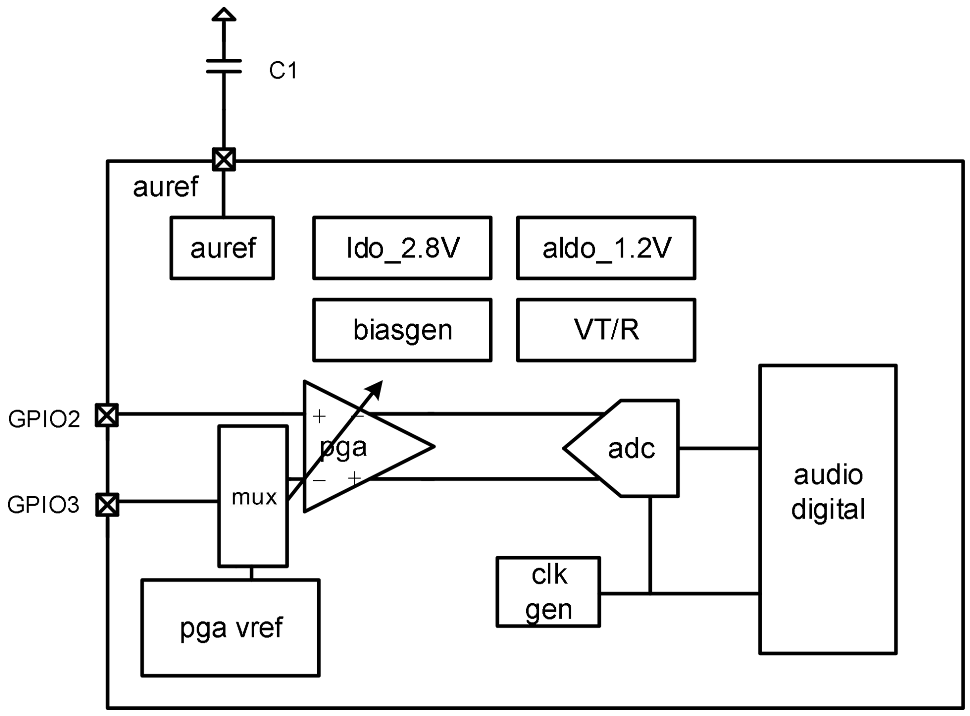


Figure 8.16 AUDIO block diagram

8.7.2 AUDIO Register Map

Offset	Name	Description
0x0000	ADC_CTRL	ADC control
0x0004	ADC_DECI_FILT_11	ADC filter coefficients 1
0x0008	ADC_DECI_FILT_12	ADC filter coefficients 2
0x000c	ADC_DECI_FILT_13	ADC filter coefficients 3
0x0010	ADC_DECI_FILT_21	ADC filter coefficients 4
0x0014	ADC_DECI_FILT_22	ADC filter coefficients 5
0x0018	ADC_DECI_FILT_23	ADC filter coefficients 6
0x001c	ADC_DECI_FILT_31	ADC filter coefficients 7
0x0020	ADC_DECI_FILT_32	ADC filter coefficients 8
0x0024	ADC_DECI_FILT_33	ADC filter coefficients 9

0x0028	ADC_DECI_FILT_41	ADC filter coefficients 10
0x002c	ADC_DECI_FILT_42	ADC filter coefficients 11
0x0030	ADC_DECI_FILT_43	ADC filter coefficients 12
0x0034	ADC_DECI_FILT_51	ADC filter coefficients 13
0x0038	ADC_DECI_FILT_52	ADC filter coefficients 14
0x003c	ADC_DECI_FILT_53	ADC filter coefficients 15
0x0040	ADC_IIR_FILT_11	ADC iir filter coefficients 1
0x0044	ADC_IIR_FILT_12	ADC iir filter coefficients 2
0x0048	ADC_IIR_FILT_13	ADC iir filter coefficients 3
0x004c	ADC_IIR_FILT_21	ADC iir filter coefficients 4
0x0050	ADC_IIR_FILT_22	ADC iir filter coefficients 5
0x0054	ADC_IIR_FILT_23	ADC iir filter coefficients 6
0x0058	ADC_IIR_FILT_31	ADC iir filter coefficients 7
0x005c	ADC_IIR_FILT_32	ADC iir filter coefficients 8
0x0060	ADC_IIR_FILT_33	ADC iir filter coefficients 9
0x0064	ADC_DRC_CTRL_1	ADC drc control register 1
0x0068	ADC_DRC_CTRL_2	ADC drc control register 2
0x006c	ADC_DRC_CTRL_3	ADC drc control register 3
0x0070	ADC_DRC_CTRL_4	ADC drc control register 4
0x0074	ADC_DRC_CTRL_5	ADC drc control register 5
0x0078	ADC_VOL_CTRL	ADC volume control
0x00c0	ADC_GAIN_READ	ADC internal gain read
0x00c4	ADC_CLK_CTRL_1	ADC clock control register 1
0x00c8	ADC_CLK_CTRL_2	ADC clock control register 2
0x00cc	ADC_CLK_CTRL_3	ADC clock control register 3
0x00d0	ADC_DITHER_CTRL_1	ADC dither control 1
0x00d4	ADC_DITHER_CTRL_2	ADC dither control 2
0x00d8	CIC_ONLY_CTRL	ADC cic filter control
0x00dc	ADC_INT_STATUS	ADC interrupt status
0x00e0	ADC_INT_CTRL	ADC interrupt control
0x00e4	CIC_ONLY_RDATA	ADC cic filter output
0x00e8	CIC_ONLY_RDATA_1	ADC cic filter output
0x00f0	ADC_PEAK_READ	ADC output peak read
0x00f8	ADC_AMP_CTRL	ADC amplitude control
0x00fc	IF_CTRL	Interface control
0x0100	ADC_ANA_CTRL_1	ADC analog control register 1
0x0104	ADC_ANA_CTRL_2	ADC analog control register 2
0x010c	ADC_ANA_PWR_1	ADC power control register 1

ADC_CTRL address offset: 0x0000

Bit	R/W	Reset	Name	Description
31:24	N/A	0x0	N/A	reserved

23	RW	0x0	cic_only	1: the cic filter output is directly as adc output
22	RW	0x0	adc_cic_3ord	1: 3-order cic filter used
21	RW	0x0	adc_cic_2ord	1: 2-order cic filter used
20	RW	0x0	adc_cic_1ord	1: 1-order cic filter used
19	RW	0x0	adc_out_sel	High sample rate adc output select 1: 4x or 5x sample rate output
18	RW	0x0	adc_cic_4ord	adc cic filter order config 1: use 4-order cic 0: use 3-order cic
17	RW	0x0	adc_anti_clip	1: when output is clipping, the gain decreased automatically
16	RW	0x0	adc_input_inv	1: adc input is inverted
15	N/A	0x0	N/A	reserved
14	RW	0x1	adc_sr_32k	1: sample rate base = 32k
13	RW	0x0	adc_sr_44k	1: sample rate base = 44k
12	RW	0x0	adc_sr_48k	1: sample rate base = 48k
11	N/A	0x0	N/A	reserved
10	RW	0x1	adc_sr_4x	1: sample rate = sample rate base/4 (12k, 11k or 8k)
9	RW	0x0	adc_sr_2x	1: sample rate = sample rate base/2 (24k, 22k or 16k)
8	RW	0x0	adc_sr_1x	1: sample rate = sample rate base (48k, 44k or 32k)
7	RW	0x0	adc_24b_en	adc output 24bit enable 1: enable 0: disable
6:4	RW	0x0	adc_cic_gain	cic filter gain 0: 0dB 1: 2dB 2: 3.5dB 3,4: 6dB 5: 8dB 6: 9.5dB 7: 12dB
3	RW	0x1	adc_dc_en	dc offset enable 1: dc offset is enable 0: dc offset is disable
2	RW	0x0	dmic_en	DMIC enable 1: digital mic as input 0: analog mic as input
1	RW	0x0	adc_sw_reset_x	ADC soft reset 0: reset

0	RW	0x0	adc_en	ADC enable 1: enable 0: disable

ADC_DECI_FILT_11 address offset: 0x0004

Bit	R/W	Reset	Name	Description
31:0	RW	0xcbed6256	adc_deci_filt_11	adc filter coeffieicnts 1

ADC_DECI_FILT_12 address offset: 0x0008

Bit	R/W	Reset	Name	Description
31:0	RW	0xe7b7117f	adc_deci_filt_12	adc filter coeffieicnts 2

ADC_DECI_FILT_13 address offset: 0x000c

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x117f	adc_deci_filt_13	adc filter coeffieicnts 3

ADC_DECI_FILT_21 address offset: 0x0010

Bit	R/W	Reset	Name	Description
31:0	RW	0xd59c5ea3	adc_deci_filt_21	adc filter coeffieicnts 4

ADC_DECI_FILT_22 address offset: 0x0014

Bit	R/W	Reset	Name	Description
31:0	RW	0xcbe12186	adc_deci_filt_22	adc filter coeffieicnts 5

ADC_DECI_FILT_23 address offset: 0x0018

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x2186	adc_deci_filt_23	adc filter coeffieicnts 6

ADC_DECI_FILT_31 address offset: 0x001c

Bit	R/W	Reset	Name	Description
31:0	RW	0xc56b6500	adc_deci_filt_31	adc filter coeffieicnts 7

ADC_DECI_FILT_32 address offset: 0x0020

Bit	R/W	Reset	Name	Description
31:0	RW	0xf3d60cb5	adc_deci_filt_32	adc filter coeffieicnts 8

ADC_DECI_FILT_33 address offset: 0x0024

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x0cb5	adc_deci_filt_33	adc filter coeffieicnts 9

ADC_DECI_FILT_41 address offset: 0x0028

Bit	R/W	Reset	Name	Description
31:0	RW	0xde5f5b59	adc_deci_filt_41	adc filter coeffieicnts 10

ADC_DECI_FILT_42 address offset: 0x002c

Bit	R/W	Reset	Name	Description
31:0	RW	0xe567118f	adc_deci_filt_42	adc filter coeffieicnts 11

ADC_DECI_FILT_43 address offset: 0x0030

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x118f	adc_deci_filt_43	adc filter coeffieicnts 12

ADC_DECI_FILT_51 address offset: 0x0034

Bit	R/W	Reset	Name	Description
31:0	RW	0xc1866709	adc_deci_filt_51	adc filter coeffieicnts 13

ADC_DECI_FILT_52 address offset: 0x0038

Bit	R/W	Reset	Name	Description
31:0	RW	0x0b1e0cb5	adc_deci_filt_52	adc filter coeffieicnts 14

ADC_DECI_FILT_53 address offset: 0x003c

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x0cb5	adc_deci_filt_53	adc filter coeffieicnts 15

ADC_IIR_FILT_11 address offset: 0x0040

Bit	R/W	Reset	Name	Description
31:0	RW	0xc5717a51	adc_iir_filt_11	adc iir filter coeffieicnts 1

ADC_IIR_FILT_12 address offset: 0x0044

Bit	R/W	Reset	Name	Description
31:0	RW	0x85903d38	adc_iir_filt_12	adc iir filter coeffieicnts 2

ADC_IIR_FILT_13 address offset: 0x0048

Bit	R/W	Reset	Name	Description
31:17	N/A	0x0	N/A	reserved
16:0	RW	0x13d38	adc_iir_filt_13	adc iir filter coeffieicnts 3

ADC_IIR_FILT_21 address offset: 0x004c

Bit	R/W	Reset	Name	Description
31:0	RW	0xd6f463e8	adc_iir_filt_21	adc iir filter coeffieicnts 4

ADC_IIR_FILT_22 address offset: 0x0050

Bit	R/W	Reset	Name	Description
31:0	RW	0x9986333d	adc_iir_filt_22	adc iir filter coeffieicnts 5

ADC_IIR_FILT_23 address offset: 0x0054

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x333d	adc_iir_filt_23	adc iir filter coeffieicnts 6

ADC_IIR_FILT_31 address offset: 0x0058

Bit	R/W	Reset	Name	Description
31:0	RW	0x08c309d7	adc_iir_filt_31	adc iir filter coeffieicnts 7

ADC_IIR_FILT_32 address offset: 0x005c

Bit	R/W	Reset	Name	Description
31:0	RW	0xf4c94128	adc_iir_filt_32	adc iir filter coeffieicnts 8

ADC_IIR_FILT_33 address offset: 0x0060

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0xf78c	adc_iir_filt_33	adc iir filter coeffieicnts 9

ADC_DRC_CTRL_1 address offset: 0x0064

Bit	R/W	Reset	Name	Description
31:16	N/A	0xe3a2	adc_drc_at1	adc drc attack time 1
15:0	RW	0xe3a2	adc_drc_at0	adc drc attack time 0

ADC_DRC_CTRL_2 address offset: 0x0068

Bit	R/W	Reset	Name	Description
31:16	N/A	0x2401	adc_drc_rt1	adc drc release time 1
15:0	RW	0xdbff	adc_drc_rt0	adc drc release time 0

ADC_DRC_CTRL_3 address offset: 0x006c

Bit	R/W	Reset	Name	Description
31:28	N/A	0x1	adc_drc_gain_jitter	adc drc gain hysteresis setting
27	N/A	0x0	N/A	reserved
26:24	RW	0x4	adc_drc_nt	adc drc noise threshold
23	N/A	0x0	N/A	reserved
22:16	RW	0x0	adc_drc_et	adc drc expand threshold
15	N/A	0x0	N/A	reserved
14:8	RW	0x0	adc_drc_ct	adc drc compress threshold
7	N/A	0x0	N/A	reserved
6:2	RW	0x0	adc_drc_lt	adc drc limit threshold

1:0	RW	0x0	adc_drc_mode_sel	0: disable 1: enable 2,3: reserved

ADC_DRC_CTRL_4 address offset: 0x0070

Bit	R/W	Reset	Name	Description
31:24	N/A	0x0	adc_drc_gain_neg_limit	drc min gain
23:16	RW	0xff	adc_drc_gain_pos_limit	drc max gain
15:12	N/A	0x0	N/A	reserved
11:10	RW	0x0	adc_drc_gain_shift	0: no gain shift 1: add gain shift
9	RW	0x0	adc_drc_es_inv	0: es no invert 1: es is inverted
8	RW	0x0	adc_drc_ns_mode	0: output is mute in noise gate 1: output is attenuated in noise gate
7:6	RW	0x2	adc_drc_delay	adc drc input delay 0: no delay 1: 4 samples delay 2: 8 samples delay 3: 12 samples delay
5:3	RW	0x0	adc_drc_es	adc drc expand slop
2:0	RW	0x0	adc_drc_cs	adc drc compress slop

ADC_DRC_CTRL_5 address offset: 0x0074

Bit	R/W	Reset	Name	Description
31:17	N/A	0x0	N/A	reserved
16	RW	0x1	adc_drc_rms_en	drc rms computation enable
15:0	RW	0x028f	adc_drc_tav	drc rms computation coefficients

ADC_VOL_CTRL address offset: 0x0078

Bit	R/W	Reset	Name	Description
31:28	N/A	0x1	adc_unmute_rate	the volume adjust rate in unmute process
27:25	N/A	0x0	N/A	reserved
24	RW	0x1	adc_vol_update	ladcvol take effect only update while adc_vol_update=1
23:16	N/A	0x0	N/A	reserved
15:8	RW	0x64	ladcvol	the volume control, 0.5dB/step
7:4	RW	0x1	adc_mute_rate	the volume adjust rate in mute process
3	N/A	0x0	N/A	reserved
2	RW	0x0	adc_mute_bypass	1: mute and unmute mechanism is bypass, volume is only controlled by

				ladcvol
1	RW	0x1	adcunmu	1: gain gradually increased from 0 to ladcvol
0	RW	0x0	adcmu	1: gain gradually decreased to 0

ADC_GAIN_READ address offset: 0x00c0

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	R	0x64	adc_gain_read	adc gain control read out (read only)

ADC_CLK_CTRL_1 address offset: 0x00c4

Bit	R/W	Reset	Name	Description
31:7	N/A	0x0	N/A	reserved
6	RW	0x0	dmic_clk_ctrl	0: output dmic_clk, 1: output dmic_clk_n
5	RW	0x0	dmic_clk_sel	0: dmic_clk=3MHz, 1: dmic_clk=2.4MHz
4	RW	0x0	clk_32k_en	0: clk_32k is gating
3	RW	0x0	i2s_rxclk_rstn_reg	i2s_rx clock reset, low active
2	RW	0x0	i2s_txclk_rstn_reg	i2s_tx clock reset, low active
1	RW	0x0	adc_clk_en_reg	adc clock enable, active high
0	RW	0x0	adc_rstn_reg	adc reset, active low

ADC_CLK_CTRL_2 address offset: 0x00c8

Bit	R/W	Reset	Name	Description
31:25	N/A	0x0	N/A	reserved
24	R	0x0	i2s_tx_sclk_done	i2s_tx_sclk ready flag
23:21	N/A	0x0	N/A	reserved
20:12	RW	0x19	i2s_tx_high_num	the high pulse of i2s_tx_ws
11:9	N/A	0x0	N/A	reserved
8	RW	0x0	i2s_tx_odd	i2s_tx_ws cofiguration: 0: high pulse=low pulse, 1: low pulse=high pulse-1
7	RW	0x1	i2s_tx_div_en	1: i2s_tx_sclk divider is enable
6	RW	0x1	i2s_tx_div_sel	0: i2s_tx_sclk = 16MHz, 1: i2s_tx_sclk=16M/i2s_tx_div_coeff
5:0	RW	0x14	i2s_tx_div_coeff	i2s_tx_sclk divider, greater than 2

ADC_CLK_CTRL_3 address offset: 0x00cc

Bit	R/W	Reset	Name	Description
31:25	N/A	0x0	N/A	reserved
24	R	0x0	i2s_rx_sclk_done	i2s_rx_sclk ready flag
23:21	N/A	0x0	N/A	reserved

20:12	RW	0x19	i2s_rx_high_num	the high pulse of i2s_rx_ws
11:9	N/A	0x0	N/A	reserved
8	RW	0x0	i2s_rx_odd	i2s_rx_ws cofiguration: 0: high pulse=low pulse, 1: low pulse=high pulse-1
7	RW	0x1	i2s_rx_div_en	1: i2s_rx_sclk divider is enable
6	RW	0x1	i2s_rx_div_sel	0: i2s_rx_sclk = 16MHz, 1: i2s_rx_sclk=16M/i2s_rx_div_coeff
5:0	RW	0x14	i2s_rx_div_coeff	i2s_rx_sclk divider, greater than 2

ADC_DITHER_CTRL_1 address offset: 0x00d0

Bit	R/W	Reset	Name	Description
31:17	N/A	0x0	N/A	reserved
16	RW	0x0	adc_dither_out_en	1: adc dither output is enable, 0: adc dither output is 0
15	RW	0x0	pnoise_en	1: pnoise generator is enable for adc dither
14	RW	0x0	adc_peak_mode	1: drc peak output 0: adc peak output
13	RW	0x0	adc_dither_en	1: adc_dither is enable
12	RW	0x1	adc_dither_pd	1: adc_dither_clk is power down
11:0	N/A	0x0	N/A	reserved

ADC_DITHER_CTRL_2 address offset: 0x00d4

Bit	R/W	Reset	Name	Description
31:16	RW	0xfdbf	adc_dither_rt	adc dither release time
15:0	RW	0xaac9	adc_dither_at	adc dither attack time

CIC_ONLY_CTRL address offset: 0x00d8

Bit	R/W	Reset	Name	Description
31:20	N/A	0x0	N/A	reserved
19:0	RW	0x3e7	cic_only_timer	the counter is to control cic filter sample rate

ADC_INT_STATUS address offset: 0x00dc

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7	RW	0x0	cic_only_int_status	in cic only mode, generate interrupt per sample, write 1 to clear the interrupt
6	RW	0x0	audio_off_int	audio power off complete interrupt, write 1 to clear the interrupt
5	RW	0x0	audio_on_int	audio power on complete interrupt, write 1 to clear the interrupt
4	R	0x0	adc_intt	all interrupts are OR

3	R	0x0	adc_unmute_int	adc unmute interrupt
2	R	0x0	adc_mute_int	adc mute interrupt
1	R	0x0	adc_outl_clip_int	adc output clipping interrupt status
0	RW	0x0	adc_signal_large_int	adc large signal interrupt status, write 1 to clear the interrupt

ADC_INT_CTRL address offset: 0x00e0

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7	RW	0x1	cic_only_int_mask	1: cic only int is mask
6	RW	0x1	audio_off_int_mask	1: audio off int is mask
5	RW	0x1	audio_on_int_mask	1: audio on int is mask
4	N/A	0x0	N/A	reserved
3	RW	0x1	adc_unmute_int_mask	1: adc unmute interrupt mask
2	RW	0x1	adc_mute_int_mask	1: adc mute interrupt mask
1	RW	0x1	adc_clip_int_mask	1: adc clip int is mask and clear
0	RW	0x1	adc_signal_large_int_im	1: interrupt is mask (the interrupt indicates adc signal is large)

CIC_ONLY_RDATA address offset: 0x00e4

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	cic_only_out[31:0]	cic filter output

CIC_ONLY_RDATA_1 address offset: 0x00e8

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4:0	RW	0x0	cic_only_out[36:32]	cic filter output

RSVD_1 address offset: 0x00ec

Bit	R/W	Reset	Name	Description
31:0	RW	0x0	rsvd_reg_1	reserved registers

ADC_PEAK_READ address offset: 0x00f0

Bit	R/W	Reset	Name	Description
31:16	R	0x0	adc_peak_2	drc rms output
15:0	R	0x0	adc_peak_1	peak output (or drc peak output), controlled by ADC_DITHER_CTRL_1[14]

ADC_AMP_CTRL address offset: 0x00f8

Bit	R/W	Reset	Name	Description
31:16	R	0x0	amp_delay_cnt	The delay time at which adc out exceeds signal_amp_thd
15:0	RW	0x1000	signal_amp_thd	the threshold of signal

IF_CTRL address offset: 0x00fc

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7	RW	0x0	i2s_txclk_mst_inv	1: pinmux output i2s_txclk_n 0: pinmux output i2s_txclk
6	RW	0x0	i2s_rxclk_mst_inv	1: pinmux output i2s_rxclk_n 0: pinmux output i2s_rxclk
5	RW	0x1	i2s_tx_ms_sel	1: i2s tx clock and ws is generated by adc 0: i2s tx clock and ws is input from gpio
4	RW	0x1	i2s_rx_ms_sel	1: i2s rx clock and ws is generated by adc 0: i2s rx clock and ws is input from gpio
3	RW	0x0	i2s_tx_cnt_en	1: i2s_tx counter is enable (counter value is read in ADC_AMP_CTRL)
2	RW	0x0	i2s_con_ctrl	0: adc connected to i2s; 1: adc and i2s connected to gpio
1	RW	0x0	transmit_en	enable data transmitor to i2s interface
0	RW	0x0	receive_en	enable data receiver from i2s interface

ADC_ANA_CTRL_1 address offset: 0x0100

Bit	R/W	Reset	Name	Description
31	RW	0x1	pd_audio_iref_vbat	
30	RW	0x1	sel_inneriref_vbat	
29	RW	0x1	pd_au_bias_gen	Power down audio bias gen,1=power down
28	RW	0x0	au_en_ref_byp	bypass reference's big-R for fast settling
27	RW	0x1	pd_auref	Power down audio reference gen, 1=power down

26	RW	0x0	au_sel_adc_input	select ADC input;
				0=normal input;
				1=test input

25	RW	0x0	au_en_tst_auadc_bias	enable ADC bias current test;
				0=no test;
				1=enable

24	RW	0x0	au_shrt_adc	short ADC input for testing
23	RW	0x0	au_rst_adc	reset auido ADC; 1=reset
22:20	RW	0x0	auadc_tstsel	ADC test selection; 000=no test
19:16	RW	0x8	au_rctune	tuning RC constant,
15:14	RW	0x1	auadc_gain	adc gain control
13	RW	0x1	au_en_adc_dem	Enable ADC DEM function; 1=enable, 0=disable
12	RW	0x1	pd_auadc_core	Power down ADC core,1=power down

11	RW	0x1	pd_auadc_bias	ADC bias ,1=power down
10:8	RW	0x0	au_sel_pgain	pga input select
7:4	RW	0x1	au_pga_gain	pga gain control
3	RW	0x1	au_en_pga_singlein	pga input config: 1=single end 0=differential
2	RW	0x1	pd_au_pgavref	Power down PGA VREF,1=power down
1	RW	0x1	pd_au_pga	Power down PGA,1=power down
0	RW	0x1	pd_au_clk	Power down audio clock,1=power down

ADC_ANA_CTRL_2 address offset: 0x0104

Bit	R/W	Reset	Name	Description
31:20	N/A	0x0	N/A	reserved
19:16	RW	0x9	sel_bias_vbat	
15	RW	0x1	en_vreg_igen_vbat	
14	RW	0x1	auldo28mod	2.8V LDO output voltage 0=1.6v, 1=2.8v
13:12	RW	0x1	ctrl_au_auldo28	2.8V DRV LDO output adjust;
11	RW	0x1	en_vreg_2p8ldo_vbat	
10	RW	0x1	en_vreg_1p2ldo_vbat	
9:8	RW	0x1	ctrl_au_auldo12	1.2V LDO output adjust; default is 01
7	RW	0x1	sel_inneriref_auvref_vbat	
6	RW	0x1	pa_au_auldo12	Power down 1.2V AVDD12 LDO, 1=power down
5	RW	0x1	pd_au_auldo28	Power down 2.8V DRV LDO,1=power down
4	RW	0x0	en_au_tst	enable audio test function
3	RW	0x0	en_audio_bias_tst	enable AUDIO bias current test; 0=no test; 1=enable
2:0	RW	0x0	audio_tstsel	audio top test selection; 000=no test

ADC_ANA_PWR_1 address offset: 0x010c

Bit	R/W	Reset	Name	Description
31:11	N/A	0x0	N/A	reserved
10	RW	0x0	audio_adc_ctrl	pga and adc control, 1: control by reg, 0: control by fsm
9	RW	0x0	audio_ldo_ctrl	ldo and bias control, 1: control by reg, 0: control by fsm
8	RW	0x0	audio_on_en	audio power sequence ctrl, 1: power on, 0: power off
7:0	RW	0x34	pd_auref_au_time	pd_auref_au slot time

8.8 UARTx

8.8.1 Introduction

The Onmicro_uart is a programmable Universal Asynchronous Receiver/Transmitter (UART). This component is an AMBA 2.0-compliant Advanced Peripheral Bus (APB) slave device and is part of the family of DesignWare Synthesizable Components .

The Onmicro_uart has been modeled after the industry-standard 16550. However, the register address space has been relocated to 32-bit data boundaries for APB bus implementation. It can be configured, synthesized, and verified using the Onmicro core Consultant GUI to produce RTL.The Onmicro_uart is used for serial communication with a peripheral, modem (data carrier equipment,DCE) or data set. Data is written from a master (CPU) over the APB bus to the UART and it is converted to serial form and transmitted to the destination device. Serial data is also received by the UART and stored for the master (CPU) to read back.The Onmicro_uart contains registers to control the character length, baud rate, parity generation/checking, and interrupt generation.Although there is only one interrupt output signal (intr) from the OM_uart, there are several prioritized interrupt types that can be responsible for its assertion. Each of the interrupt types can be separately enabled/disabled with the control registers .

8.8.2 Main Features

- AMBA APB interface allows easy integration into AMBA SoC implementations
- Configurable parameters for the following:
  - APB data bus widths of 8, 16 and 32
  - Additional DMA interface signals for compatibility with DesignWare DMA interface
  - DMA interface signal polarity
  - Transmit and receive FIFO depths of none, 16, 32, 64,…,2048
  - Use of two clocks (pclk and sclk) instead of one (pclk)
  - IrDA 1.0 SIR mode support with up to 115.2 Kbaud data rate and a pulse duration (width) as follows: width = 3/16 × bit period as specified in the IrDA physical layer specification
  - IrDA 1.0 SIR low-power reception capabilities
  - Baud clock reference output signal
  - Clock gate enable output(s) used to indicate that the TX and RX pipeline is clear (no data) and no activity has occurred for more than one character time, so clocks may be gated
  - FIFO access mode (for FIFO testing) so that the receive FIFO can be written by the master and the transmit FIFO can be read by the master
  - Additional FIFO status registers

  - Shadow registers to reduce software overhead and also include a software programmable reset
  - Auto Flow Control mode as specified in the 16750 standard
  - Loopback mode that enables greater testing of Modem Control and Auto Flow Control features (Loopback support in IrDA SIR mode is available)
  - Transmitter Holding Register Empty (THRE) interrupt mode
  - Busy functionality
- Ability to set some configuration parameters in instantiation
- Configuration identification registers present
- Functionality based on the 16550 industry standard, as follows:
  - Programmable character properties, such as number of data bits per character (5-8), optional parity bit (with odd or even select) and number of stop bits (1, 1.5 or 2)
  - Line break generation and detection
  - DMA signaling with two programmable modes
  - Prioritized interrupt identification
- Programmable FIFO enable/disable
- Programmable serial data baud rate as calculated by the following:
  - baud rate = (serial clock frequency)/(16× divisor)
- External read enable signal for RAM wake-up when using external RAMs
- Modem and status lines are independently controlled
- Complete RTL version
- Separate system resets for each clock domain to prevent metastability

8.8.3 Function Description

8.8.3.1.UART(RS232) Serial Protocol

Because the serial communication between the UART and a selected device is asynchronous, additional bits (start and stop) are added to the serial data to indicate the beginning and end.Utilizing these bits allows two devices to be synchronized.This structure of serial data—accompanied by start and stop bits—is referred to as a character, as shown in the following figure.

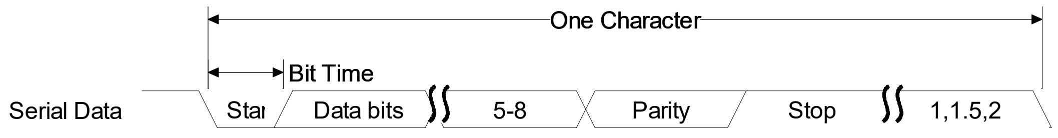


Figure 8.17 Serial Data Format

An additional parity bit can be added to the serial character. This bit appears after the last data bit and before the stop bit(s) in the character structure in order to provide the UART with the ability to perform simple error checking on the received data.

The UART Line Control Register (“LINE_CTRL” register) is used to control the serial character characteristics. The individual bits of the data word are sent after the start bit, starting with the least significant bit (LSB). These are followed by the optional parity bit, followed by the stop bit(s), which can be 1, 1.5, or 2.

All the bits in the transmission (with exception to the half stop bit when 1.5 stop bits are used) are transmitted for exactly the same time duration. This is referred to as a Bit Period or Bit Time. One Bit Time equals 16 baud clocks. To ensure stability on the line the receiver samples the serial input data at approximately the mid point of the Bit Time once the start bit has been detected. As the exact number of baud clocks that each bit was transmitted for is known, calculating the mid point for sampling is not difficult, that is every 16 baud clocks after the mid point sample of the start bit.The following figure shows the sampling points of the first couple of bits in a serial character .

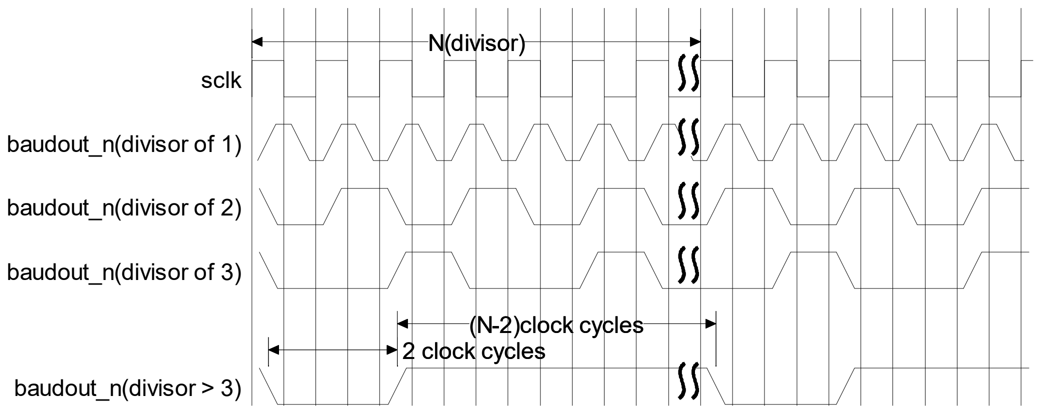


Figure 8.18 Receiver Serial Data Sample Points

As part of the 16550 standard an optional baud clock reference output signal (baudout_n) is supplied to provide timing information to receiving devices that require it. The baud rate of the Onmicro_uart is controlled by the serial clock (sclk or pclk in a single clock implementation) and the Divisor Latch Register (DLH and DLL).The following figure shows the timing diagram for the baudout_n output for different divisor values.

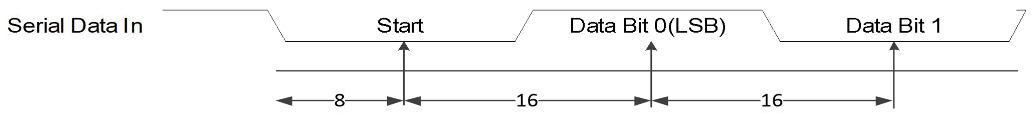


Figure 8.19 Baud Clock Reference Timing Diagram

8.8.3.2.IrDA 1.0 SIR Protocol

The Infrared Data Association (IrDA) 1.0 Serial Infrared (SIR) mode supports bi-directional data communications with remote devices using infrared radiation as the transmission medium. IrDA 1.0 SIR mode specifies a maximum baud rate of 115.2 Kbaud.

8.8.3.3.Attention

Information provided on IrDA SIR mode in this section assumes that the reader is fully familiar with the IrDa Serial Infrared Physical Layer Specifications.

The data format is similar to the standard serial (sout and sin) data format. Each data character is sent serially, beginning with a start bit, followed by 8 data bits, and ending with at least one stop bit. Thus, the number of data bits that can be sent is fixed. No parity information can be supplied and only one stop bit is used while in this mode.Trying to adjust the number of data bits sent or enable parity with the Line Control Register (LCR) has no effect. When the Onmicro_uart is configured to support IrDA 1.0 SIR it can be enabled with Mode Control Register (MCR) bit 6. When the Onmicro_uart is not configured to support IrDA SIR mode,none of the logic is implemented and the mode cannot be activated, reducing total gate counts. When SIR mode is enabled and active, serial data is transmitted and received on the sir_out_n and sir_in ports, respectively.

Transmitting a single infrared pulse signals a logic zero, while a logic one is represented by not sending a pulse. The width of each pulse is 3/16ths of a normal serial bit time. Thus, each new character begins with an infrared pulse for the start bit. However, received data is inverted from transmitted data due to the infrared pulses energizing the photo transistor base of the IrDA receiver, pulling its output low. This inverted transistor output is then fed to the Onmicro_uart sir_in port, which then has correct UART polarity.The following figure shows the timing diagram for the IrDA SIR data format in comparison to the standard serial format.

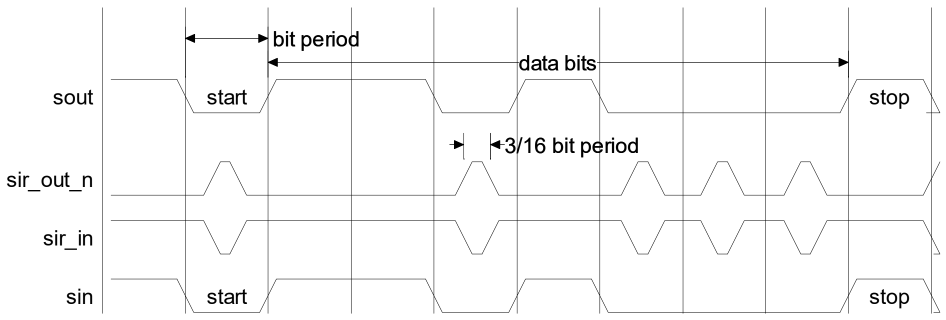


Figure 8.20 IrDA SIR Data Format

As detailed in the IrDA 1.0 SIR, the Onmicro_uart can be configured to support a low-power reception mode. When the Onmicro_uart is configured in this mode, the reception of SIR pulses of 1.41 microseconds (minimum pulse duration) is possible, as well as nominal 3/16 of a normal serial bit time. Using this low-power reception mode requires programming the Low Power Divisor Latch (LPDLL/LPDLH) registers. It should be noted that for all sclk frequencies greater than or equal to 7.37MHz (and obey the requirements of the Low Power Divisor Latch registers), pulses of 1.41uS are detectable. However there are several values of sclk that do not allow the detection of such a narrow pulse and these are as follows:

SCLK	Low Power Divisor Latch Register Value	Min Pulse Width for Detection*
1.84MHz	1	3.77uS
3.69MHz	2	2.086uS
5.53MHz	3	1.584uS
* 10% has been added to the internal pulse width signal to cushion the effect of pulse reduction due to the synchronization and data integrity logic so that a pulse slightly narrower than these may be detectable.		

Table 8.5 Detection of SCLK values for narrow pulses is not allowed

When IrDA SIR mode is enabled, the Onmicro_uart operation is similar to when the mode is disabled,with one exception; data transfers can only occur in half-duplex fashion when IrDA SIR mode is enabled. This is because the IrDA SIR physical layer specifies a minimum of 10ms delay between transmission and reception. This 10ms delay must be generated by software.

8.8.3.4.FIFO Support

The Onmicro_uart can be configured to implement FIFOs to buffer transmit and receive data. If FIFO support is not selected, then no FIFOs are implemented and only a single receive data byte and transmit data byte can be stored at a time in the RBR and THR. This implies a 16450-compatible mode of operation. In this mode most of the enhanced features are unavailable.

In FIFO mode, the FIFOs can be selected to be either external customer-supplied FIFO RAMs or internal DesignWare D-flip-flop based RAMs (Onmicro_ram_r_w_s_dff). If the configured FIFO depth is greater than 256, the FIFO memory selection is restricted to be external. In addition, selection of internal memory restricts the Memory Read Port Type to D-flip-flop based, Synchronous read port RAMs.

When external RAM support is chosen, either synchronous or asynchronous RAMs can be used.Asynchronous RAM provides read data during the clock cycle that has the memory address and read signals active, for sampling on the next rising clock edge. Synchronous single stage RAM registers the data at the current address out and is not available until the next clock cycle (second rising clock edge).The following figure shows the timing diagram for both asynchronous and synchronous RAMs.

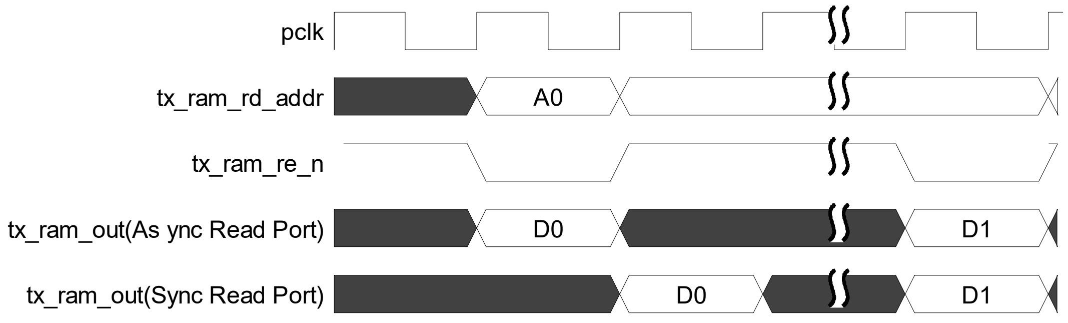


Figure 8.21 Timing for RAM Reads

Note: This timing diagram illustrated in Figure 8.21 assumes the RAM used has a chip select port that is tied to an active value, therefore, the chip is always enabled. This is why the second synchronous read data appears at the same cycle as the asynchronous read data. That is, the address for the second read has been sampled along with the chip select on an earlier edge. Once the output enable (tx_ram_re_n) asserts the data, value on the register output is seen on that same cycle.

Similarly, you can use synchronous RAM for writes, which registers the data at the current address out.The following figure shows the timing diagram for RAM writes.

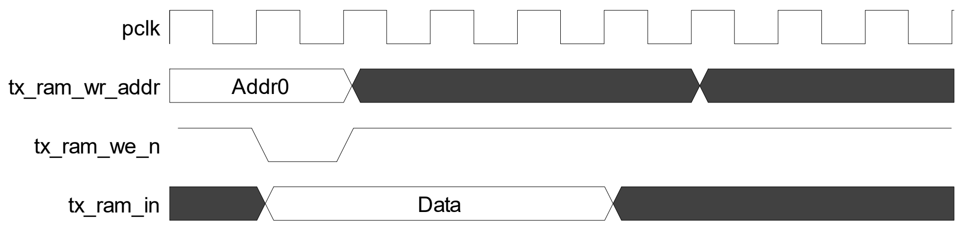


Figure 8.22 Timing for RAM Writes

When FIFO support is selected, an optional programmable FIFO Access mode is available for test purposes, which allows the receive FIFO to be written by the master and the transmit FIFO to be read by the master. When FIFO Access mode is not selected, none of the corresponding logic is implemented and the mode cannot be enabled, reducing overall gate counts. When FIFO Access mode has been selected it can be enabled with the FIFO Access Register (FAR[0]). Once enabled, the control portions of the transmit and receive FIFOs are reset and the FIFOs are treated as empty.

Data can be written to the transmit FIFO as normal; however no serial transmission occurs in this mode (normal operation halted) and hence no data leave the FIFO. The data that has been written to the transmit FIFO can be read back with the Transmit FIFO Read (TFR) register, which when read gives the current data at the top of the transmit FIFO.

Similarly, data can be read from the receive FIFO as normal. Since the normal operation of the Onmicro_uart is halted in this mode, data must be written to the receive FIFO so it may be read back. Data is written to the receive FIFO with the Receive FIFO Write (RFW) register. The upper two bits of the 10 bit register are used to write framing error and parity error detection information to the receive FIFO. Setting bit RFW[9] to indicate a framing error and RFW[8] to indicate a parity error. Although these bit can not be read back via the Receive Buffer Register they can be checked by reading the Line Status Register and checking the corresponding bits when the data in question is at the top of the receive FIFO .

8.8.3.5.Clock Support

The Onmicro_uart can be configured to have either one system clock (pclk) or two system clocks (pclk and sclk). Having the second asynchronous serial clock (sclk) implemented accommodates accurate serial baud rate settings, as well as APB bus interface requirements. When using a single system clock, the system clock settings available for accurate baud rates are greatly restricted.

When a two clock design is chosen, a synchronization module is implemented for synchronization of all control and data across the two system clock boundaries. The RTL diagram for the data synchronization module is shown in Figure 8.23.The data synchronization module can have pending data capability. The timing diagram shown in Figure 8.24 shows this process.

The arrival of new source domain data is indicated by the assertion of start. Since data is now available for synchronization the process is started and busy status is set. If start is asserted while busy and pending data capability has been selected, the new data is stored. When no longer busy the synchronization process starts on the stored pending data. Otherwise the busy status is removed when the current data has been synchronized to the destination domain and the process continues. If only one clock is implemented, all synchronization logic is absent and signals are simply passed through this module.

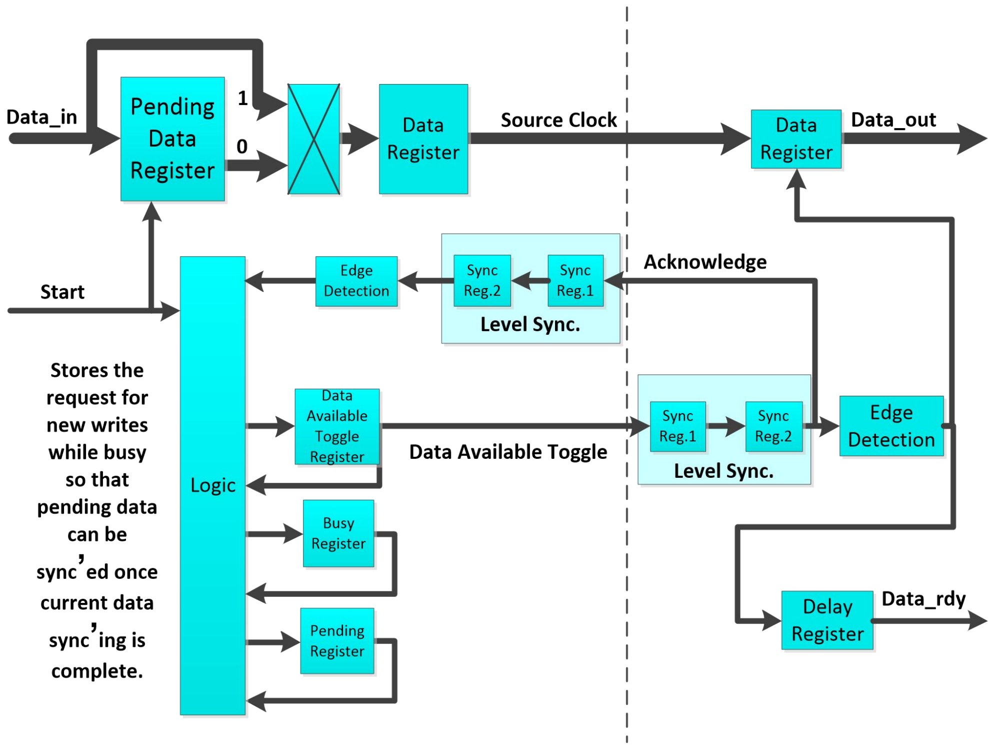


Figure 8.23 RTL Diagram of Data Synchronization Module

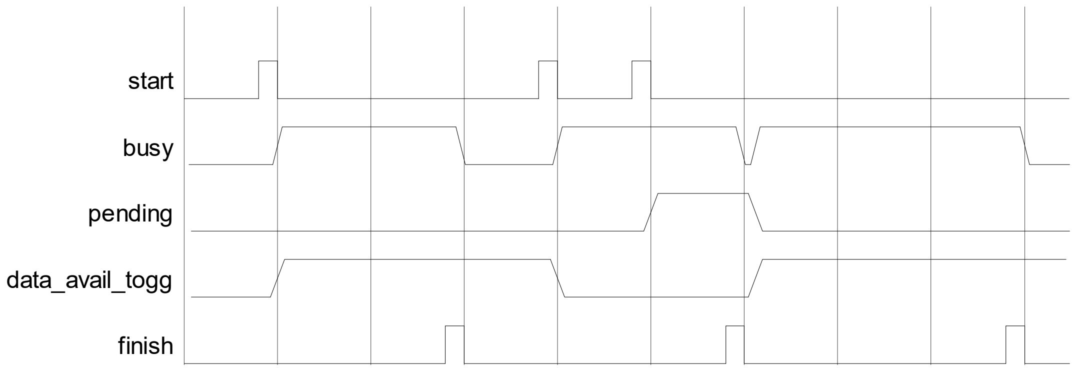


Figure 8.24 Timing Diagram for Data Synchronization Module

Full synchronization handshake takes place on all signals that are “data synchronized”. All signals that are “level synchronized” are simply passed through two destination clock registers. Both synchronization types incur additional data path latencies. However, this additional latency has no negative affect on received or transmitted data, other than to limit the serial clock (sclk) to being no faster than four-times the pclk clock for back-to-back serial communications with no idle assertion.

A serial clock faster than four-times the pclk signal does not leave enough time for a complete incoming character to be received and pushed into the receiver FIFO. However, in most cases, the pclk signal is faster than the serial clock and this should never be an issue.There is also slightly more time required after initial serial control register programming, before serial data can be transmitted or received.

The serial clock modules must have time to see new register values and reset their respective state machines. This total time is guaranteed to be no more than eight clock cycles of the slower of the two system clocks. Therefore, no data should be transmitted or received before this maximum time expires,after initial configuration.

In systems where only one clock is implemented, there are no additional latencies.

8.8.3.6.Interrupts

The assertion of the Onmicro_uart interrupt output signal (intr) occurs whenever one of the several prioritized interrupt types are enabled and active. The following interrupt types can be enabled with the IER register:

- Receiver Error
- Receiver Data Available
- Character Timeout (in FIFO mode only)
- Transmitter Holding Register Empty at/below threshold (in Programmable THRE interrupt mode)
- Modem Status
- Busy Detect Indication

8.8.3.7.Auto Flow Control

The Onmicro_uart can be configured to have a 16750-compatible Auto RTS and Auto CTS serial data flow control mode available. If FIFOs are not implemented, then this mode cannot be selected. When Auto Flow Control is not selected, none of the corresponding logic is implemented and the mode cannot be enabled, reducing overall gate counts. When Auto Flow Control mode has been selected it can be enabled with the Modem Control Register (MCR[5]).The following figure shows a block diagram of the Auto Flow Control functionality.

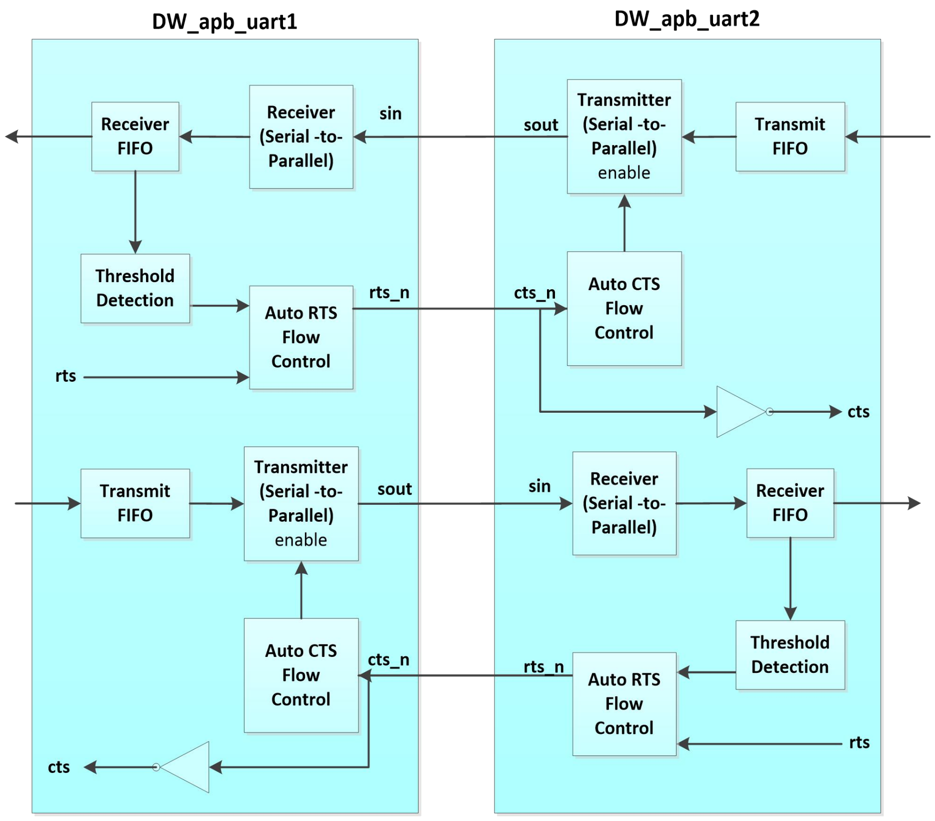


Figure 8.25 Auto Flow Control Block Diagram

Auto RTS – Becomes active when the following occurs:

- Auto Flow Control is selected during configuration
- FIFOs are implemented
- RTS (MCR[1] bit and MCR[5]bit are both set)
- FIFOs are enabled (FCR[0]) bit is set)
- SIR mode is disabled (MCR[6] bit is not set)

When Auto RTS is enabled (active), the rts_n output is forced inactive (high) when the receiver FIFO level reaches the threshold set by FCR[7:6]. When rts_n is connected to the cts_n input of another UART device, the other UART stops sending serial data until the receiver FIFO has available space (until it is completely empty).

The selectable receiver FIFO threshold values are: 1, ¼, ½, and “2 less than full”. Since one additional character may be transmitted to the Onmicro_uart after rts_n has become inactive (due to data already having entered the transmitter block in the other UART), setting the threshold to “2 less than full” allows maximum use of the FIFO with a safety zone of one character.

Once the receiver FIFO becomes completely empty by reading the Receiver Buffer Register (RBR), rts_n again becomes active (low), signalling the other UART to continue sending data. It is important to note that even if everything else is selected and the correct MCR bits are set, if the FIFOs are disabled through FCR[0] or the UART is in SIR mode (MCR[6] is set to one), Auto Flow Control is also disabled. When Auto RTS is not implemented or disabled, rts_n is controlled solely by MCR[1].The following fgure shows a timing diagram of Auto RTS operation.

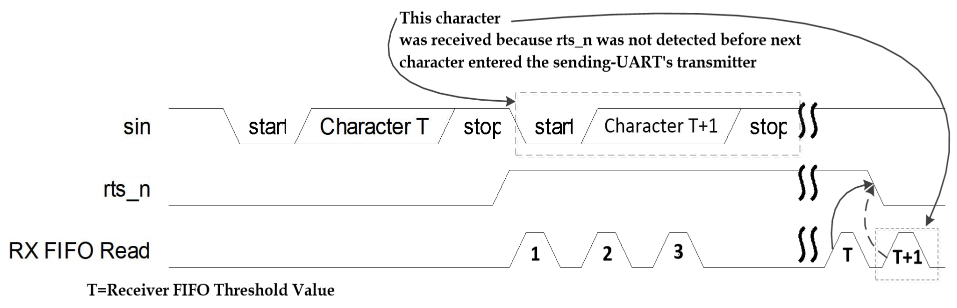

This character was received because rts_n was not detected before next character entered the sending-UART's transmitter
sin
start
Character T
stop
start
Character T+1
stop
rts_n
RX FIFO Read
1
2
3
T
T+1
T=Receiver FIFO Threshold Value

Figure 8.26 Auto RTS Timing

This character was received because rts_n was not detected before next character entered the sending-UART's transmitter T=Receiver FIFO Threshold Value Auto CTS – becomes active when the following occurs:

- Auto Flow Control is selected during configuration
- FIFOs are implemented
- AFCE (MCR[5] bit is set)
- FIFOs are enabled through FIFO Control Register FCR[0] bit
- SIR mode is disabled (MCR[6] bit is not set)

When Auto CTS is enabled (active), the Onmicro_uart transmitter is disabled whenever the cts_n input becomes inactive (high). This prevents overflowing the FIFO of the receiving UART. If the cts_n input is not inactivated before the middle of the last stop bit, another character is transmitted before the transmitter is disabled. While the transmitter is disabled, the transmitter FIFO can still be written to, and even overflowed.

Therefore, when using this mode, the following happens:

- The UART status register can be read to check if the transmit FIFO is full (USR[1] set to zero)
- The current FIFO level can be read via the TFL register
- The Programmable THRE Interrupt mode must be enabled to access the "FIFO full" status via the Line Status Register (LSR).

When using the "FIFO full" status, software can poll this before each write to the Transmitter FIFO.See "Programmable THRE Interrupt for details. When the cts_n input becomes active (low) again, transmission resumes. It is important to note that even if everything else is selected, if the FIFOs are disabled via FCR[0], Auto Flow Control is also disabled. When Auto CTS is not implemented or disabled, the transmitter is unaffected by cts_n. A Timing Diagram showing Auto CTS operation can be seen in the following figure.

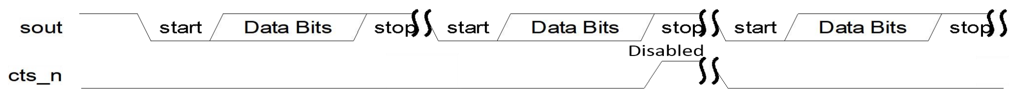


Figure 8.27 Auto CTS Timing

8.8.3.8.Programmable THRE Interrupt

The UART can be configured for a Programmable TX_HDG_EMPTY Interrupt mode in order to increase system performance;if FIFOs are not implemented, then this mode cannot be selected.

- When Programmable TX_HDG_EMPTY Interrupt mode is not selected, none of the logic is implemented and the mode cannot be enabled, reducing the overall gate counts.
- When Programmable TX_HDG_EMPTY Interrupt mode is selected, it can be enabled using the Interrupt Enable Register (INT_EN[7]).

When FIFOs and TX_HDG_EMPTY mode are implemented and enabled, the TX_HDG_EMPTY Interrupts and dma_tx_req_n are active at, and below, a programmed transmitter FIFO empty threshold level, as opposed to empty, as shown in the flowchart in the following figure.

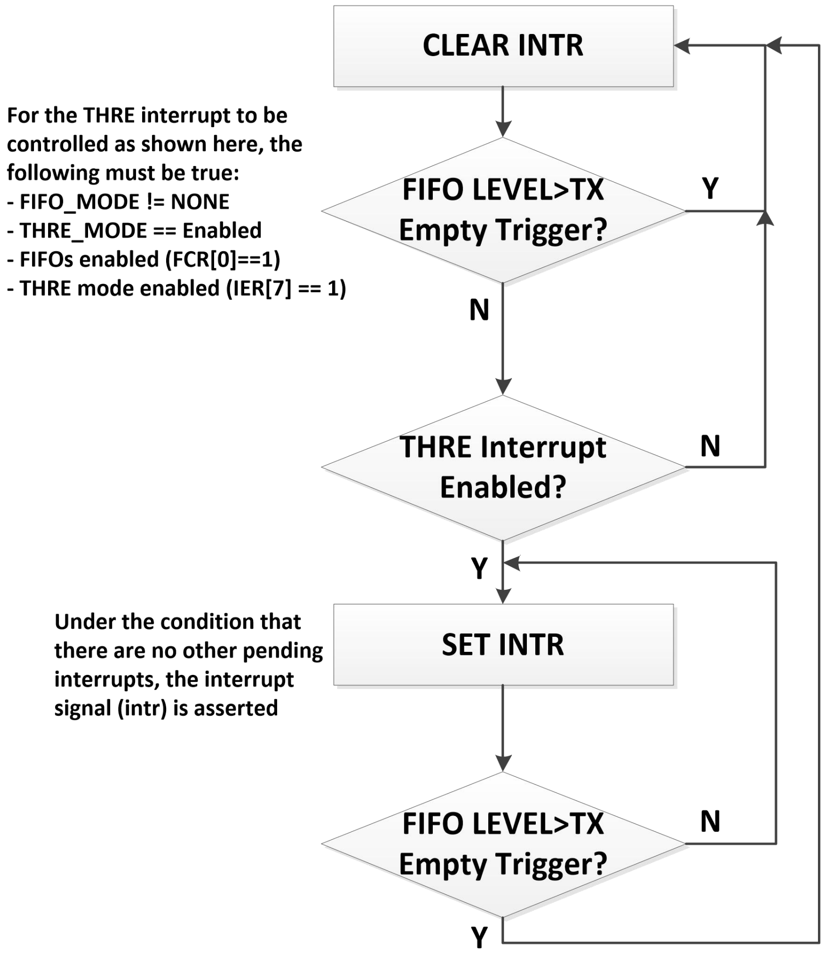

CLEAR INTR
For the THRE interrupt to be controlled as shown here, the following must be true:
- FIFO_MODE != NONE
- THRE_MODE == Enabled
- FIFOs enabled (FCR[0]==1)
- THRE mode enabled (IER[7] == 1)
FIFO LEVEL>TX Empty Trigger?
Y
N
THRE Interrupt Enabled?
N
Y
Under the condition that there are no other pending interrupts, the interrupt signal (intr) is asserted
SET INTR
FIFO LEVEL>TX Empty Trigger?
N
Y

Figure 8.28 Flowchart of Interrupt Generation for Programmable THRE Interrupt Mode

The threshold level is programmed into FIFO_CTRL[5:4]. Available empty thresholds are: empty, 2, ¼ , ½ . Selection of the best threshold value depends on the system's ability to start a new transmission sequence in a timely manner. However, one of these thresholds should be optimal for increasing system performance by preventing the transmitter FIFO from running empty.

In addition to the interrupt change, the Line Status Register (LINE_STAT[5]) also switches from indicating that the transmitter FIFO is empty to the FIFO being full. This allows software to fill the FIFO for each transmit sequence by polling LINE_STAT[5] before writing another character. The flow then allows the transmitter FIFO to be filled whenever an interrupt occurs and there is data to transmit, rather than waiting until the FIFO is completely empty. Waiting until the FIFO is empty causes a reduction in performance whenever the system is too busy to respond immediately. Further system efficiency is achieved when this mode is enabled in combination with Auto Flow Control.

Even if everything else is selected and enabled, if the FIFOs are disabled using the FIFO_CTRL[0] bit, the Programmable TX_HDG_EMPTY Interrupt mode is also disabled. When not selected or disabled, TX_HDG_EMPTY interrupts and the LSR[5] bit function normally, signifying an empty TX_HDG or FIFO. The flowchart of THRE interrupt generation when not in programmable THRE interrupt mode is shown in the following figure.

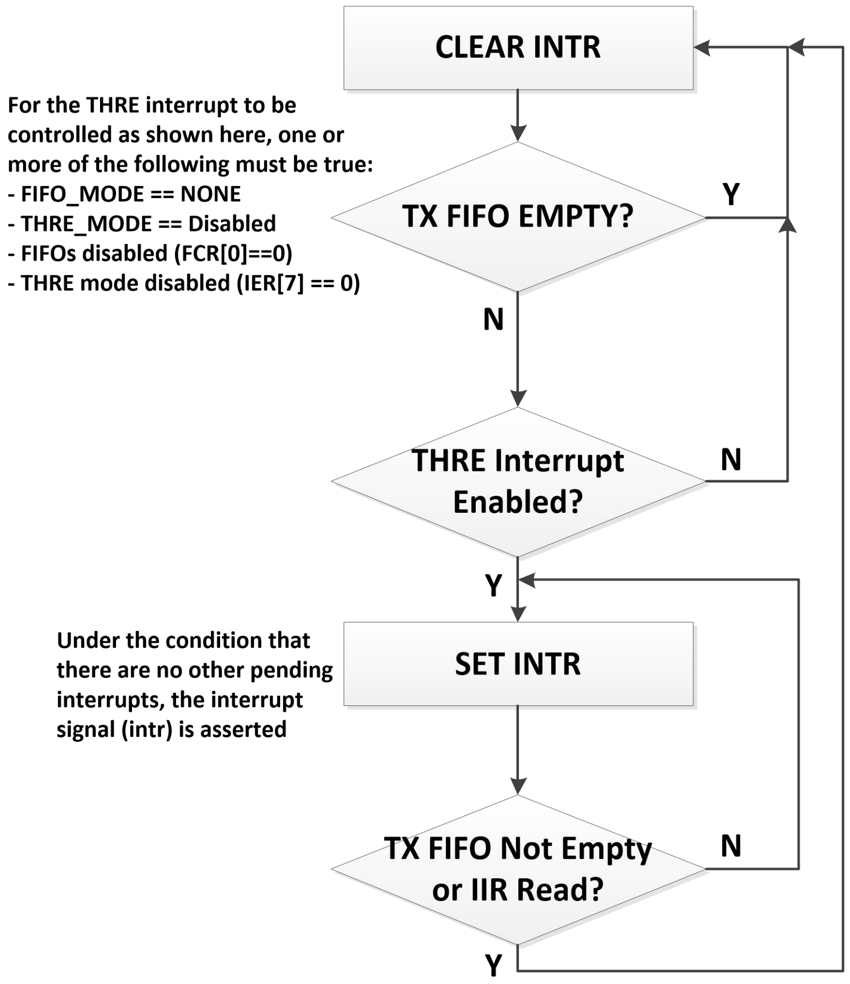

CLEAR INTR
For the THRE interrupt to be controlled as shown here, one or more of the following must be true:
- FIFO_MODE == NONE
- THRE_MODE == Disabled
- FIFOs disabled (FCR[0]==0)
- THRE mode disabled (IER[7] == 0)
TX FIFO EMPTY?
Y
N
THRE Interrupt Enabled?
N
Y
Under the condition that there are no other pending interrupts, the interrupt signal (intr) is asserted
SET INTR
TX FIFO Not Empty or IIR Read?
N
Y

Figure 8.29 Flowchart of Interrupt generation when nor in Programmable THRE Interrupt Mode

8.8.3.9.Clock Gate Enable

The Onmicro_uart can be configured to have a clock gate enable output. When the clock enable option is not selected, none of the logic is implemented, reducing the overall gate counts.

When the clock gate enable option is selected the clock gate enable signal(s) (uart_lp_req_pclk for single clock implementations or uart_lp_req_pclk and uart_lp_req_sclk for two clock implementations) is used to indicate that the transmit and receive pipeline is clear (no data), no activity has occurred, and the modem control input signals have not changed in more than one character time (the time taken to TX/RX a character) so clocks may be gated. (A character is made up of: start bit + data bits + parity (optional) + stop bit(s)). It is an indication that the UART is inactive, so clocks may be gated to put the device in a low power (lp) mode. Therefore, the following must be true for at least one character time for the assertion of the clock gate enable signal(s) to occur:

- No data in the RBR (in non-FIFO mode) or the RX FIFO is empty (in FIFO mode)
- No data in the THR (in non-FIFO mode) or the TX FIFO is empty (in FIFO mode)
- sin/sir_in and sout/sir_out_n are inactive (sin/sir_in are kept high and sout is high or sir_out_n is low) indicating no activity
- No change on the modem control input signals

Note:the clock gate enable assertion does not occur in the following modes of operation:

- Loopback mode
- FIFO access mode
- When transmitting a break

For example, assume a Onmicro_uart that is configured to have a single clock (pclk) and is programmed to transmit and receive characters of 7 bits (1 start bit, 5 data bits and 1 stop bit) and the baud clock divisor is set to 1.Therefore, the uart_lp_req_pclk signal is asserted if the transmit and receive pipeline is clear, no activity has occurred and the modem control input signals have not changed for 112 (7 × 16) pclk cycles.The following figure illustrates this example .

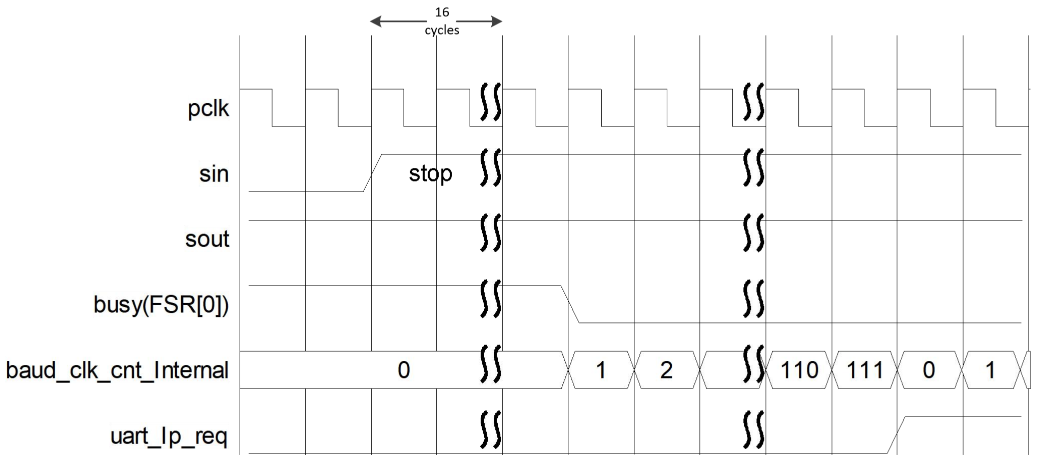


Figure 8.30 Clock Gate Enable Timing

When either of the signals sin or sir_in goes low, or a write to any of the registers is performed, or the modem control input signals have changed when the Onmicro_uart is in low power (sleep) mode, the clock gate enable signal(s) are de-asserted (as the assertion criteria are no longer met) so that the clock(s) is resumed. The time taken for the clock(s) to resume is important in the prevention of receive data synchronization problems. This is due to the fact that the Onmicro_uart RX block samples at the mid-point of each bit period (after approximately 8 baud clocks) in UART (RS323) mode and then every 16 baud clocks after that for a baud divisor of 1 that is 16 sclks (which for a single clock implementation is 16 pclks). Thus, if 8 or more sclk periods pass before the serial clock starts up again, the UART may get out of sync with the serial data it is receiving. That is, the receiver may sample into the second bit period and if it is still zero, think this is the start bit and so on. Therefore, to avoid this problem the clock should be resumed within 5 clock periods of the baud clock, which is the same as sclk if the baud divisor is set to one. This is worst case. If the divisor is greater, it gives a greater number of sclk cycles available before the clock must resume. This means a sample point at the 13 baud clock (at the latest) out of the 16 that is transmitted for each bit period of the character in non-SIR mode.

The following figure shows the timing diagram that illustrates the previous scenario. This problem is magnified in SIR mode as the pulse width is only 3/16 of a bit period (3 baud clocks, which for a divisor of 1 is 3 sclks). Hence, it could be missed completely. The clocks must resume before 3 baud clock periods elapse. If the first character received while in sleep mode is used purely for wake up reasons and the actual character value is unimportant, this may not be a problem at all.

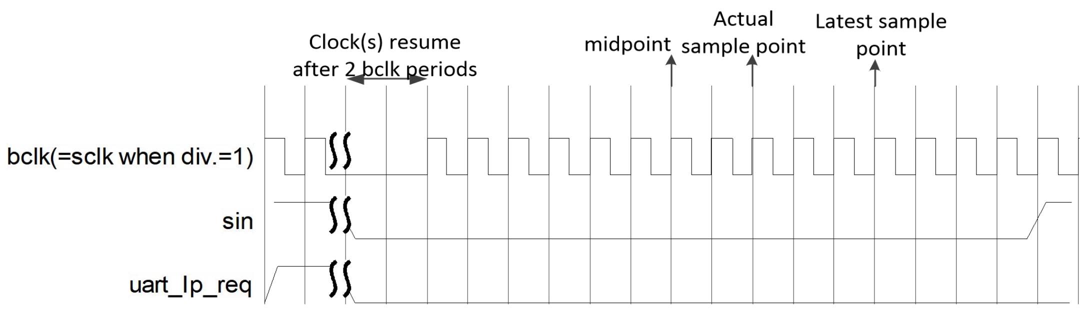


Figure 8.31 Resuming Clock(s) After Low Power Mode Timing

When the Onmicro_uart is configured to have two clocks, if the timing of the received signal is not affected by the synchronization problem, then the minimum time to receive a character, if the baud divisor is 1, is 112 sclks (1 start bit + 5 data bits + 1 stop bit = 7 × 16 =112). Therefore the pclk must be available before 112 sclk cycles pass so that the received character can be synchronized over to the pclk domain and stored in the RBR (in non-FIFO mode) or the RX FIFO (in FIFO mode).

8.8.4 UARTx Register Map

Offset	Name	Description
0x0000	RBR	Receive Buffer Register Dependencies:LCR[7]bit = 0
	THR	Transmit Holding Register Dependencies:LCR[7]bit = 0
	DLL	Divisor Latch (Low) Dependencies: LCR[7] bit = 1
0x0004	DLH	Divisor Latch (High) Dependencies: LCR[7] bit = 1

	IER	Interrupt Enable Register Dependencies: LCR[7] bit = 0
0x0008	IIR	Interrupt Identification Register
	FCR	FIFO Control Register
0x000C	LCR	Line Control Register
0x0010	MCR	Modem Control Register
0x0014	LSR	Line Status Register
0x0018	MSR	Modem Status Register
0X001C	SCR	Scratchpad Register
0x0020	LPDLL	Low Power Divisor Latch (Low) Register
0x0024	LPDLH	Low Power Divisor Latch (High) Register
0x0028	ISO7816_CTRL0	ISO7816 Control Register (only valid in UART1)
0x002C	ISO7816_CTRL1	ISO7816 Control Register (only valid in UART1)
0x0030--0x006C	SRBR	Shadow Receive Buffer Register Dependencies: LCR[7] bit = 0
	STHR	Shadow Transmit Holding Register Dependencies: LCR[7] bit = 0
0x0070	FAR	FIFO Access Register
0x0074	TFR	Transmit FIFO Read
0x0078	RFW	Receive FIFO Write
0x007C	USR	UART Status Register
0x0080	TFL	Transmit FIFO Level Width: FIFO_ADDR_WIDTH + 1
0x0084	RFL	Receive FIFO Level Width: FIFO_ADDR_WIDTH + 1
0x0088	SRR	Software Reset Register
0x008C	SRTS	Shadow Request to Send
0x0090	SBCR	Shadow Break Control Register
0x0094	SDMAM	Shadow DMA Mode
0x0098	SFE	Shadow FIFO Enable
0x009C	SRT	Shadow RCVR Trigger
0x00A0	STET	Shadow TX Empty Trigger
0x00A4	HTX	Halt TX
0x00A8	DMASA	DMA Software Acknowledge
0x00AC--0x00F0		
0x00F4	CPR	Component Parameter Register
0x00F8	UCV	UART Component Version
0x00FC	CTR	Component Type Register

RBR address offset: 0x0000

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	R	0x0	RBR	Data byte received on the serial input port (sin) in UART mode, or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line Status Register (LCR) is set. If in non-FIFO mode (FIFO_MODE == NONE) or FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an over-run error. If in FIFO mode (FIFO_MODE != NONE) and FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO is preserved, but any incoming data are lost and an overrun error occurs.

THR address offset: 0x0000

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	W	0x0	THR	Data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If in non-FIFO mode or FIFOs are disabled (FCR[0] = 0) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If in FIFO mode and FIFOs are enabled (FCR[0] = 1) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.

DLH address offset: 0x0004

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	RW	0x0	DLH	Upper 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. The output baud rate is equal to the serial clock (pclk if one clock design, sclk if two clock design (CLOCK_MODE == Enabled)) frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock freq) / (16 * divisor). Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications occur. Also, once the DLH is set, at least 8 clock cycles of the slowest Onmicro_uart clock should be allowed to pass before transmitting or receiving data.

DLL address offset: 0x0000

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	RW	0x0	DLL	Lower 8 bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. The output baud rate is equal to the serial clock (pclk if one clock design, sclk if two clock design (CLOCK_MODE == Enabled)) frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock freq) / (16 * divisor). Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications occur. Also, once the DLL is set, at least 8 clock cycles of the slowest Onmicro_uart clock should be allowed to pass before transmitting or receiving data.

IER address offset: 0x0004

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7	RW	0x0	PTIME	This is used to enable/disable the generation of THRE Interrupt 0 = disabled 1 = enabled

6:4	N/A	0x0	N/A	reserved
3	RW	0x0	EDSSI	Enable Modem Status Interrupt. This is used to enable/disable the generation of Modem Status Interrupt. This is the fourth highest priority interrupt. 0 = disabled 1 = enabled
2	RW	0x0	ELSI	Enable Receiver Line Status Interrupt. This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt 0 = disabled 1 = enabled
1	RW	0x0	ETBEI	Enable Transmit Holding Register Empty Interrupt. This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt. This is the third highest priority interrupt. 0 = disabled 1 = enabled
0	RW	0x0	ERBFI	Enable Received Data Available Interrupt. This is used to enable/disable the generation of Received Data Available Interrupt and the Character Timeout Interrupt (if in FIFO mode and FIFOs enabled). These are the second highest priority interrupts. 0 = disabled 1 = enabled

IIR address offset: 0x0008

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:6	R	0x0	FIFOSE	FIFOs Enabled. This is used to indicate whether the FIFOs are enabled or disabled. 00 = disabled 11 = enabled
5:4	N/A	0x0	N/A	reserved
3:0	R	0x1	IID	Interrupt ID. This indicates the highest priority pending interrupt which can be one of the following types 0000 = modem status 0001 = no interrupt pending 0010 = THR empty 0100 = received data available

				0110 = receiver line status 0111 = busy detect 1100 = character timeout

FCR address offset: 0x0008

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:6	W	0x0	RT	RCVR Trigger. This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated. In auto flow control mode it is used to determine when the rts_n signal is de-asserted. For details on DMA support, The following trigger levels are supported: 00 = 1 character in the FIFO 01 = FIFO/4 full 10 = FIFO/2 full 11 = FIFO 2 less than full
5:4	W	0x0	TET	TX Empty Trigger. This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active. For details on DMA support, The following trigger levels are supported: 00 = FIFO empty 01 = 2 characters in the FIFO 10 = FIFO/4 full 11 = FIFO/2 full
3	W	0x0	DMAM	DMA Mode. For details on DMA support, 0 = mode 0； 1 = mode 1
2	W	0x0	XFIFOR	XMIT FIFO Reset. This resets the control portion of the transmit FIFO and treats the FIFO as empty. This bit is 'self-clearing'.
1	W	0x0	RFIFOR	RCVR FIFO Reset. This resets the control portion of the receive FIFO and treats the FIFO as empty. This bit is 'self-clearing'.
0	W	0x0	FIFOE	FIFO Enable. This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs.Whenever the value of this bit is changed both the XMIT and RCVR controller portion of FIFOs is reset.

LCR address offset: 0x000C

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved

7	RW	0x0	DLAB	Divisor Latch Access Bit. This bit is used to enable reading and writing of the Divisor Latch register (DLL and DLH) to set the baud rate of the UART.
6	RW	0x0	BC	Break Control Bit.This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared. If SIR_MODE == Enabled and active (MCR[6] set to one) the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver and the sir_out_n line is forced low.
5	N/A	0x0	N/A	reserved
4	RW	0x0	EPS	Even Parity Select.If UART_16550_COMPATIBLE == NO, then writeable only when UART is not busy (USR[0] is zero); otherwise always writable, always readable. This is used to select between even and odd parity, when parity is enabled (PEN set to one). If set to one, an even number of logic 1s is transmitted or checked. If set to zero, an odd number of logic 1s is transmitted or checked. 0 = mode 0 1 = mode 1
3	RW	0x0	PEN	Parity Enable.If UART_16550_COMPATIBLE == NO, then writeable only when UART is not busy (USR[0] is zero); otherwise always writable, always readable. This bit is used to enable and disable parity generation and detection in transmitted and received serial character respectively 0 = parity disabled 1 = parity enabled
2	RW	0x0	STOP	Number of stop bits. If UART_16550_COMPATIBLE == NO, then writeable only when UART is not busy (USR[0] is zero); otherwise always writable, always readable. This is used to select the number of stop bits per character that the peripheral transmits and receives. If set to zero, one stop bit

				is transmitted in the serial data. If set to one and the data bits are set to 5 (LCR[1:0] set to zero) one and a half stop bits is transmitted. Otherwise, two stop bits are transmitted. Note that regardless of the number of stop bits selected, the receiver checks only the first stop bit. 0 = 1 stop bit 1 = 1.5 stop bits (DLS==0) 1 = 2 stop bits (DLS!=0)
1:0	RW	0x0	DLS	Data Length Select This is used to select the number of data bits per character that the peripheral transmits and receives. The number of bit that may be selected areas follows: 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits

MCR address offset: 0x0010

Bit	R/W	Reset	Name	Description
31:7	N/A	0x0	N/A	reserved
6	RW	0x0	SIRE	SIR Mode Enable. 0 = disable 1 = enable
5	RW	0x0	AFCE	Auto Flow Control Enable. 0 = Auto Flow Control Mode disabled 1 = Auto Flow Control Mode enabled
4	RW	0x0	LB	LoopBack Bit. This is used to put the UART into a diagnostic mode for test purposes. If operating in UART mode (SIR_MODE != Enabled or not active, MCR[6] set to zero), data on the sout line is held high, while serial data output is looped back to the sin line, internally. In this mode all the interrupts are fully functional. Also, in loopback mode, the modem control inputs (dsr_n, cts_n, ri_n, dcd_n) are disconnected and the modem control outputs (dtr_n, rts_n, out1_n, out2_n) are looped back to the inputs, internally. If operating in infrared mode (SIR_MODE == Enabled AND active, MCR[6] set to one), data on the sir_out_n line is held low, while serial data

				output is inverted and looped back to the sir_in line.
3	RW	0x0	OUT2	OUT2. This is used to directly control the user-designated Output2 (out2_n) output. The value written to this location is inverted and driven out on out2_n, that is: 0 = out2_n de-asserted (logic 1) 1 = out2_n asserted (logic 0)
2	RW	0x0	OUT1	OUT1. This is used to directly control the user-designated Output1 (out1_n) output. The value written to this location is inverted and driven out on out1_n, that is: 0 = out1_n de-asserted (logic 1) 1 = out1_n asserted (logic 0)
1	RW	0x0	RTS	Request to Send. This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the UART is ready to exchange data. When Auto RTS Flow Control is not enabled (MCR[5] set to zero), the rts_n signal is set low by programming MCR[1] (RTS) to a high.In Auto Flow Control, AFCE_MODE == Enabled and active (MCR[5] set to one) and FIFOs enable (FCR[0] set to one), the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold). The rts_n signal is de-asserted when MCR[1] is set low. Note that in Loopback mode (MCR[4] set to one), the rts_n output is held inactive high while the value of this location is internally looped back to an input.
0	RW	0x0	DTR	Data Terminal Ready. This is used to directly control the Data Terminal Ready (dtr_n) output. The value written to this location is inverted and driven out on dtr_n, that is: 0 = dtr_n de-asserted (logic 1) 1 = dtr_n asserted (logic 0)

LSR address offset: 0x0014

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7	R	0x0	RFE	Receiver FIFO Error bit.This bit is only relevant when FIFO_MODE != NONE AND FIFOs are

				enabled (FCR[0] set to one). This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO. 0 = no error in RX FIFO 1 = error in RX FIFO
6	R	0x1	TEMT	Transmitter Empty bit. If in FIFO mode (FIFO_MODE != NONE) and FIFOs enabled (FCR[0] set to one), this bit is set whenever the Transmitter Shift Register and the FIFO are both empty. If in non-FIFO mode or FIFOs are disabled, this bit is set whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty.
5	R	0x1	THRE	Transmit Holding Register Empty bit.If THRE_MODE_USER == Disabled or THRE mode is disabled (IER[7] set to zero) and regardless of FIFO's being implemented/enabled or not, this bit indicates that the THR or TX FIFO is empty. This bit is set whenever data is transferred from the THR or TX FIFO to the transmitter shift register and no new data has been written to the THR or TX FIFO. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. If THRE_MODE_USER == Enabled AND FIFO_MODE != NONE and both modes are active (IER[7] set to one and FCR[0] set to one respectively), the functionality is switched to indicate the transmitter FIFO is full, and no longer controls THRE interrupts, which are then controlled by the FCR[5:4] threshold setting.
4	R	0x0	BI	Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data. If in UART mode (SIR_MODE == Disabled), it is set whenever the serial input, sin, is held in a logic '0' state for longer than the sum of start time + data bits + parity + stop bits. If in infrared mode (SIR_MODE == Enabled), it is set whenever the serial input, sir_in, is continuously pulsed to logic '0' for longer than the sum of start time + data bits + parity + stop bits. A break condition on serial input causes one and only one character, consisting of all zeros, to be received by the UART.

				In the FIFO mode, the character associated with the break condition is carried through the FIFO and is revealed when the character is at the top of the FIFO. Reading the LSR clears the BI bit. In the non-FIFO mode, the BI indication occurs immediately and persists until the LSR is read.
3	R	0x0	FE	Framing Error bit.This is used to indicate the occurrence of a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data. In the FIFO mode, since the framing error is associated with a character received, it is revealed when the character with the framing error is at the top of the FIFO. When a framing error occurs, the UART tries to resynchronize. It does this by assuming that the error was due to the start bit of the next character and then continues receiving the other bit i.e. data, and/or parity and stop. It should be noted that the Framing Error (FE) bit (LSR[3]) is set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]). 0 = no framing error 1 =framing error
2	R	0x0	PE	Parity Error bit.This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PEN) bit (LCR[3]) is set. In the FIFO mode, since the parity error is associated with a character received, it is revealed when the character with the parity error arrives at the top of the FIFO. It should be noted that the Parity Error (PE) bit (LSR[2]) is set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]). 0 = no parity error 1 = parity error
1	R	0x0	OE	Overrun error bit.This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read. In the non-FIFO mode, the OE bit is set when a

				new character arrives in the receiver before the previous character was read from the RBR. When this happens, the data in the RBR is overwritten. In the FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver. The data in the FIFO is retained and the data in the receive shift register is lost. 0 = no overrun error 1 = overrun error
0	R	0x0	DR	Data Ready bit. This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO. 0 = no data ready 1 = data ready

MSR address offset: 0x0018

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7	R	0x0	DCD	Data Carrier Detect.This is used to indicate the current state of the modem control line dcd_n. This bit is the complement of dcd_n. When the Data Carrier Detect input (dcd_n) is asserted it is an indication that the carrier has been detected by the modem or data set. 0 = dcd_n input is de-asserted (logic 1) 1 = dcd_n input is asserted (logic 0)
6	R	0x0	RI	Ring Indicator. This is used to indicate the current state of the modem control line ri_n. This bit is the complement of ri_n. When the Ring Indicator input (ri_n) is asserted it is an indication that a telephone ringing signal has been received by the modem or data set. 0 = ri_n input is de-asserted (logic 1) 1 = ri_n input is asserted (logic 0)
5	R	0x0	DSR	Data Set Ready.This is used to indicate the current state of the modem control line dsr_n. This bit is the complement of dsr_n. When the Data Set Ready input (dsr_n) is asserted it is an indication that the modem or data set is ready to establish communications with the OM_uart. 0 = dsr_n input is de-asserted (logic 1) 1 = dsr_n input is asserted (logic 0)
4	R	0x0	CTS	Clear to Send.This is used to indicate the current

				state of the modem control line cts_n. This bit is the complement of cts_n. When the Clear to Send input (cts_n) is asserted it is an indication that the modem or data set is ready to exchange data with the OM_uart. 0 = cts_n input is de-asserted (logic 1) 1 = cts_n input is asserted (logic 0)
3	R	0x0	DDCD	Delta Data Carrier Detect.This is used to indicate that the modem control line dcd_n has changed since the last time the MSR was read. 0 = no change on dcd_n since last read of MSR 1 = change on dcd_n since last read of MSR Reading the MSR clears the DDCD bit. In Loopback Mode (MCR[4] = 1), DDCD reflects changes on MCR[3] (Out2). Note, if the DDCD bit is not set and the dcd_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDCD bit is set when the reset is removed if the dcd_n signal remains asserted.
2	R	0x0	TERI	Trailing Edge of Ring Indicator.This is used to indicate that a change on the input ri_n (from an active-low to an inactive-high state) has occurred since the last time the MSR was read. 0 = no change on ri_n since last read of MSR 1 = change on ri_n since last read of MSR
1	R	0x0	DDSR	Delta Data Set Ready.This is used to indicate that the modem control line dsr_n has changed since the last time the MSR was read. 0 = no change on dsr_n since last read of MSR 1 = change on dsr_n since last read of MSR Reading the MSR clears the DDSR bit. In Loopback Mode (MCR[4] = 1), DDSR reflects changes on MCR[0] (DTR). Note, if the DDSR bit is not set and the dsr_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDSR bit is set when the reset is removed if the dsr_n signal remains asserted.
0	R	0x0	DCTS	Delta Clear to Send.This is used to indicate that the modem control line cts_n has changed since the last time the MSR was read. 0 = no change on cts_n since last read of MSR

				1 = change on cts_n since last read of MSR Reading the MSR clears the DCTS bit. In Loopback Mode (MCR[4] = 1), DCTS reflects changes on MCR[1] (RTS). Note, if the DCTS bit is not set and the cts_n signal is asserted (low) and a reset occurs (software or otherwise), then the DCTS bit is set when the reset is removed if the cts_n signal remains asserted.

SCR address offset: 0x001C

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	Reserved and read as zero
7:0	RW	0x0	Scratchpad Register	This register is for programmers to use as a temporary storage space. It has no defined purpose in the OM_uart.

LPDLL address offset: 0x0020

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	Reserved
7:0	RW	0x0	LPDLL	This register makes up the lower 8-bits of a 16-bit, read/write, Low Power Divisor Latch register that contains the baud rate divisor for the UART,which must give a baud rate of 115.2K. This is required for SIR Low Power (minimum pulse width) detection at the receiver. If UART_16550_COMPATIBLE == No, then this register may only be accessed when the DLAB bit (LCR[7]) is set and the UART is not busy (USR[0] is zero); otherwise this register may be accessed only when the DLAB bit (LCR[7]) is set. The output low-power baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows: Low power baud rate = (serial clock frequency)/(16* divisor) Therefore, a divisor must be selected to give a baud rate of 115.2K. NOTE: When the Low Power Divisor Latch registers (LPDLL and LPDLH) are set to 0, the low-power baud clock is disabled and no low-power pulse detection (or any pulse detection) occurs at the receiver. Also, once the

				LPDLL is set, at least eight clock cycles of the slowest Onmicro_uart clock should be allowed to pass before transmitting or receiving data.

LPDLH address offset: 0x0024

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	RW	0x0	LPDLL	This register makes up the upper 8-bits of a 16-bit, read/write, Low Power Divisor Latch register that contains the baud rate divisor for the UART,which must give a baud rate of 115.2K. This is required for SIR Low Power (minimum pulse width) detection at the receiver. If UART_16550_COMPATIBLE == No, then this register may only be accessed when the DLAB bit (LCR[7]) is set and the UART is not busy (USR[0] is zero); otherwise this register may be accessed only when the DLAB bit (LCR[7]) is set. The output low-power baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows: Low power baud rate = (serial clock frequency)/(16* divisor) Therefore, a divisor must be selected to give a baud rate of 115.2K. NOTE: When the Low Power Divisor Latch registers (LPDLL and LPDLH) are set to 0, the low-power baud clock is disabled and no low-power pulse detection (or any pulse detection) occurs at the receiver. Also, once the LPDLH is set, at least eight clock cycles of the slowest Onmicro_uart clock should be allowed to pass before transmitting or receiving data.

FAR address offset: 0x0070

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	Reserved and read as zero
7:0	RW	0x0	FIFO Access Register	Writes have no effect when FIFO_ACCESS == No, always readable. This register is use to enable a FIFO access mode for testing, so that the receive FIFO can be written by the master and the transmit FIFO can be read by the master when FIFOs are implemented and enabled. When

				FIFOs are not implemented or not enabled it allows the RBR to be written by the master and the THR to be read by the master. 0 = FIFO access mode disabled 1 = FIFO access mode enabled Note, that when the FIFO access mode is enabled/disabled, the control portion of the receive FIFO and transmit FIFO is reset and the FIFOs are treated as empty.

ISO7816_CTRL0 address offset: 0x0028

Bit	R/W	Reset	Name	Description
31:13	N/A	0x0	N/A	reserved
12	R	0x0	tx_done	TX is done
11:4	RW	0x0	sample_dly	sample_dly is used to adjust the sample timing of SIN
3	RW	0x0	retrans_en	parity error re-trans enable
2	RW	0x0	trx_oen	0: TX 1: RX
1	RW	0x0	nack_enable	noack is enable
0	RW	0x0	iso7816_en	ISO7816 is enable

ISO7816_CTRL1 address offset: 0x002C

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:8	R	0x0	tx_perr_cnt	tx parity error counter
7:0	R	0x0	rx_perr_cnt	rx parity error counter

SRBR address offset: 0x0030--0x006C

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	R	0x0	SRBR	This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If in non-FIFO mode (FIFO_MODE == NONE) or FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an overrun

				error. If in FIFO mode (FIFO_MODE != NONE) and FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO are preserved, but any incoming data is lost. An overrun error also occurs.

STHR address offset: 0x0030--0x006C

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	W	0x0	STHR	This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If in non-FIFO mode or FIFOs are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If in FIFO mode and FIFOs are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.

FAR address offset: 0x0070

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	Reserved and read as zero
7:0	RW	0x0	FIFO Access Register	Writes have no effect when FIFO_ACCESS == No, always readable. This register is use to enable a FIFO access mode for testing, so that the receive FIFO can be written by the master and the transmit FIFO can be read by the master when FIFOs are implemented and enabled. When FIFOs are not implemented or not enabled it allows the RBR to be written by the master and the THR to be read by

				the master. 0 = FIFO access mode disabled 1 = FIFO access mode enabled Note, that when the FIFO access mode is enabled /disabled, the control portion of the receive FIFO and transmit FIFO is reset and the FIFOs are treated as empty.

TFR address offset: 0x0074

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	Reserved and read as zero
7:0	R	0x0	Transmit FIFO Read	Transmit FIFO Read. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFOs are implemented and enabled, reading this register gives the data at the top of the transmit FIFO. Each consecutive read pops the transmit FIFO and gives the next data value that is currently at the top of the FIFO. When FIFOs are not implemented or not enabled, reading this register gives the data in the THR.

RFW address offset: 0x0078

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	Reserved and read as zero
9	W	0x0	RFFE	Receive FIFO Framing Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFOs are implemented and enabled, this bit is used to write framing error detection information to the receive FIFO. When FIFOs are not implemented or not enabled, this bit is used to write framing error detection information to the RBR.
8	W	0x0	RFPE	Receive FIFO Parity Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFOs are implemented and enabled, this bit is used to write parity error detection information to the receive FIFO. When FIFOs are not implemented or not enabled, this bit is used to write parity error detection information to the RBR.
7:0	W	0x0	RFWD	Receive FIFO Write Data. These bits are only valid when FIFO access mode is enabled (FAR[0]

				is set to one). When FIFOs are implemented and enabled, the data that is written to the RFWD is pushed into the receive FIFO. Each consecutive write pushes the new data to the next write location in the receive FIFO. When FIFOs are not implemented or not enabled, the data that is written to the RFWD is pushed into the RBR.

USR address offset: 0x007C

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4	R	0x0	RFF	Receive FIFO Full.This bit is only valid when FIFO_STAT == YES. This is used to indicate that the receive FIFO is completely full. 0 = Receive FIFO not full 1 = Receive FIFO Full
3	R	0x0	RFNE	Receive FIFO Not Empty.This bit is only valid when FIFO_STAT == YES. This is used to indicate that the receive FIFO contains one or more entries. 0 = Receive FIFO is empty 1 = Receive FIFO is not empty
2	R	0x1	TFE	Transmit FIFO Empty.This bit is only valid when FIFO_STAT == YES. This is used to indicate that the transmit FIFO is completely empty. 0 = Transmit FIFO is not empty 1 = Transmit FIFO is empty
1	R	0x1	TFNF	Transmit FIFO Not Full.This bit is only valid when FIFO_STAT == YES. This is used to indicate that the transmit FIFO in not full. 0 = Transmit FIFO is full 1 = Transmit FIFO is not full
0	R	0x0	BUSY	UART Busy.This bit is valid only when UART_16550_COMPATIBLE == NO and indicates that a serial transfer is in progress, ; when cleared, indicates that the Onmicro_uart is idle or inactive. 0 = uart is idle or inactive 1 = uart is busy (actively transferring data)

TFL address offset: 0x0080

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4:0	R	0x0	TFL	Transmit FIFO Level. This is indicates the number

				of data entries in the transmit FIFO.

RFL address offset: 0x0084

Bit	R/W	Reset	Name	Description
31:5	N/A	0x0	N/A	reserved
4:0	R	0x0	RFL	Receive FIFO Level. This is indicates the number of data entries in the receive FIFO.

SRR address offset: 0x0088

Bit	R/W	Reset	Name	Description
31:3	N/A	0x0	N/A	reserved
2	W	0x0	XFR	XMIT FIFO Reset.This is a shadow register for the XMIT FIFO Reset bit (FCR[2]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the transmit FIFO. This resets the control portion of the transmit FIFO and treats the FIFO as empty. This also de-asserts the DMA TX request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA == YES). Note that this bit is 'self-clearing'. It is not necessary to clear this bit.
1	W	0x0	RFR	RCVR FIFO Reset.This is a shadow register for the RCVR FIFO Reset bit (FCR[1]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the receive FIFO This resets the control portion of the receive FIFO and treats the FIFO as empty. This also de-asserts the DMA RX request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA == YES). Note that this bit is 'self-clearing'. It is not necessary to clear this bit.
0	W	0x0	UR	UART Reset. This asynchronously resets the uart and synchronously removes the reset assertion.For a two clock implementation both pclk and sclk domains are reset.

SRTS address offset: 0x008C

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	SRTS	Shadow Request to Send.This is a shadow

				register for the RTS bit (MCR[1]), this can be used to remove the burden of having to performing a read-modify-write on the MCR. This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the Onmicro_uart is ready to exchange data. When Auto RTS Flow Control is not enabled (MCR[5] = 0), the rts_n signal is set low by programming MCR[1] (RTS) to a high. In Auto Flow Control, AFCE_MODE == Enabled and active (MCR[5] = 1) and FIFOs enable (FCR[0] = 1), the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold). Note that in Loop back mode (MCR[4] = 1), the rts_n output is held inactive-high while the value of this location is internally looped back to an input.

SBCR address offset: 0x0090

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	SBCR	Shadow Break Control Bit.This is a shadow register for the Break bit (LCR[6]), this can be used to remove the burden of having to performing a read modify write on the LCR. This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loop back Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared. If SIR_MODE == Enabled and active (MCR[6] = 1) the sir_out_n line is continuously pulsed. When in Loop back Mode, the break condition is internally looped back to the receiver.

SDMAM address offset: 0x0094

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	SDMAM	Shadow DMA Mode. This is a shadow register for the DMA mode bit (FCR[3]). This can be used to

				remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the DMA Mode bit gets updated. This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals when additional DMA handshaking signals are not selected (DMA_EXTRA == NO). 0 = mode 0 1 = mode 1

SFE address offset: 0x0098

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	SFE	Shadow FIFO Enable.This is a shadow register for the FIFO enable bit (FCR[0]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the FIFO enable bit gets updated.This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs. If this bit is set to zero (disabled) after being enabled then both the XMIT and RCVR controller portion of FIFOs are reset.

SRT address offset: 0x009C

Bit	R/W	Reset	Name	Description
31:2	N/A	0x0	N/A	reserved
1:0	RW	0x0	SRT	Shadow RCVR Trigger. This is a shadow register for the RCVR trigger bits (FCR[7:6]).This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the RCVR trigger bit gets updated. This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated. It also determines when the dma_rx_req_n signal is asserted when DMA Mode (FCR[3]) = 1. The following trigger levels are supported: 00 = 1 character in the FIFO 01 = FIFO ¼ full 10 = FIFO ½ full 11 = FIFO 2 less than full

STET address offset: 0x00A0

Bit	R/W	Reset	Name	Description
31:2	N/A	0x0	N/A	reserved
1:0	RW	0x0	STET	Shadow TX Empty Trigger. This is a shadow register for the TX empty trigger bits (FCR[5:4]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the TX empty trigger bit gets updated. This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active. The following trigger levels are supported: 00 = FIFO empty 01 = 2 characters in the FIFO 10 = FIFO ¼ full 11 = FIFO ½ full Dependencies: Writes have no effect when THRE_MODE_USER = = Disabled.

HTX address offset: 0x00A4

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	HTX	This register is used to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFOs are implemented and enabled. 0 = Halt TX disabled 1 = Halt TX enabled Note, if FIFOs are implemented and not enabled, the setting of the halt TX register has no effect on operation. Dependencies: Writes have no effect when FIFO_MODE == None.

DMASA address offset: 0x00A8

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	W	0x0	DMASA	This register is use to perform a DMA software acknowledge if a transfer needs to be terminated due to an error condition. For example, if the DMA disables the channel, then the Onmicro_uart should clear its request. This causes the TX

				request, TX single, RX request and RX single signals to de-assert. Note that this bit is 'self-clearing'. It is not necessary to clear this bit.

CPR address offset: 0x00F4

Bit	R/W	Reset	Name	Description
31:24	N/A	0x0	N/A	Reserved and read as zero
23:16	R	0x0	FIFO_MODE	0x00 = 0 0x01 = 16 0x02 = 32 0x80 = 2048 0x81- 0xff = reserved
15:14	R	0x0	N/A	Reserved and read as zero
13	R	0x0	DMA_EXTRA	0 = FALSE 1 = TRUE
12	R	0x0	UART_ADD_ENCODED_PARAMS	0 = FALSE 1 = TRUE
11	R	0x0	SHADOW	0 = FALSE 1 = TRUE
10	R	0x0	FIFO_STAT	0 = FALSE 1 = TRUE
9	R	0x0	FIFO_ACCESS	0 = FALSE 1 = TRUE
8	R	0x0	ADDITIONAL_FEAT	0 = FALSE 1 = TRUE
7	R	0x0	SIR_LP_MODE	0 = FALSE 1 = TRUE
6	R	0x0	SIR_MODE	0 = FALSE 1 = TRUE
5	R	0x0	THRE_MODE	0 = FALSE 1 = TRUE
4	R	0x0	AFCE_MODE	0 = FALSE 1 = TRUE
3:2	N/A	0x0	N/A	Reserved and read as zero
1:0	R	0x0	APB_DATA_WIDTH	00 = 8 bits 01 = 16 bits 10 = 32 bits 11 = reserved

UCV address offset: 0x00F8

Bit	R/W	Reset	Name	Description
31:0	R	See the releases table in the AMBA	UART Component	ASCII value for each number in the version, followed by *. For example

		2 release notes.	Version	32_30_31_2A represents the version 2.01*

CTR address offset: 0x00FC

Bit	R/W	Reset	Name	Description
31:0	R	0x44570110	Peripheral ID	This register contains the peripherals identification code.

8.9 SPIx

8.9.1 Introduction

The serial peripheral interface (SPI) allows half/ full-duplex, synchronous, serial communication with external devices. The interface can be configured as the master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

It may be used for a variety of purposes, including simplex synchronous transfers on two lines with a possible bidirectional data line or reliable communication using CRC checking.

8.9.2 Main Features

- Master or slave operation
- Multimaster mode capability
- Faster communication for both master and slave
- NSS management by hardware or software for both master and slave
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- Hardware CRC feature for reliable communication:
    - CRC value can be transmitted as last byte in Tx mode
    - Automatic CRC error checking for last received byte
- Master mode fault, overrun and CRC error flags with interrupt capability
- 1-byte transmission and reception buffer with DMA capability:Tx and Rx requests

8.9.3 Function Description

8.9.3.1.Block Diagram

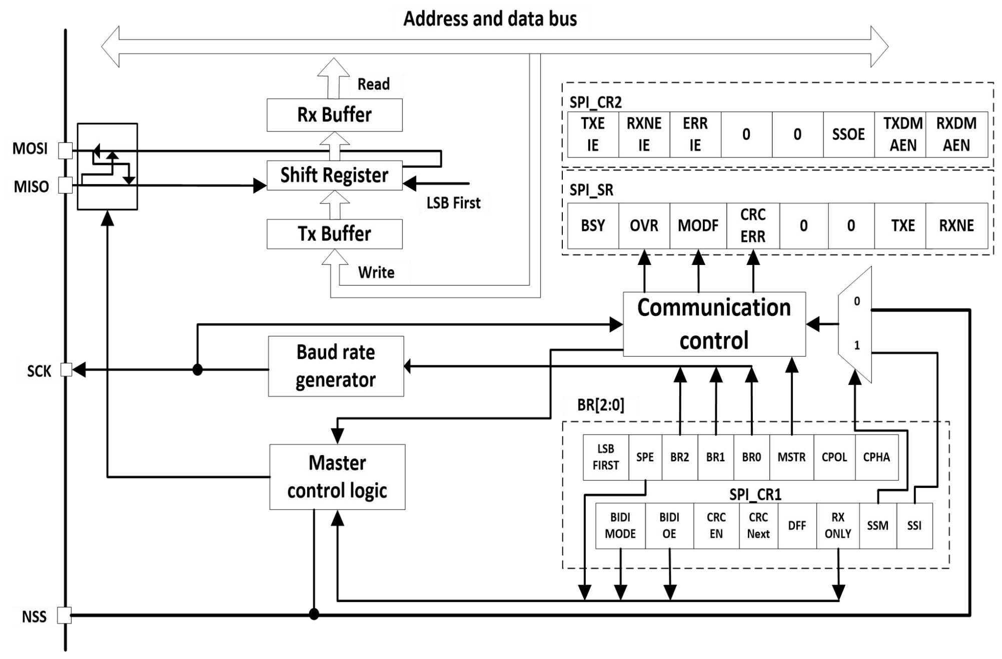


Figure 8.32 SPI block diagram

8.3.9.2.General description

Usually, the SPI is connected to external devices through four pins:

- MISO: Master In / Slave Out data.This pin can be used to transmit data in slave mode and receive data in master mode.
- MOSI: Master Out / Slave In data.This pin can be used to transmit data in master mode and receive data in slave mode.
- SCK:Serial Clock output for SPI masters and input for SPI slaves.
- NSS:Slave select. This is an optional pin to select a slave device.This pin acts as a 'chip select' to let the SPI master communicate with slaves individually and to avoid contention on the data lines.Slave NSS inputs can be driven by standard IO ports on the master device.The NSS pin may also be used as an output if enabled (SSOE bit) and driven low if the SPI is in master configuration.In this manner,all NSS pins from devices connected to the Master NSS pin see a low level and become slaves when they are configured in NSS hardware mode. When configured in master mode with NSS configured as an input (MSTR=1 and SSOE=0) and if NSS is pulled low, the SPI enters the master mode fault state: the MSTR bit is automatically cleared and the device is configured in slave mode.

A basic example of interconnections between a single master and a single slave is illustrated in the following figure.

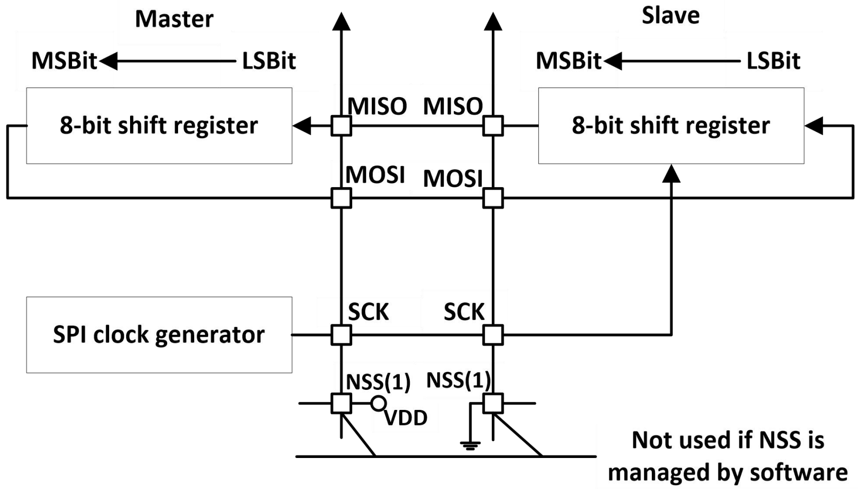


Figure 8.33 Single master/ single slave application

The MOSI pins are connected together and the MISO pins are connected together.In this way data is transferred serially between master and slave (most significant bit first).

The communication is always initiated by the master. When the master device transmits data to a slave device via the MOSI pin, the slave device responds via the MISO pin.This implies full-duplex communication with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

Slave select (NSS) pin management

Hardware or software slave select management can be set using the SSM bit in the SPI_CR1 register.

- Software NSS management (SSM = 1)
- The slave select information is driven internally by the value of the SSI bit in the SPI_CR1 register. The external NSS pin remains free for other application uses.
- Hardware NSS management (SSM = 0)
- Two configurations are possible depending on the NSS output configuration (SSOE bit in register SPI_CR2).
    - NSS output enabled (SSM = 0, SSOE = 1)

This configuration is used only when the device operates in master mode. The NSS signal is driven low when the master starts the communication and is kept low until the SPI is disabled.

- NSS output disabled (SSM = 0, SSOE = 0)

This configuration allows multimaster capability for devices operating in master mode. For devices set as slave, the NSS pin acts as a classical NSS input: the slave is selected when NSS is low and deselected when NSS high.

Clock phase and clock polarity

Four possible timing relationships may be chosen by software,using the CPOL and CPHA bits in the SPI_CR1 register. The CPOL (clock polarity) bit controls the steady state value of the clock when no data is being transferred. This bit affects both master and slave modes.If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA (clock phase) bit is set, the second edge on the SCK pin (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set) is the MSBit capture strobe. Data are latched on the occurrence of the second clock transition.If the CPHA bit is reset, the first edge on the SCK pin (falling edge if CPOL bit is set, rising edge if CPOL bit is reset) is the MSBit capture strobe. Data are latched on the occurrence of the first clock transition.

The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

The following figure shows an SPI transfer with the four combinations of the CPHA and CPOL bits.The diagram may be interpreted as a master or slave timing diagram where the SCK pin,the MISO pin, the MOSI pin are directly connected between the master and the slave device.

- Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.
- Master and slave must be programmed with the same timing mode.
- The idle state of SCK must correspond to the polarity selected in the SPI_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).
- The Data Frame Format (8- or 16-bit) is selected through the DFF bit in SPI_CR1 register,and determines the data length during transmission/reception.

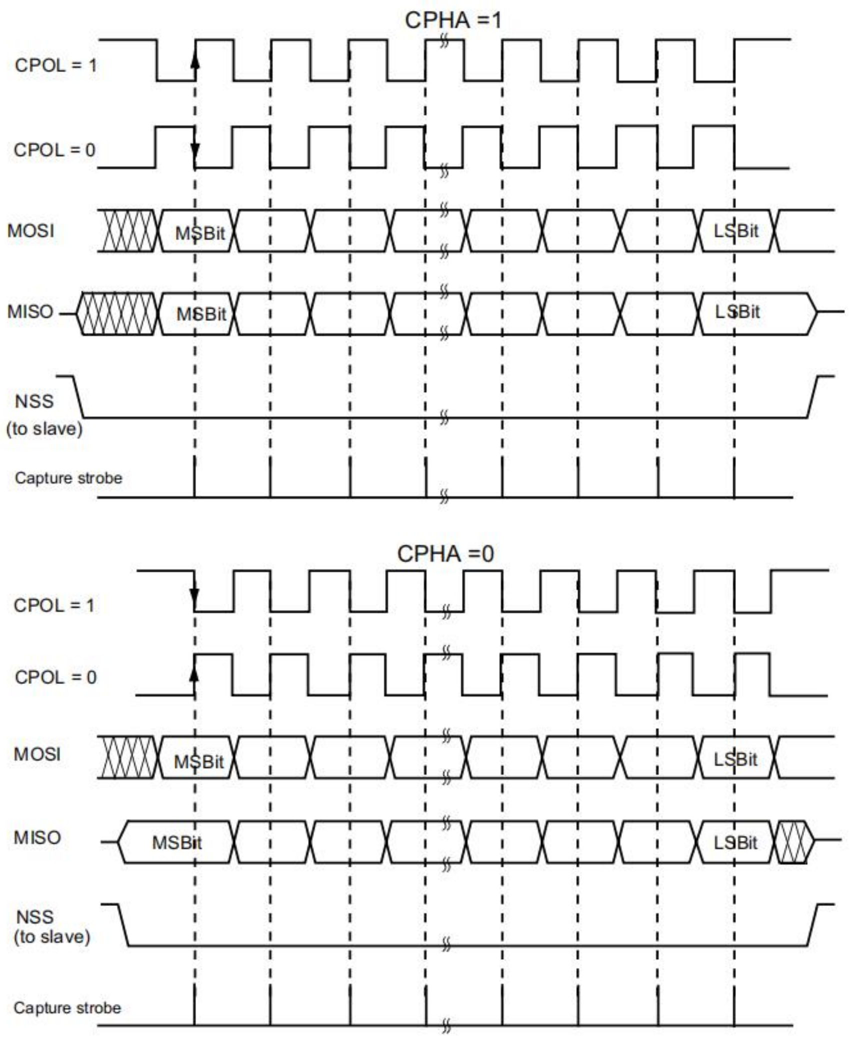


Figure 8.34 Data clock timing diagram

Note: These timings are shown with the LSBFIRST bit reset in the SPI_CR1 register.

Data frame format

Data can be shifted out either MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI_CR1 register.

Each data frame is 8 or 16 bits long depending on the size of the data programmed using the DFF bit in the SPI_CR1 register. The selected data frame format is applicable for transmission and/or reception.

8.9.3.3.Configuring the SPI in slave mode

In the slave configuration, the serial clock is received on the SCK pin from the master device. The value set in the BR[2:0] bits in the SPI_CR1 register, does not affect the data transfer rate.

Note: It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication. It is mandatory to have the polarity of the communication clock set to the steady state value before the slave and the master are enabled.

Follow the procedure below to configure the SPI in slave mode:

- Set the DFF bit to define 8- or 16-bit data frame format
- Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock.For correct data transfer, the CPOL and CPHA bits must be configured in the same way in the slave device and the master device.
- The frame format (MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI_CR1 register) must be the same as the master device.
- In Hardware mode, the NSS pin must be connected to a low level signal during the complete byte transmit sequence. In NSS software mode, set the SSM bit and clear the SSI bit in the SPI_CR1 register.
- Clear the MSTR bit and set the SPE bit (both in the SPI_CR1 register) to assign the pins to alternate functions.

In this configuration the MOSI pin is a data input and the MISO pin is a data output.

Transmit sequence

The data byte is parallel-loaded into the Tx buffer during a write cycle.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin. The remaining bits (the 7 bits in 8-bit data frame format, and the 15 bits in 16-bit data frame format) are loaded into the shift-register. The TXE flag in the SPI_SR register is set on the transfer of data from the Tx Buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set.

Receive sequence

For the receiver, when data transfer is complete:

- The Data in shift register is transferred to Rx Buffer and the RXNE flag (SPI_SR register) is set.
- An Interrupt is generated if the RXNEIE bit is set in the SPI_CR2 register.

After the last sampling clock edge the RXNE bit is set, a copy of the data byte received in the shift register is moved to the Rx buffer. When the SPI_DR register is read, the SPI peripheral returns this buffered value.

Clearing of the RXNE bit is performed by reading the SPI_DR register.

8.9.3.4.Configuring the SPI in master mode

In the master configuration, the serial clock is generated on the SCK pin.

Procedure

- Select the BR[2:0] bits to define the serial clock baud rate (see SPI_CR1 register).
- Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock,(This step is not required when the TI mode is selected).
- Set the DFF bit to define 8- or 16-bit data frame format.
- Configure the LSBFIRST bit in the SPI_CR1 register to define the frame format,(This step is not required when the TI mode is selected).
- If the NSS pin is required in input mode, in hardware mode, connect the NSS pin to a high-level signal during the complete byte transmit sequence. In NSS software mode, set the SSM and SSI bits in the SPI_CR1 register. If the NSS pin is required in output mode, the SSOE bit only should be set,(This step is not required when the TI mode is selected).
- Set the FRF bit in SPI_CR2 to select the TI protocol for serial communications.
- The MSTR and SPE bits must be set (they remain set only if the NSS pin is connected to a high-level signal).

In this configuration the MOSI pin is a data output and the MISO pin is a data input.

Transmit sequence

The transmit sequence begins when a byte is written in the Tx Buffer.

The data byte is parallel-loaded into the shift register (from the internal bus) during the first bit transmission and then shifted out serially to the MOSI pin MSB first or LSB first depending on the LSBFIRST bit in the SPI_CR1 register. The TXE flag is set on the transfer of data from the Tx Buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set.

Receive sequence

For the receiver, when data transfer is complete:

- The data in the shift register is transferred to the RX Buffer and the RXNE flag is set.
- An interrupt is generated if the RXNEIE bit is set in the SPI_CR2 register.

At the last sampling clock edge the RXNE bit is set, a copy of the data byte received in the shift register is moved to the Rx buffer. When the SPI_DR register is read, the SPI peripheral returns this buffered value.Clearing the RXNE bit is performed by reading the SPI_DR register.

A continuous transmit stream can be maintained if the next data to be transmitted is put in the Tx buffer once the transmission is started. Note that TXE flag should be ‘1 before any attempt to write the Tx buffer is made.

Note: When a master is communicating with SPI slaves which need to be de-selected between transmissions, the NSS pin must be configured as GPIO or another GPIO must be used and toggled by software.

8.9.3.5.Configuring the SPI for half-duplex communication

The SPI is capable of operating in half-duplex mode in 2 configurations.

- 1 clock and 1 bidirectional data wire
- 1 clock and 1 data wire (receive-only or transmit-only)

1 clock and 1 bidirectional data wire (BIDIMODE = 1)

This mode is enabled by setting the BIDIMODE bit in the SPI_CR1 register. In this mode SCK is used for the clock and MOSI in master or MISO in slave mode is used for data communication. The transfer direction (Input/Output) is selected by the BIDIOE bit in the SPI_CR1 register. When this bit is 1, the data line is output otherwise it is input.

1 clock and 1 unidirectional data wire (BIDIMODE = 0)

In this mode, the application can use the SPI either in transmit-only mode or in receive-only mode.

- Transmit-only mode is similar to full-duplex mode (BIDIMODE=0, RXONLY=0): the data are transmitted on the transmit pin (MOSI in master mode or MISO in slave mode) and the receive pin (MISO in master mode or MOSI in slave mode) can be used as a general-purpose IO. In this case, the application just needs to ignore the Rx buffer (if the data register is read, it does not contain the received value).
- In receive-only mode, the application can disable the SPI output function by setting the RXONLY bit in the SPI_CR1 register. In this case, it frees the transmit IO pin (MOSI in master mode or MISO in slave mode), so it can be used for other purposes.

To start the communication in receive-only mode, configure and enable the SPI:

- In master mode, the communication starts immediately and stops when the SPE bit is cleared and the current reception stops. There is no need to read the BSY flag in this mode. It is always set when an SPI communication is ongoing.
- In slave mode, the SPI continues to receive as long as the NSS is pulled down (or the SSI bit is cleared in NSS software mode) and the SCK is running.

8.9.3.7.Data transmission and reception procedures

Rx and Tx buffers

In reception, data are received and then stored into an internal Rx buffer while In transmission, data are first stored into an internal Tx buffer before being transmitted.

A read access of the SPI_DR register returns the Rx buffered value whereas a write access to the SPI_DR stores the written data into the Tx buffer.

Start sequence in master mode

- In full-duplex (BIDIMODE=0 and RXONLY=0)
    - The sequence begins when data are written into the SPI_DR register (Tx buffer).
    - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MOSI pin.
    - At the same time, the received data on the MISO pin is shifted in serially to the 8- bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
- In unidirectional receive-only mode (BIDIMODE=0 and RXONLY=1)
    - The sequence begins as soon as SPE=1
    - Only the receiver is activated and the received data on the MISO pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
- In bidirectional mode, when transmitting (BIDIMODE=1 and BIDIOE=1)
    - The sequence begins when data are written into the SPI_DR register (Tx buffer).
    - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MOSI pin.
    - No data are received.
- In bidirectional mode, when receiving (BIDIMODE=1 and BIDIOE=0)
    - The sequence begins as soon as SPE=1 and BIDIOE=0.
    - The received data on the MOSI pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
    - The transmitter is not activated and no data are shifted out serially to the MOSI pin.

Start sequence in slave mode

- In full-duplex mode (BIDIMODE=0 and RXONLY=0)
    - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MOSI pin. The 7 remaining bits are loaded into the shift register.
    - At the same time, the data are parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission, and then shifted out serially to the MISO pin. The software must have written the data to be sent before the SPI master device initiates the transfer.
- In unidirectional receive-only mode (BIDIMODE=0 and RXONLY=1)
    - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MOSI pin. The 7 remaining bits are loaded into the shift register.
    - The transmitter is not activated and no data are shifted out serially to the MISO pin.
- In bidirectional mode, when receiving (BIDIMODE=1 and BIDIOE=0)

  - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MISO pin.
  - The received data on the MISO pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
  - The transmitter is not activated and no data are shifted out serially to the MISO pin.
- In bidirectional mode, when transmitting (BIDIMODE=1 and BIDIOE=1)
  - The sequence begins when the slave device receives the clock signal and the first bit in the Tx buffer is transmitted on the MISO pin.
  - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MISO pin. The software must have written the data to be sent before the SPI master device initiates the transfer.
  - No data are received.

Handling data transmission and reception

The TXE flag (Tx buffer empty) is set when the data are transferred from the Tx buffer to the shift register. It indicates that the internal Tx buffer is ready to be loaded with the next data.

An interrupt can be generated if the TXEIE bit in the SPI_CR2 register is set. Clearing the TXE bit is performed by writing to the SPI_DR register.

Note: The software must ensure that the TXE flag is set to 1 before attempting to write to the Tx buffer. Otherwise, it overwrites the data previously written to the Tx buffer.

The RXNE flag (Rx buffer not empty) is set on the last sampling clock edge, when the data are transferred from the shift register to the Rx buffer. It indicates that data are ready to be read from the SPI_DR register. An interrupt can be generated if the RXNEIE bit in the SPI_CR2 register is set. Clearing the RXNE bit is performed by reading the SPI_DR register.

For some configurations, the BSY flag can be used during the last data transfer to wait until the completion of the transfer.

Full-duplex transmit and receive procedure in master or slave mode (BIDIMODE=0 and RXONLY=0)

The software has to follow this procedure to transmit and receive data:

- Enable the SPI by setting the SPE bit to 1.
- Write the first data item to be transmitted into the SPI_DR register (this clears the TXE flag).
- Wait until TXE=1 and write the second data item to be transmitted. Then wait until RXNE=1 and read the SPI_DR to get the first received data item (this clears the RXNE bit). Repeat this operation for each data item to be transmitted/received until the n–1 received data.
- Wait until RXNE=1 and read the last received data.
- Wait until TXE=1 and then wait until BSY=0 before disabling the SPI.This procedure can also be implemented using dedicated interrupt subroutines launched at each rising edges of the RXNE or TXE flag.

Transmit-only procedure (BIDIMODE=0 RXONLY=0)

In this mode, the procedure can be reduced as described below and the BSY bit can be used to wait until the completion of the transmission.

- Enable the SPI by setting the SPE bit to 1.
- Write the first data item to send into the SPI_DR register (this clears the TXE bit).
- Wait until TXE=1 and write the next data item to be transmitted. Repeat this step for each data item to be transmitted.
- After writing the last data item into the SPI_DR register, wait until TXE=1, then wait until BSY=0, this indicates that the transmission of the last data is complete.

This procedure can be also implemented using dedicated interrupt subroutines launched at each rising edge of the TXE flag.

Note: First,During discontinuous communications, there is a 2 APB clock period delay between the write operation to SPI_DR and the BSY bit setting. As a consequence, in transmit-only mode, it is mandatory to wait first until TXE is set and then until BSY is cleared after writing the last data.Second,After transmitting two data items in transmit-only mode, the OVR flag is set in the SPI_SR register since the received data are never read.

Bidirectional transmit procedure (BIDIMODE=1 and BIDIOE=1)

In this mode, the procedure is similar to the procedure in Transmit-only mode except that the BIDIMODE and BIDIOE bits both have to be set in the SPI_CR2 register before enabling the SPI.

Unidirectional receive-only procedure (BIDIMODE=0 and RXONLY=1)

In this mode, the procedure can be reduced as described:

- Set the RXONLY bit in the SPI_CR2 register.
- Enable the SPI by setting the SPE bit to 1:
  - In master mode, this immediately activates the generation of the SCK clock, and data are serially received until the SPI is disabled (SPE=0).
  - In slave mode, data are received when the SPI master device drives NSS low and generates the SCK clock.
- Wait until RXNE=1 and read the SPI_DR register to get the received data (this clears the RXNE bit). Repeat this operation for each data item to be received.

This procedure can also be implemented using dedicated interrupt subroutines launched at each rising edge of the RXNE flag.

Bidirectional receive procedure (BIDIMODE=1 and BIDIOE=0)

In this mode, the procedure is similar to the Receive-only mode procedure except that the BIDIMODE bit has to be set and the BIDIOE bit cleared in the SPI_CR2 register before enabling the SPI.

Continuous and discontinuous transfers

When transmitting data in master mode, if the software is fast enough to detect each rising edge of TXE (or TXE interrupt) and to immediately write to the SPI_DR register before the ongoing data transfer is complete, the communication is said to be continuous. In this case, there is no discontinuity in the generation of the SPI clock between each data item and the BSY bit is never cleared between each data transfer.

On the contrary, if the software is not fast enough, this can lead to some discontinuities in the communication. In this case, the BSY bit is cleared between each data transmission.

In Master receive-only mode (RXONLY=1), the communication is always continuous and the BSY flag is always read at 1.

In slave mode, the continuity of the communication is decided by the SPI master device.In any case,even if the communication is continuous,the BSY flag goes low between each transfer for a minimum duration of one SPI clock cycle.

8.9.3.8.CRC calculation

A CRC calculator has been implemented for communication reliability. Separate CRC calculators are implemented for transmitted data and received data. The CRC is calculated using a programmable polynomial serially on each bit. It is calculated on the sampling clock edge defined by the CPHA and CPOL bits in the SPI_CR1 register.

Note: This SPI offers two kinds of CRC calculation standard which depend directly on the data frame format selected for the transmission and/or reception: 8-bit data (CR8) and 16-bit data (CRC16).

CRC calculation is enabled by setting the CRCEN bit in the SPI_CR1 register. This action resets the CRC registers (SPI_RXCRCR and SPI_TXCRCR). In full duplex or transmitter only mode, when the transfers are managed by the software (CPU mode), it is necessary to write the bit CRCNEXT immediately after the last data to be transferred is written to the SPI_DR.At the end of this last data transfer, the SPI_TXCRCR value is transmitted.

In receive only mode and when the transfers are managed by software (CPU mode), it is necessary to write the CRCNEXT bit after the second last data has been received. The CRC is received just after the last data reception and the CRC check is then performed.

At the end of data and CRC transfers, the CRCERR flag in the SPI_SR register is set if corruption occurs during the transfer.

If data are present in the TX buffer, the CRC value is transmitted only after the transmission of the data byte. During CRC transmission, the CRC calculator is switched off and the register value remains unchanged.

SPI communication using the CRC is possible through the following procedure:

- Program the CPOL, CPHA, LSBFirst, BR, SSM, SSI and MSTR values.
- Program the polynomial in the SPI_CRCPR register.
- Enable the CRC calculation by setting the CRCEN bit in the SPI_CR1 register.This also clears the SPI_RXCRCR and SPI_TXCRCR registers.
- Enable the SPI by setting the SPE bit in the SPI_CR1 register.
- Start the communication and sustain the communication until all but one byte or half word have been transmitted or received.
    - In full duplex or transmitter-only mode, when the transfers are managed by software, when writing the last byte or half word to the Tx buffer, set the CRCNEXT bit in the SPI_CR1 register to indicate that the CRC will be transmitted after the transmission of the last byte.
    - In receiver only mode, set the bit CRCNEXT just after the reception of the second to last data to prepare the SPI to enter in CRC Phase at the end of the reception of the last data. CRC calculation is frozen during the CRC

transfer.

- After the transfer of the last byte or half word, the SPI enters the CRC transfer and check phase. In full duplex mode or receiver-only mode, the received CRC is compared to the SPI_RXCRCR value. If the value does not match, the CRCERR flag in SPI_SR is set and an interrupt can be generated when the ERRIE bit in the SPI_CR2 register is set.

Note: When the SPI is in slave mode, be careful to enable CRC calculation only when the clock is stable, that is, when the clock is in the steady state. If not, a wrong CRC calculation may be done. In fact, the CRC is sensitive to the SCK slave input clock as soon as CRCEN is set, and this, whatever the value of the SPE bit.

With high bitrate frequencies, be careful when transmitting the CRC. As the number of used CPU cycles has to be as low as possible in the CRC transfer phase, it is forbidden to call software functions in the CRC transmission sequence to avoid errors in the last data and CRC reception. In fact, CRCNEXT bit has to be written before the end of the transmission/reception of the last data.

For high bit rate frequencies, it is advised to use the DMA mode to avoid the degradation of the SPI speed performance due to CPU accesses impacting the SPI bandwidth.

When the devices are configured as slaves and the NSS hardware mode is used, the NSS pin needs to be kept low between the data phase and the CRC phase.

When the SPI is configured in slave mode with the CRC feature enabled, CRC calculation takes place even if a high level is applied on the NSS pin. This may happen for example in case of a multislave environment where the communication master addresses slaves alternately.

Between a slave deselection (high level on NSS) and a new slave selection (low level on NSS), the CRC value should be cleared on both master and slave sides in order to resynchronize the master and slave for their respective CRC calculation.

To clear the CRC, follow the procedure below:

- Disable SPI (SPE = 0).
- Clear the CRCEN bit.
- Set the CRCEN bit.
- Enable the SPI (SPE = 1).

8.9.3.9.Status flags

Four status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

When it is set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can be loaded into the buffer. The TXE flag is cleared when writing to the SPI_DR register.

Rx buffer not empty (RXNE)

When set, this flag indicates that there are valid received data in the Rx buffer. It is cleared when SPI_DR is read.

BUSY flag

This BSY flag is set and cleared by hardware (writing to this flag has no effect). The BSY flag indicates the state of the communication layer of the SPI.

When BSY is set, it indicates that the SPI is busy communicating. There is one exception in master mode / bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0) where the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software wants to disable the SPI and enter Halt mode (or disable the peripheral clock). This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is also useful to avoid write collisions in a multimaster system.

The BSY flag is set when a transfer starts, with the exception of master mode / bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0).

It is cleared:

- when a transfer is finished (except in master mode if the communication is continuous).
- When the SPI is disabled.
- When a master mode fault occurs (MODF=1).
- When communication is not continuous, the BSY flag is low between each communication.
- When communication is continuous:
    - in master mode, the BSY flag is kept high during all the transfers.
    - in slave mode, the BSY flag goes low for one SPI clock cycle between each transfer.

Note: Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.

8.9.3.10.Disabling the SPI

When a transfer is terminated, the application can stop the communication by disabling the SPI peripheral. This is done by clearing the SPE bit.

For some configurations, disabling the SPI and entering the Halt mode while a transfer is ongoing can cause the current transfer to be corrupted and/or the BSY flag might become unreliable.

To avoid any of those effects, it is recommended to respect the following procedure when disabling the SPI:

In master or slave full-duplex mode (BIDIMODE=0, RXONLY=0)

- Wait until RXNE=1 to receive the last data
- Wait until TXE=1
- Then wait until BSY=0
- Disable the SPI (SPE=0) and, eventually, enter the Halt mode (or disable the peripheral clock)

In master or slave unidirectional transmit-only mode (BIDIMODE=0, RXONLY=0) or bidirectional transmit mode (BIDIMODE=1, BIDIOE=1)

After the last data is written into the SPI_DR register:

- Wait until TXE=1
- Then wait until BSY=0
- Disable the SPI (SPE=0) and, eventually, enter the Halt mode (or disable the peripheral clock)

In master unidirectional receive-only mode (MSTR=1, BIDIMODE=0, RXONLY=1) or bidirectional receive mode (MSTR=1, BIDIMODE=1, BIDIOE=0)

This case must be managed in a particular way to ensure that the SPI does not initiate a new transfer:

- Wait for the second to last occurrence of RXNE=1 (n–1)
- Then wait for one SPI clock cycle (using a software loop) before disabling the SPI (SPE=0)
- Then wait for the last RXNE=1 before entering the Halt mode (or disabling the peripheral clock)

Note: In master bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0), the BSY flag is kept low during transfers.

In slave receive-only mode (MSTR=0, BIDIMODE=0, RXONLY=1) or bidirectional receive mode (MSTR=0, BIDIMODE=1, BIDOE=0)

- You can disable the SPI (write SPE=1) at any time: the current transfer will complete before the SPI is effectively disabled.
- Then, if you want to enter the Halt mode, you must first wait until BSY = 0 before entering the Halt mode (or disabling the peripheral clock).

8.9.3.11.SPI communication using DMA

To operate at its maximum speed, the SPI needs to be fed with the data for transmission and the data received on the Rx buffer should be read to avoid overrun. To facilitate the transfers, the SPI features a DMA capability implementing a simple request/acknowledge protocol.

A DMA access is requested when the enable bit in the SPI_CR2 register is enabled. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPI_DR register (this clears the TXE flag).
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPI_DR register (this clears the RXNE flag).

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received are not read.

When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (flag TCIF is set in the DMA_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering the Stop mode. The software must first wait until TXE=1 and then until BSY=0.

Note: During discontinuous communications, there is a 2 APB clock period delay between the write operation to SPI_DR and the BSY bit setting. As a consequence, it is mandatory to wait first until TXE=1 and then until BSY=0 after writing the last data.

DMA capability with CRC

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication are automatic that is without using the bit CRCNEXT. After the CRC reception, the CRC must be read in the SPI_DR register to clear the RXNE flag.

At the end of data and CRC transfers, the CRCERR flag in SPI_SR is set if corruption occurs during the transfer.

8.9.4 SPIx Register Map

Offset	Name	Description
0x0000	SPI_CTRL	SPI Control Register
0x0004	SPI_WDATA	Data transmit to the SPI port
0x0008	SPI_RDATA	Data receive from the SPI port
0x000c	SPI_STAT	SPI status registers
0x0010	DMACR	DMA control registers
0x0014	DMATDLR	DMA tx request level registers
0x0018	DMARDLR	DMA rx request level registers
0x001c	CSNCTRL	CSN control register

SPI_CTRL address offset: 0x0000

Bit	R/W	Reset	Name	Description
31	R/W	0x0	tx_fifo_enable	Data write to Write Data Register go to 128 byte Transmitter FIFO 1: enable 0: disable
30	R/W	0x0	rx_fifo_enable	Data read from Read Data register come from 128 byte Receiver FIFO 1: enable 0: disable
29	R/W	0x0	tx_clr_fifo	Reset Transmitter FIFO pointers and Byte counters and Overrun status
28	R/W	0x0	rx_clr_fifo	Reset Receiver FIFO pointers and Byte counters and Overrun status
27:26	R/W	0x0	rx_trigger_level	Receiver FIFO Trigger Level: set the trigger level of interrupt 00=8byte 01=32byte 10=64byte 11=96byte

25	R/W	0x1	inactive_do_en	1=SPI_DO pin is high-Z while byte is not being transferred 0=SPI_DO pin is driven while byte is not being transferred
24	R/W	0x0	active_do_en	1=SPI_DO pin is high-Z while byte is being transferred 0=SPI_DO pin is driven while byte is being transferred
23	R/W	0x0	bidirect_data_h	1=Data is written and read on SPI_DO pin 0=Data is written on SPI_DO pin, and read on SPI_DI pin
22	R/W	0x0	use_rdy_out_h	1=Master/Slave use SPI_RDY pin as bidirect Ready line 0=Master/Slave use SPI_RDY pin as SPI chip enable
21	R/W	0x0	invert_clock_h	1=Clock is inverted(low when IDEL) 0=Clock is not inverted(high when IDLE)
20	R/W	0x0	msb_first_h	1=MSB is sent/received first 0=LSB is sent/received first
19	R/W	0x0	soft_reset_h	1=Soft reset SPI hardware, except setup registers 0=Allow SPI to run
18	R/W	0x0	master_ce_at_end	Level of SPI_RDY(CE) pin after a transfer in master mode
17	R/W	0x0	mode1_h	1= mode 1(use second clock edge) 0= mode 0 (use first clock edge)
16	R/W	0x0	master_enable_h	1: SPI in master mode 0: SPI in slave mode
15:0	R/W	0x0	clk_divider	For Master mode only. Set high and low time of SPI_CLK to (Clk_Divider+1)

SPI_WDATA address offset: 0x0004

Bit	R/W	Reset	Name	Description
31:8	R/W	N/A	N/A	reserved
7:0	R/W	0x0	write_data	data to be written out the SPI port

SPI_RDATA address offset: 0x0008

Bit	R/W	Reset	Name	Description
31:8	R	0x0	N/A	reserved
7:0	R	0x0	read_data	data read from the SPI port

SPI_STAT address offset: 0x000c

Bit	R/W	Reset	Name	Description
31	R/W	0x0	spi_int	When read,it reflects the SPI interrupt status.When write 1,it dis-asserts the SPI interrupt
30:28	R	0x0	bit_count	Bit count of current byte being written/read
27	R	0x0	wait_for_rdy_h	1=SPI is waiting for SPI_RDY line to go high
26	R	0x0	spi_rdy_out	Level being driven on SPI_RDY pin
25	R	0x0	spi_rdy_in	Level being driven on SPI_RDY pin
24	R	0x0	spi_active_h	1=SPI transfer is in process
23:16	R/W	0x0	tx_byte_cnt	TX FIFO byte count
15:8	R/W	0x0	rx_byte_cnt	RX FIFO byte count
7	R/W	0x0	rx_fifo_trig	RX FIFO trigger level reached
6	R/W	0x0	rx_trig_int_en	This bit enable RX FIFO trigger level interrupt
5	R/W	0x0	tx_empty	TX FIFO empty flag
4	R/W	0x0	tx_empty_int_en	This bit enable Transmitter FIFO empty interrupt 1: enable 0: disable
3	R/W	0x0	N/A	reserved
2	R/W	0x0	tx_fifo_overrun	Overrun Error. Set When transmitter FIFO is full and shift register contains next character.
1	R/W	0x0	rx_fifo_overrun	RX overrun error flag Overrun Error. Set When receiver FIFO is full and shift register contains next character.
0	R/W	0x0	rx_notempty_h	This bit set to one whenever a complete incoming byte has been received. This bit reset to zero by reading all of the data in the Receiver FIFO 1: no empty 0: empty

SPI_ DMACR address offset: 0x0010

Bit	R/W	Reset	Name	Description
31:2	NA	0x0	N/A	reserved
1	R/W	0x0	TDMAE	transmit DMA Enable, this bit enables/disables

				the transmit FIFO DMA channel. This bit-field enable the dma tx transfer, while set 1 and the value of **tx_full** is 0, the **dma_tx_single** which indicates the status of SPI DMA TX transfer will be set 1. And **dma_tx_single** will be reset when this bit's value is 0,or any value of **dma_tx_ack** and **tx_full** is 1. 0: Transmit DMA disable 1: Transmit DMA enable
0	R/W	0x0	RDMAE	Receive DMA enable. This bit enables/disables the transmit FIFO DMA channel. This bit-field enable the dma tx transfer, while set 1 and the value of **rx_empty** is 0, the **dma_rx_single** which indicates the status of SPI DMA RX transfer will be set 1. And **dma_rx_single** will be reset when this bit's value is 0,or any value of **dma_rx_ack** and **rx_empty** is 1. 0: Receive DMA disable 1: Receive DMA enable

SPI_DMATDLR address offset: 0x0014

Bit	**R/W**	**Reset**	**Name**	**Description**
SPI_TX_ABW-1:0	RW	0x0	DMATDLR	This register is only valid when the SPI is configured with a set of DMA interface signals. When SPI is not configured for DMA operation , this register will not exist and writing to its address will have no effect; reading from its address will return zero. This bit field controls the level at which a DMA request is made by the transmit logic. while **txflr <= dmatdlr,dma_tx_req** while be set 1.

SPI_DMARDLR address offset: 0x0018

Bit	**R/W**	**Reset**	**Name**	**Description**
SPI_RX_ABW-1:0	RW	0x0	DMARDLR	This register is only valid when the SPI is configured with a set of DMA interface signals. When SPI is not configured for DMA operation , this register will not exist and writing to its address will have no effect; reading from its address will

				return zero. while **rxflr >= dmardlr + 1,dma_rx_req** while be set 1.

SPIx_CSNCTRL address offset: 0x001c

Bit	R/W	Reset	Name	Description
31:2	N/A	0x0	N/A	reserved
1	RW	0x0	cs_gpo	When the value of cs_mode is 1,the value of **o_spi_mst_csn** is equal to cs_gpo
0	RW	0x0	cs_mode	0:**o_spi_mst_csn** will be valued by device 1: **o_spi_mst_csn** will be valued by cs_gpo

8.10 I2C

8.10.1 Introduction

The I2C bus is a two-wire serial interface, consisting of a serial data line (SDA) and a serial clock (SCL).

These wires carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a “transmitter” or “receiver,” depending on the function of the device. Devices can also be considered as masters or slaves when performing data transfers. A master is a device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

8.10.2 Main Features

- Two-wire I2C serial interface – consists of a serial data line (SDA) and a serial clock (SCL)
- Three speeds:
  - Standard mode (100 Kb/s)
  - Fast mode (400 Kb/s)
  - High-speed mode (3.4 Mb/s)
- Clock synchronization
- Master OR slave I2C operation
- 7-bit or 10-bit addressing
- 7-bit or 10-bit combined format transfers
- Bulk transmit mode
- Ignores CBUS addresses (an older ancestor of I2C that used to share the I2C bus)
- Transmit and receive buffers
- Interrupt or polled-mode operation
- Handles Bit and Byte waiting at all bus speeds
- Simple software interface consistent with Design Ware APB peripherals
- Component parameters for configurable software driver support
- DMA handshaking interface compatible with the Onmicro_ahb_dmac handshaking interface
- Add a bullet point here for programmable SDA hold time (tHD;DAT)

The Onmicro_i2c requires external hardware components as support in order to be compliant in an I2C system. The descriptions are detailed later in this document.

It must also be noted that the Onmicro_i2c should only be operated either as (but not both):

- A master in an I2C system and programmed only as a Master; OR

- A slave in an I2C system and programmed only as a Slave.

8.10.3 Function Description

8.10.3.1.I2C Bus Terms

The following terms relate to how the role of the I2C device and how it interacts with other I2C devices on the bus.

- Transmitter – the device that sends data to the bus. A transmitter can either be a device that initiates the data transmission to the bus (a master-transmitter) or responds to a request from the master to send data to the bus (a slave-transmitter).
- Receiver – the device that receives data from the bus. A receiver can either be a device that receives data on its own request (a master-receiver) or in response to a request from the master (a slave-receiver).
- Master -– the component that initializes a transfer (START command), generates the clock (SCL) signal and terminates the transfer (STOP command). A master can be either a transmitter or a receiver.
- Slave – the device addressed by the master. A slave can be either receiver or transmitter.These concepts are illustrated in the following figure.

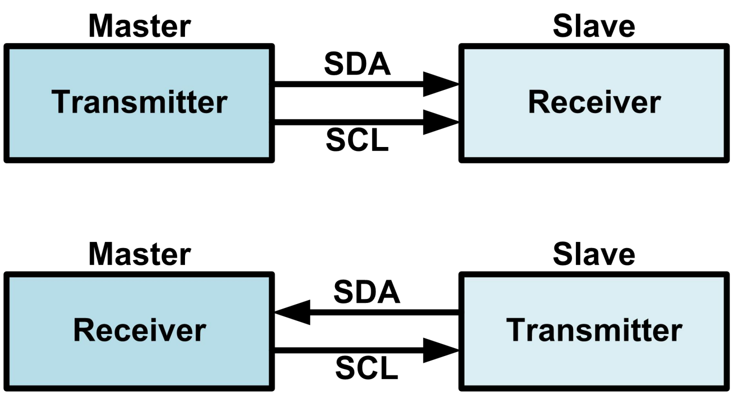


Figure 8.35 Master/Slave and Transmitter/Receiver Relationships

- Multi-master – the ability for more than one master to co-exist on the bus at the same time without collision or data loss.
- Arbitration – the predefined procedure that authorizes only one master at a time to take control of the bus.
- Synchronization – the predefined procedure that synchronizes the clock signals provided by two or more masters.
- SDA – data signal line (Serial DAta)
- SCL – clock signal line (Serial CLock)

8.10.3.2.Bus Transfer Terms

The following terms are specific to data transfers that occur to/from the I2C bus.

- START (RESTART) – data transfer begins with a START or RESTART condition. The level of the SDA data line changes from high to low, while the SCL clock line remains high. When this occurs, the bus becomes busy.

Note:START and RESTART conditions are functionally identical.

- STOP – data transfer is terminated by a STOP condition. This occurs when the level on the SDA data line passes from the low state to the high state, while the SCL clock line remains high. When the data transfer has been terminated, the bus is free or idle once again. The bus stays busy if a RESTART is generated instead of a STOP condition.

8.10.3.3.I2C Behavior

The Onmicro_i2c can be controlled via software to be either:

- An I2C master only, communicating with other I2C slaves; OR
- An I2C slave only, communicating with one more I2C masters.

The master is responsible for generating the clock and controlling the transfer of data. The slave is responsible for either transmitting or receiving data to/from the master. The acknowledgement of data is sent by the device that is receiving data, which can be either a master or a slave. As mentioned previously, the I2C protocol also allows multiple masters to reside on the I2C bus and uses an arbitration procedure to determine bus ownership.

Each slave has a unique address that is determined by the system designer. When a master wants to communicate with a slave, the master transmits a START/RESTART condition that is then followed by the slave’s address and a control bit (R/W) to determine if the master wants to transmit data or receive data from the slave. The slave then sends an acknowledge (ACK) pulse after the address.

If the master (master-transmitter) is writing to the slave (slave-receiver), the receiver gets one byte of data.

This transaction continues until the master terminates the transmission with a STOP condition. If the master is reading from a slave (master-receiver), the slave transmits (slave-transmitter) a byte of data to the master, and the master then acknowledges the transaction with the ACK pulse. This transaction continues until the master terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a RESTART condition. This behavior is illustrated in the following figure.

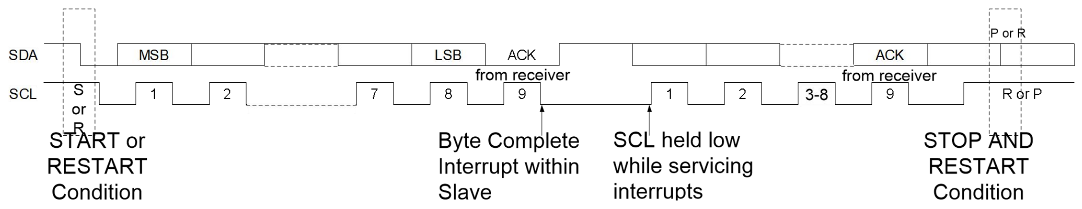

SDA
SCL
MSB
LSB
ACK
from receiver
ACK
from receiver
P or R
S or R
1
2
7
8
9
1
2
3-8
9
R or P
START or RESTART Condition
Byte Complete Interrupt within Slave
SCL held low while servicing interrupts
STOP AND RESTART Condition

Figure 8.36 Data transfer on the I2C Bus

The Onmicro_i2c is a synchronous serial interface.The SDA line is a bidirectional signal and changes only while the SCL line is low, except for STOP,START,and RESTART conditions.The output drivers are open-drain or open-collector to perform wire-AND functions on the bus.The maximum number of devices on the bus is limited by only the maximum capacitance specification of 400 pF.Data is transmitted in byte packages.

Note: Putting data into the FIFO generates a START,and emptying the FIFO generates a STOP.

START and STOP Generation

When operating as an I2C master, putting data into the transmit FIFO causes the Onmicro_i2c to generate a START condition on the I2C bus.Allowing the transmit FIFO to empty causes the Onmicro_i2c to generate a STOP condition on the I2C bus. When operating as a slave, the Onmicro_i2c does not generate START and STOP conditions, as per the protocol.However,if a read request is made to the Onmicro_i2c, it holds the SCL line low until read data has been supplied to it. This stalls the I2C bus until read data is provided to the slave Onmicro_i2c,or the Onmicro_i2c slave is disabled by writing a 0 to IC_ENABLE.

Combined Formats

The Onmicro_i2c supports mixed read and write combined format transactions in both 7-bit and 10-bit addressing modes.

The Onmicro_i2c does not support mixed address and mixed address format—that is, a 7-bit address transaction followed by a 10-bit address transaction or vice versa—combined format transactions.

To initiate combined format transfers, IC_CON.IC_RESTART_EN should be set to 1. With this value set and operating as a master, when the Onmicro_i2c completes an I2C transfer, it checks the transmit FIFO and executes the next transfer. If the direction of this transfer differs from the previous transfer, the combined format is used to issue the transfer. If the transmit FIFO is empty when the current I2C transfer completes, a STOP is issued and the next transfer is issued following a START condition.

8.10.3.4.I2C Protocols

START and STOP Conditions

When the bus is idle, both the SCL and SDA signals are pulled high through external pull-up resistors on the bus. When the master wants to start a transmission on the bus, the master issues a START condition.

This is defined to be a high-to-low transition of the SDA signal while SCL is 1.When the master wants to terminate the transmission, the master issues a STOP condition.This is defined to be a low-to-high transition of the SDA line while SCL is 1.The following figure shows the timing of the START and STOP conditions.When data is being transmitted on the bus, the SDA line must be stable when SCL is 1.

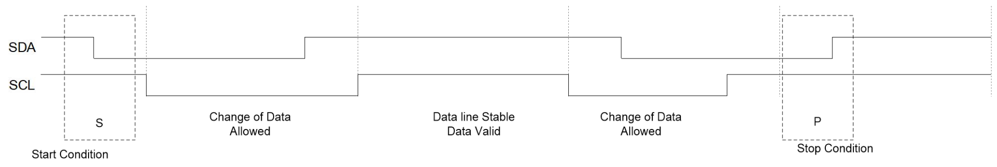


Figure 8.37 START and STOP Condition

Note: The signal transitions for the START/STOP conditions,as depicted in Figure 8.37,reflect those observed at the output signals of the Master driving the I2C bus.Care should be taken when observing the SDA/SCL signals at the input signals of the Slave(s), because unequal line delays may result in an incorrect SDA/SCL timing relationship.

Addressing Slave Protocol

There are two address formats: the 7-bit address format and the 10-bit address format.

7-bit Address Format

During the 7-bit address format, the first seven bits (bits 7:1) of the first byte set the slave address and the LSB bit (bit 0) is the R/W bit as shown in the following figure. When bit 0 (R/W) is set to 0, the master writes to the slave. When bit 0 (R/W) is set to 1, the master reads from the slave.

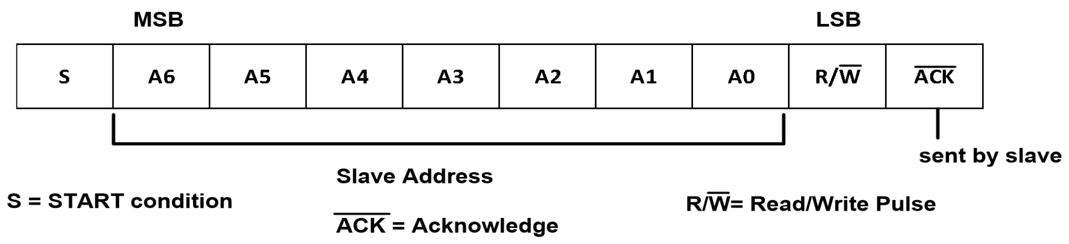


Figure 8.38 7-bit Address Format

10-bit Address Format

During 10-bit addressing,two bytes are transferred to set the 10-bit address.The transfer of the first byte contains the following bit definition. The first five bits (bits 7:3) notify the slaves that this is a 10-bit transfer followed by the next two bits (bits 2:1), which set the slaves address bits 9:8, and the LSB bit (bit 0) is the R/W bit. The second byte transferred sets bits 7:0 of the slave address. The following figure shows the 10-bit address format, and the following table defines the special purpose and reserved first byte addresses.

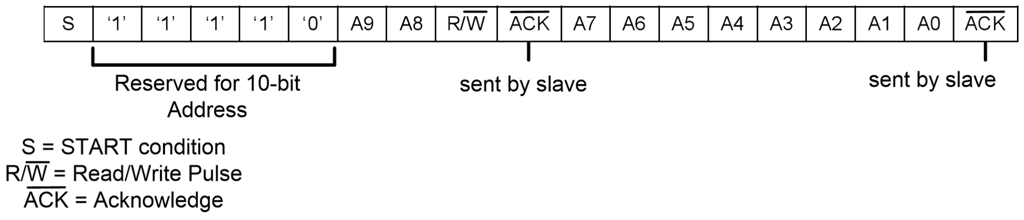


Figure 8.39 10-bit Address Format

Slave Address	R/W Bit	Description
0000 000	0	General Call Address. Onmicro_i2c places the data in the receive buffer and issues a General Call interrupt.
0000 000	1	START byte.
0000 001	X	CBUS address. Onmicro_i2c ignores these accesses.
0000 010	X	Reserved.
0000 011	X	Reserved.
0000 1XX	X	High-speed master code.
1111 1XX	X	Reserved.

1111 0XX	X	10-bit slave addressing.

Table 8.6 I2C Definition of Bits in First Byte

Onmicro_i2c does not restrict you from using these reserved addresses. However, if you use these reserved addresses, you may run into incompatibilities with other I2 C components.

Transmitting and Receiving Protocol

The master can initiate data transmission and reception to/from the bus, acting as either a master-transmitter or master-receiver. A slave responds to requests from the master to either transmit data or receive data to/from the bus, acting as either a slave-transmitter or slave-receiver, respectively.

Master-Transmitter and Slave-Receiver

All data is transmitted in byte format, with no limit on the number of bytes transferred per data transfer.

After the master sends the address and R/W bit or the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge signal (ACK). When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a STOP condition. The slave must leave the SDA line high so that the master can abort the transfer.

If the master-transmitter is transmitting data as shown in the following figure, then the slave-receiver responds to the master-transmitter with an acknowledge pulse after every byte of data is received.

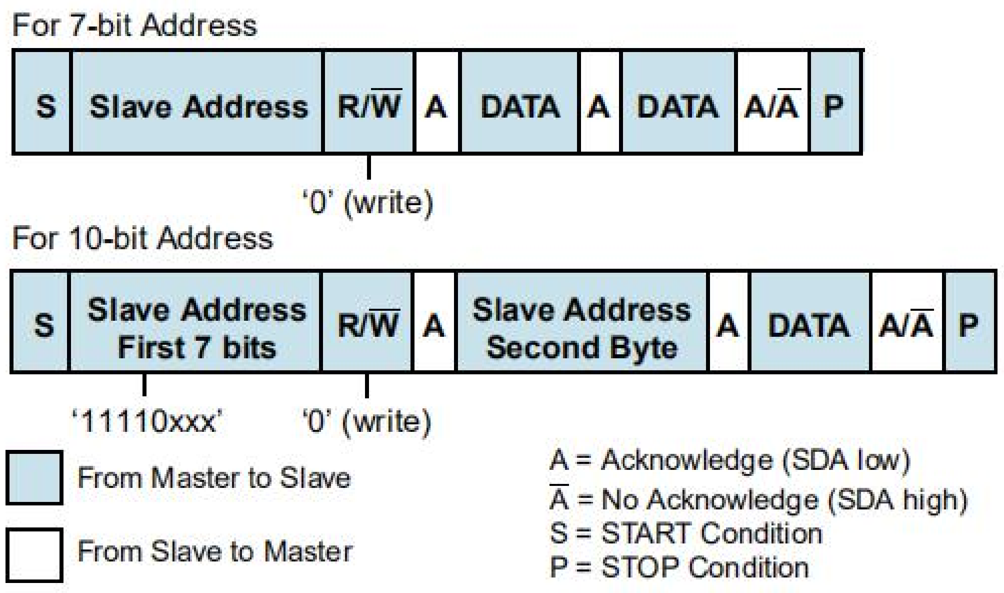


Figure 8.40 Master-Transmitter Protocol

Master-Receiver and Slave-Transmitter

If the master is receiving data as shown in the following figure, then the master responds to the slave-transmitter with an acknowledge pulse after a byte of data has been received, except for the last byte. This is the way the master-receiver notifies the slave-transmitter that this is the last byte. The slave-transmitter relinquishes the SDA line after detecting the No Acknowledge (NACK) so that the master can issue a STOP condition.

When a master does not want to relinquish the bus with a STOP condition, the master can issue a RESTART condition. This is identical to a START condition except it occurs after the ACK pulse. Operating in master mode, the i2c can then communicate with the same slave using a transfer of a different direction.

Note: The i2c must be completely disabled—if I2C_DYNAMIC_TAR_UPDATE = 0—or inactive on the serial port—if I2C_DYNAMIC_TAR_UPDATE = 1—before the target slave address register (IC_TAR) can be reprogrammed.

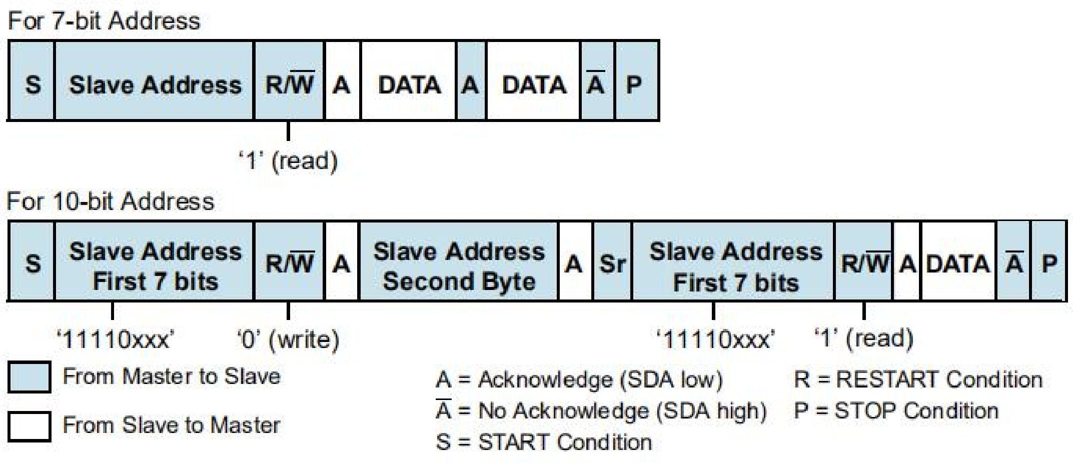


Figure 8.41 Master-Receiver Protocol

START BYTE Transfer Protocol

The START BYTE transfer protocol is set up for systems that do not have an on-board dedicated I2C hardware module. When the Onmicro_i2c is addressed as a slave, it always samples the I2C bus at the highest speed supported so that it never requires a START BYTE transfer. However, when Onmicro_i2c is a master, it supports the generation of START BYTE transfers at the beginning of every transfer in case a slave device requires it. This protocol consists of seven zeros being transmitted followed by a 1, as illustrated in the following figure. This allows the processor that is polling the bus to under-sample the address phase until 0 is detected. Once the micro-controller detects a 0, it switches from the under sampling rate to the correct rate of the master.

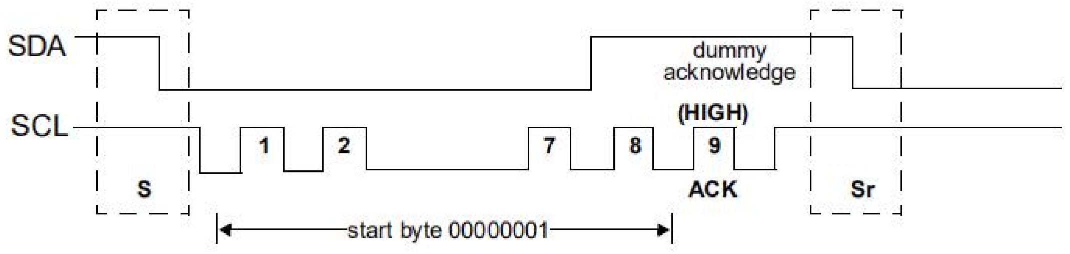


Figure 8.42 START BYTE Transfer

The START BYTE procedure is as follows:

- Master generates a START condition.
- Master transmits the START byte (0000 0001).
- Master transmits the ACK clock pulse. (Present only to conform with the byte handling format used on the bus)

- No slave sets the ACK signal to 0.
- Master generates a RESTART (R) condition.

A hardware receiver does not respond to the START BYTE because it is a reserved address and resets after the RESTART condition is generated.

Multiple Master Arbitration

The Onmicro_i2c bus protocol allows multiple masters to reside on the same bus. If there are two masters on the same I²C-bus, there is an arbitration procedure if both try to take control of the bus at the same time by generating a START condition at the same time. Once a master (for example, a micro-controller) has control of the bus, no other master can take control until the first master sends a STOP condition and places the bus in an idle state.

Arbitration takes place on the SDA line, while the SCL line is 1.The master, which transmits a 1 while the other master transmits 0, loses arbitration and turns off its data output stage. The master that lost arbitration can continue to generate clocks until the end of the byte transfer. If both masters are addressing the same slave device, the arbitration could go into the data phase.

Upon detecting that it has lost arbitration to another master, the Onmicro_i2c will stop generating SCL (ic_clk_oe).

The following figure illustrates the timing of when two masters are arbitrating on the bus.

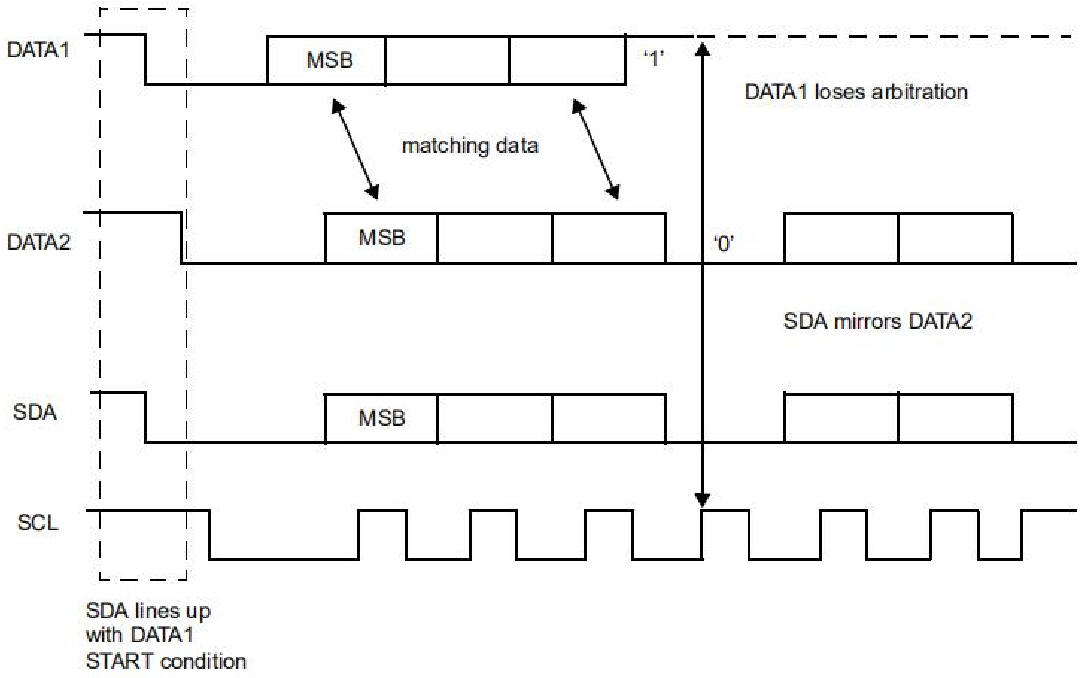


Figure 8.43 Multiple Master Arbitration

For high-speed mode, the arbitration cannot go into the data phase because each master is programmed with a unique high-speed master code. This 8-bitcode is defined by the system designer and is set by writing to the High Speed Master Mode Code Address Register, IC_HS_MADDR. Because the codes are unique, only one master can win arbitration, which occurs by the end of the transmission of the high-speed master code.

Control of the bus is determined by address or master code and data sent by competing masters, so there is no central master nor any order of priority on the bus.

Arbitration is not allowed between the following conditions:

- A RESTART condition and a data bit
- A STOP condition and a data bit
- A RESTART condition and a STOP condition

Note: Slaves are not involved in the arbitration process.

Clock Synchronization

When two or more masters try to transfer information on the bus at the same time, they must arbitrate and synchronize the SCL clock. All masters generate their own clock to transfer messages. Data is valid only during the high period of SCL clock. Clock synchronization is performed using the wired-AND connection to the SCL signal. When the master transitions the SCL clock to 0, the master starts counting the low time of the SCL clock and transitions the SCL clock signal to 1 at the beginning of the next clock period. However, if another master is holding the SCL line to 0, then the master goes into a HIGH wait state until the SCL clock line transitions to 1.

All masters then count off their high time, and the master with the shortest high time transitions the SCL line to 0. The masters then counts out their low time and the one with the longest low time forces the other master into a HIGH wait state. Therefore, a synchronized SCL clock is generated, which is illustrated in the following figure. Optionally, slaves may hold the SCL line low to slow down the timing on the I2C bus.

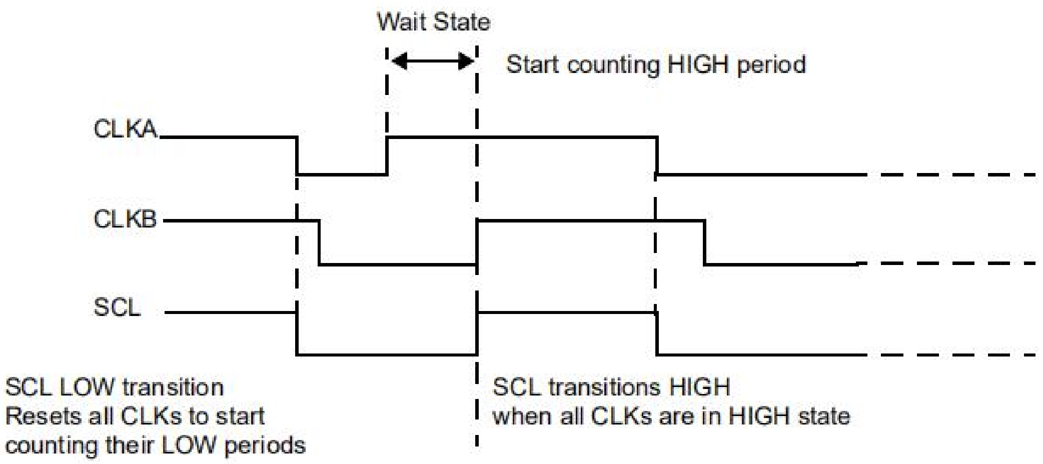

Wait State
Start counting HIGH period
CLKA
CLKB
SCL
SCL LOW transition
Resets all CLKs to start
counting their LOW periods
SCL transitions HIGH
when all CLKs are in HIGH state

Figure 8.44 Multi-Master Clock Synchronization

Operation Modes

This section provides information on operation modes.

Note:It is important to note that the Onmicro_i2c should only be set to operate as an I2C Master, or I2C Slave, but not both simultaneously. This is achieved by ensuring that bit 6 (IC_SLAVE_DISABLE) and 0 (IC_MASTER_MODE) of the **IC_CON** register are never set to 0 and 1, respectively.

Slave Mode Operation

Initial Configuration

To use the Onmicro_i2c as a slave, perform the following steps:

- Disable the Onmicro_i2c by writing a ‘0’ to bit 0 of the IC_ENABLE register.

- Write to the IC_SAR register (bits 9:0) to set the slave address.This is the address to which the Onmicro_i2c responds.
- Write to the IC_CON register to specify which type of addressing is supported (7- or 10-bit by setting bit 3).Enable the Onmicro_i2c in slave-only mode by writing a '0' into bit 6 (IC_SLAVE_DISABLE) and a '0' to bit 0 (MASTER_MODE).

Note: Slaves and masters do not have to be programmed with the same type of addressing 7-bit or10-bit address.For instance,a slave can be programmed with 7-bit addressing and a master with 10-bit addressing, and vice versa.

Enable the Onmicro_i2c by writing a '1' in bit 0 of the IC_ENABLE register.

Note: Depending on the reset values chosen, steps 2 and 3 may not be necessary because the reset values can be configured. For instance, if the device is only going to be a master, there would be no need to set the slave address because you can configure Onmicro_i2c to have the slave disabled after reset and to enable the master after reset. The values stored are static and do not need to be reprogrammed if the Onmicro_i2c is disabled.

Slave-Transmitter Operation for a Single Byte

When another I2C master device on the bus addresses the Onmicro_i2c and requests data, the Onmicro_i2c acts as a slave-transmitter and the following steps occur:

- The other I2C master device initiates an I2C transfer with an address that matches the slave address in the IC_SAR register of the Onmicro_i2c.
- The Onmicro_i2c acknowledges the sent address and recognizes the direction of the transfer to indicate that it is acting as a slave-transmitter.
- The Onmicro_i2c asserts the RD_REQ interrupt (bit 5 of the IC_RAW_INTR_STAT register) and holds the SCL line low. It is in a wait state until software responds.

If the RD_REQ interrupt has been masked,due to IC_INTR_MASK[5] register (M_RD_REQ bit field) being set to 0,then it is recommended that a hardware and/or software timing routine be used to instruct the CPU to perform periodic reads of the IC_RAW_INTR_STAT register.

- Reads that indicate IC_RAW_INTR_STAT[5] (R_RD_REQ bit field) being set to 1 must be treated as the equivalent of the RD_REQ interrupt being asserted.
- Software must then act to satisfy the I2C transfer.
- The timing interval used should be in the order of 10 times the fastest SCL clock period the Onmicro_i2c can handle. For example, for 400 kb/s, the timing interval is 25us.

Note: The value of 10 is recommended here because this is approximately the amount of time required for a single byte of data transferred on the I2C bus.

- If there is any data remaining in the TX FIFO before receiving the read request, then the Onmicro_i2c asserts a TX_ABRT interrupt (bit 6 of the IC_RAW_INTR_STAT register) to flush the old data from the TX FIFO.

Note: Because the Onmicro_i2c's TX FIFO is forced into a flushed/reset state whenever a TX_ABRT event occurs,it is necessary for software to release the Onmicro_i2c from this state by reading the IC_CLR_TX_ABRT register before attempting to write into the TX FIFO.See register IC_RAW_INTR_STAT for more details.

If the TX_ABRT interrupt has been masked,due to of IC_INTR_MASK[6] register (M_TX_ABRT bit field) being set to 0,then it is recommended that re-using the timing routine (described in the previous step),or a similar one,be used to read the IC_RAW_INTR_STAT register.

- Reads that indicate bit 6 (R_TX_ABRT) being set to 1 must be treated as the equivalent of the TX_ABRT interrupt being asserted.
- There is no further action required from software.
- The timing interval used should be similar to that described in the previous step for the IC_RAW_INTR_STAT[5] register.
- Software writes to the IC_DATA_CMD register with the data to be written (by writing a '0' in bit8).
- Software must clear the RD_REQ and TX_ABRT interrupts (bits 5 and 6, respectively) of the IC_RAW_INTR_STAT register before proceeding.

If the RD_REQ and/or TX_ABRT interrupts have been masked, then clearing of the IC_RAW_INTR_STAT register will have already been performed when either the R_RD_REQ or R_TX_ABRT bit has been read as 1.

- The Onmicro_i2c releases the SCL and transmits the byte.
- The master may hold the I2C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

Slave-Receiver Operation for a Single Byte

When another I2C master device on the bus addresses the Onmicro_i2c and is sending data, the Onmicro_i2c acts as a slave-receiver and the following steps occur:

- The other I2C master device initiates an I2 C transfer with an address that matches the Onmicro_i2c's slave address in the IC_SAR register.
- The Onmicro_i2c acknowledges the sent address and recognizes the direction of the transfer to indicate that the Onmicro_i2c is acting as a slave-receiver.
- Onmicro_i2c receives the transmitted byte and places it in the receive buffer.

Note: If the RX FIFO is completely filled with data when a byte is pushed, then an overflow occurs and the Onmicro_i2c continues with subsequent I2C transfers. Because a NACK is not generated, software must recognize the overflow when indicated by the Onmicro_i2c (by the R_RX_OVER bit in the IC_INTR_STAT register) and take appropriate actions to recover from lost data. Hence, there is a real time constraint on software to service the RX FIFO before the latter overflow as there is no way to reapply pressure to the remote transmitting master. You must select a deep enough RX FIFO depth to satisfy the interrupt service interval of their system.

Onmicro_i2c asserts the RX_FULL interrupt (IC_RAW_INTR_STAT[2] register). If the RX_FULL interrupt has been masked, due to setting IC_INTR_MASK[2] register to 0 or setting IC_TX_TL to a value larger than 0, then it is recommended that a timing routine (described in "Slave-Transmitter Operation for a Single Byte" on page 40) be implemented for periodic reads of the IC_STATUS register. Reads of the IC_STATUS register, with bit 3 (RFNE) set at 1, must then be treated by software as the equivalent of the RX_FULL interrupt being asserted.

- Software may read the byte from the IC_DATA_CMD register (bits 7:0).
- The other master device may hold the I2 C bus by issuing a RESTART condition

or release the bus by issuing a STOP condition.

Slave-Transfer Operation For Bulk Transfers

In the standard I2 C protocol, all transactions are single byte transactions and the programmer responds to a remote master read request by writing one byte into the slave's TX FIFO. When a slave (slave-transmitter) is issued with a read request (RD_REQ) from the remote master (master-receiver), at a minimum there should be at least one entry placed into the slave-transmitter's TX FIFO. Onmicro_i2c is designed to handle more data in the TX FIFO so that subsequent read requests can take that data without raising an interrupt to get more data. Ultimately, this eliminates the possibility of significant latencies being incurred between raising the interrupt for data each time had there been a restriction of having only one entry placed in the TX FIFO.

This mode only occurs when Onmicro_i2c is acting as a slave-transmitter. If the remote master acknowledges the data sent by the slave-transmitter and there is no data in the slave's TX FIFO, the Onmicro_i2c holds the I2 C SCL line low while it raises the read request interrupt (RD_REQ) and waits for data to be written into the TX FIFO before it can be sent to the remote master.

If the RD_REQ interrupt is masked, due to bit 5 (M_RD_REQ) of the IC_INTR_STAT register being set to 0, then it is recommended that a timing routine be used to activate periodic reads of the IC_RAW_INTR_STAT register. Reads of IC_RAW_INTR_STAT that return bit 5 (R_RD_REQ) set to 1 must be treated as the equivalent of the RD_REQ interrupt referred to in this section.

The RD_REQ interrupt is raised upon a read request, and like interrupts, must be cleared when exiting the interrupt service handling routine (ISR). The ISR allows you to either write 1 byte or more than 1 byte into the TX FIFO. During the transmission of these bytes to the master, if the master acknowledges the last byte. then the slave must raise the RD_REQ again because the master is requesting for more data.

If the programmer knows in advance that the remote master is requesting a packet of n bytes, then when another master addresses Onmicro_i2c and requests data, the TX FIFO could be written with n number bytes and the remote master receives it as a continuous stream of data. For example, the Onmicro_i2c slave continues to send data to the remote master as long as the remote master is acknowledging the data sent and there is data available in the TX FIFO. There is no need to hold the SCL line low or to issue RD_REQ again.

If the remote master is to receive n bytes from the Onmicro_i2c but the programmer wrote a number of bytes larger than n to the TX FIFO, then when the slave finishes sending the requested n bytes, it clears the TX FIFO and ignores any excess bytes.

The the Onmicro_i2c generates a transmit abort (TX_ABRT) event to indicate the clearing of the TX FIFO in this example. At the time an ACK/NACK is expected, if a NACK is received, then the remote master has all the data it wants. At this time, a flag is raised within the slave's state machine to clear the leftover data in the TX FIFO. This flag is transferred to the processor bus clock domain where the FIFO exists and the contents of the TX FIFO is cleared at that time.

Master Mode Operation

Initial Configuration

The initial configuration procedure for Master Mode Operation depends on the configuration parameter I2C_DYNAMIC_TAR_UPDATE. When set to “Yes” (1), the target address and address format can be changed dynamically without having to disable Onmicro_i2c. This parameter only applies to when Onmicro_i2c is acting as a master because the slave requires the component to be disabled before any changes can be made to the address.

The procedures are very similar and are only different with regard to where the IC_10BITADDR_MASTER bit is set (either bit 4 of IC_CON register or bit 12 of IC_TAR register).

I2C_DYNAMIC_TAR_UPDATE = 0

To use the Onmicro_i2c as a master when the I2C_DYNAMIC_TAR_UPDATE configuration parameter is set to “No” (0), perform the following steps:

- Disable the Onmicro_i2c by writing 0 to the IC_ENABLE register.
- Write to the IC_CON register to set the maximum speed mode supported (bits 2:1) and the desired speed of the Onmicro_i2c master-initiated transfers, either 7-bit or 10-bit addressing (bit 4). Ensure that bit 6 (IC_SLAVE_DISABLE) is written with a ‘1’ and bit 0 (MASTER_MODE) is written with a ‘1’.

Note: Slaves and masters do not have to be programmed with the same type of addressing 7-bit or 10-bit address.For instance,a slave can be programmed with 7-bit addressing and a master with 10-bit addressing, and vice versa.

- Write to the IC_TAR register the address of the I2 C device to be addressed (bits 9:0).This register also indicates whether a General Call or a START BYTE command is going to be performed by I2C.
- Only applicable for high-speed mode transfers. Write to the IC_HS_MADDR register the desired master code for the Onmicro_i2c. The master code is programmer-defined.
- Enable the Onmicro_i2c by writing a ‘1’ in bit 0 of the IC_ENABLE register.
- Now write transfer direction and data to be sent to the IC_DATA_CMD register. If the IC_DATA_CMD register is written before the Onmicro_i2c is enabled, the data and commands are lost as the buffers are kept cleared when Onmicro_i2c is disabled. This step generates the START condition and the address byte on the Onmicro_i2c. Once Onmicro_i2c is enabled and there is data in the TX FIFO, Onmicro_i2c starts reading the data.

Note: Depending on the reset values chosen,steps 2,3,4,and 5 may not be necessary because the reset values can be configured. The values stored are static and do not need to be reprogrammed if the Onmicro_i2c is disabled, with the exception of the transfer direction and data.

I2C_DYNAMIC_TAR_UPDATE = 1

To use the Onmicro_i2c as a master when the I2C_DYNAMIC_TAR_UPDATE configuration parameter is set to “Yes” (1), perform the following steps:

- Disable the Onmicro_i2c by writing 0 to the IC_ENABLE register.
- Write to the IC_CON register to set the maximum speed mode supported for slave operation (bits 2:1) and to specify whether the Onmicro_i2c starts its

transfers in 7/10 bit addressing mode when the device is a slave (bit 3).

- Write to the IC_TAR register the address of the I2C device to be addressed. It also indicates whether a General Call or a START BYTE command is going to be performed by I2C. The desired speed of the Onmicro_i2c master-initiated transfers, either 7-bit or 10-bit addressing, is controlled by the IC_10BITADDR_MASTER bit field (bit 12).
- Only applicable for high-speed mode transfers. Write to the IC_HS_MADDR register the desired master code for the Onmicro_i2c. The master code is programmer-defined.
- Enable the Onmicro_i2c by writing a 1 in the IC_ENABLE register.
- Now write the transfer direction and data to be sent to the IC_DATA_CMD register. If the IC_DATA_CMD register is written before the Onmicro_i2c is enabled, the data and commands are lost as the buffers are kept cleared when Onmicro_i2c is not enabled.

Note: For multiple I2C transfers, perform additional writes to the TX FIFO such that the TX FIFO does not become empty during the I2C transaction. If the TX FIFO is completely emptied at any stage, then further writes to the TX FIFO results in an independent I2C transaction.

Dynamic IC_TAR or IC_10BITADDR_MASTER Update

The Onmicro_i2c supports dynamic updating of the IC_TAR (bits 9:0) and IC_10BITADDR_MASTER (bit 12) bit fields of the IC_TAR register. In order to perform a dynamic update of the IC_TAR register, the I2C_DYNAMIC_TAR_UPDATE configuration parameter must be set to “Yes” (1). You can dynamically write to the IC_TAR register provided the following conditions are met:

- Onmicro_i2c is not enabled (IC_ENABLE=0);
- Onmicro_i2c is enabled (IC_ENABLE=1);
- Onmicro_i2c is not engaged in any Master (tx,rx) operation (IC_STATUS[5]=0);
- Onmicro_i2c is enabled to operate in Master mode (IC_CON[0]=1);
- And there are no entries in the TX FIFO (IC_STATUS[2]=1).

Master Transmit and Master Receive

The Onmicro_i2c supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the I2C Rx/Tx Data Buffer and Command Register (IC_DATA_CMD). The CMD bit [8] should be written to 0 for I2C write operations. Subsequently, a read command may be issued by writing “don’t cares” to the lower byte of the IC_DATA_CMD register, and a 1 should be written to the CMD bit. The Onmicro_i2c master continues to initiate transfers as long as there are commands present in the transmit FIFO. If the transmit FIFO becomes empty, the Onmicro_i2c inserts a STOP condition after completing the current transfers.

Disabling Onmicro_i2c

The register **IC_ENABLE_STATUS** is added to allow software to unambiguously determine when the hardware has completely shutdown in response to the **IC_ENABLE** register being set from 1 to 0. Only one register is required to be monitored, as opposed to monitoring two registers (IC_STATUS and IC_RAW_INTR_STAT) which is a requirement for Onmicro_i2c versions 1.05a or earlier.

Procedure

- Define a timer interval (ti2c_poll) equal to the 10 times the signaling period for the highest I2C transfer speed used in the system and supported by Onmicro_i2c.For example, if the highest I2C transfer mode is 400 kb/s, then this ti2c_poll is 25us.
- Define a maximum time-out parameter, MAX_T_POLL_COUNT, such that if any repeated polling operation exceeds this maximum value, an error is reported.
- Execute a blocking thread/process/function that prevents any further I2 C master transactions to be started by software, but allows any pending transfers to be completed.

Note: This step can be ignored if Onmicro_i2c is programmed to operate as an I2C slave only.

- The variable POLL_COUNT is initialized to zero.
- Set IC_ENABLE to 0.
- Read the IC_ENABLE_STATUS register and test the IC_EN bit (bit 0).Increment POLL_COUNT by one.If POLL_COUNT >= MAX_T_POLL_COUNT,exit with the relevant error code.
- If IC_ENABLE_STATUS[0] is 1,then sleep for ti2c_poll and proceed to the previous step.Otherwise, exit with a relevant success code.

IC_CLK Frequency Configuration

When the Onmicro_i2c is configured as a master, the *CNT registers must be set before any I2 C bus transaction can take place in order to ensure proper I/O timing. The *CNT registers are:

- IC_SS_SCL_HCNT
- IC_SS_SCL_LCNT
- IC_FS_SCL_HCNT
- IC_FS_SCL_LCNT
- IC_HS_SCL_HCNT
- IC_HS_SCL_LCNT

Note: It is not necessary to program any of the *CNT registers if the Onmicro_i2c is enabled to operate only as an I2C slave, since these registers are used only to determine the SCL timing requirements for operation as an I2C master.

Minimum High and Low Counts

When the Onmicro_i2c operates as an I2C master, in both transmit and receive transfers:

- Minimum value that can be programmed in the *_LCNT registers is 8
- Minimum value allowed for the *_HCNT registers is 6

The minimum value of 8 for the *_LCNT registers is due to the time required for the Onmicro_i2c to drive SDA after a negative edge of SCL.

The minimum value of 6 for the *_HCNT register is due to the time required for the Onmicro_i2c to sample SDA during the high period of SCL.

The Onmicro_i2c adds one cycle to the programmed *_LCNT value in order to generate the low period of the SCL clock. This is due to the counting logic for SCL low counting to (*_LCNT+1).

The Onmicro_i2c adds eight cycles to the programmed *_HCNT value in order to generate the high period of the SCL clock. This is due to the following factors:

- The counting logic for SCL high counts to (*_HCNT+1).
- The digital filtering applied to the SCL line incurs a delay of four ic_clk cycles.This filtering includes metastability removal and a 2-out-of-3 majority vote processing on SDA and SCL edges.
- Whenever SCL is driven 1 to 0 by the Onmicro_i2c—that is, completing the SCL high time—an internal logic latency of three ic_clk cycles is incurred.

Consequently,the minimum SCL low time of which the Onmicro_i2c is capable is nine (9) ic_clk periods (8+1),while the minimum SCL high time is fourteen (14) ic_clk periods (6+1+4+3).

Minimum IC_CLK Frequency

This section describes the minimum ic_clk frequencies that the Onmicro_i2c supports for each speed mode, and the associated high and low count values. It should be noted that these limits apply to the Onmicro_i2c in both master and slave modes. The limits for slave mode are required so that the Onmicro_i2c does not break the Thd;dat maximum I2C protocol timing requirement.

Standard and Fast Modes

This section details how to derive a minimum ic_clk value for standard and fast modes of the Onmicro_i2c.

Although the following method shows how to do fast mode calculations, you can also use the same method in order to do calculations for standard mode.

Given conditions and calculations for the minimum Onmicro_i2c ic_clk value in fast mode:

- Fast mode has data rate of 400kb/s; implies SCL period of 1/400khz = 2.5us
- Minimum hcnt value of 14 as a seed value; IC_HCNT_FS = 14
- Protocol minimum SCL high and low times:
  - MIN_SCL_LOWtime_FS = 1300ns
  - MIN_SCL_HIGHtime_FS = 600ns

High-Speed Modes

The method used for standard and fast modes is not enough to derive correct ic_clk values for the high-speed modes. For example, given a high-speed mode with a 100pf bus loading, using the standard and fast modes method produces the following:

- IC_LCNT_HS = 17
- IC_HCNT_HS = 14
- ic_clk = 105.4 Mhz

Depending on glitch suppression, the Onmicro_i2c can take up to nine ic_clk cycles to drive SDA after a negative edge of SCL; however, the protocol requires a maximum Thd;dat of 70ns for this mode. For example:

- 105Mhz => IC_CLK_PERIOD = 9.48ns
- 9.48ns * 9 = 85.32ns
- 85.32ns is a maximum violation of Thd;dat

Thus, these values cannot be used.To satisfy this rule, IC_CLK_PERIOD can be derived as follows:

70ns/9 = 7.77nS

From this value, high and low count values can be derived:

IC_LCNT_HS * IC_CLK_PERIOD ≥ MIN_SCL_LOWtime_HS

IC_LCNT_HS * 7.77ns ≥ 160ns

IC_LCNT_HS≥ 21

The minimum value of 14 for IC_HCNT_HS easily accommodates the MIN_SCL_HIGHtime_HS requirement of 60ns for this requirement. Therefore:

MIN_SCL_HIGHtime_FS = 14

MIN_SCL_HIGHtime_FS = 21

This derivation gives a baud rate higher than the allowed 3.4Mb/s, but the high or low count can be scaled up to give the desired baud rate.

Given:

SCL_PERIOD = 1/3.4Mhz = 294ns

IC_CLK_PERIOD = 7.77ns

Required:

roundup(294/7.77) = 38 ic_clk periods for a baud rate of 3.4Mb/s

To achieve this, the low count must be scaled up by 3 to give:

MIN_SCL_HIGHtime_FS = 14

MIN_SCL_HIGHtime_FS = 24

The values for HS mode with a bus loading of 400pf can be derived in the same way.

The following table lists the minimum ic_clk values for all modes with high and low count values.

Speed Mode	**Ic_clk$_{freq}$ (MHz)**	**SCL Low Count**	**SCL Low Program Value**	**SCL Low Time**	**SCL High Count**	**SCL High Program Value**	**SCL High Time**
SS	2.7	13	12	4.7μs	14	6	5.2μs
FS	12.0	16	15	1.33μs	14	6	1.16μs
HS (400pf)	60.2	22	21	365ns	14	6	232ns
HS (100pf)	128.5	24	23	186ns	14	6	108ns

Table 8.7 ic_clk in Relation to High and Low Counts

Note: The IC_*_SCL_LCNT and IC_*_SCL_HCNT registers are programmed using the SCL low and high program values in Table 8.7, which are calculated using SCL low count minus 1, and SCL high counts minus 8, respectively.

Calculating High and Low Counts

The calculations below show how to calculate SCL high and low counts for each speed mode in the Onmicro_i2c.For the calculations to work, the ic_clk frequencies used must not be less than the minimum ic_clk frequencies specified in Table 8.7.

The Onmicro_i2c coreConsultant GUI can automatically calculate SCL high and low count values. By specifying an integer ic_clk period value in nanoseconds for the IC_CLK_PERIOD parameter, SCL high and low count values are automatically calculated for each speed mode. The ic_clk period must not specify a clock of a lower frequency than required for all supported speed modes. It is possible that the automatically calculated values may result in a baud rate higher than the maximum rate specified by the protocol. If this happens, either the low or high count values can be scaled up to reduce the baud rate.

The minimum IC_CLK calculations for high-speed mode show how to do this; for details, refer to “High-Speed Modes”.

SDA Hold Time

The I2C protocol specification requires 300ns of hold time on the SDA signal (tHD;DAT) in standard and fast speed modes, and a hold time long enough to bridge the undefined part between logic 1 and logic 0 of the falling edge of SCL in high speed mode.

Board delays on the SCL and SDA signals can mean that the hold-time requirement is met at the I2 C master, but not at the I2C slave (or vice-versa). As each application will encounter differing board delays, the Onmicro_i2c contains a software programmable register (IC_SDA_HOLD) to enable dynamic adjustment of the SDA hold-time.

The IC_SDA_HOLD register can be used to alter the timing of the generated SDA (ic_data_oe) signal by the Onmicro_i2c. Each value in the IC_SDA_HOLD register represents a unit of one ic_clk period.

When the Onmicro_i2c is operating in Master Mode, the minimum tHD:DAT timing is one ic_clk period.

Therefore even when IC_SDA_HOLD has a value of zero, the Onmicro_i2c will drive SDA (ic_data_oe) one ic_clk cycle after driving SCL (ic_clk_oe) to logic 0. For all other values of IC_SDA_HOLD, the following is true:

- Drive on SDA (ic_data_oe) will occur IC_SDA_HOLD ic_clk cycles after driving SCL (ic_clk_oe) to logic0.

When the Onmicro_i2c is operating in Slave Mode, the minimum tHD:DAT timing is eight ic_clk periods.

This delay is to allow for synchronization and filtering on the SCL (ic_clk_in) sample. Therefore, even when IC_SDA_HOLD has a value less than 8, the Onmicro_i2c will drive SDA (ic_data_oe) eight ic_clk cycles after SCL (ic_clk_in) has transitioned to logic 0. For all other values of IC_SDA_HOLD, the following is true:

- Drive on SDA (ic_data_oe) will occur IC_SDA_HOLD ic_clk cycles after SCL (ic_clk_in) has transitioned to logic 0.

If different SDA hold times are required for different speed modes, the IC_SDA_HOLD register must be reprogrammed when the speed mode is being changed. The IC_SDA_HOLD register cab be programmed only when the Onmicro_i2c is disabled (IC_ENABLE = 0).

The reset value of the IC_SDA_HOLD register can be set via the coreConsultant parameter IC_DEFAULT_SDA_HOLD.

The following figure shows the tHD:DAT timing generated by the Onmicro_i2c operating in Master Mode when IC_SDA_HOLD = 3.

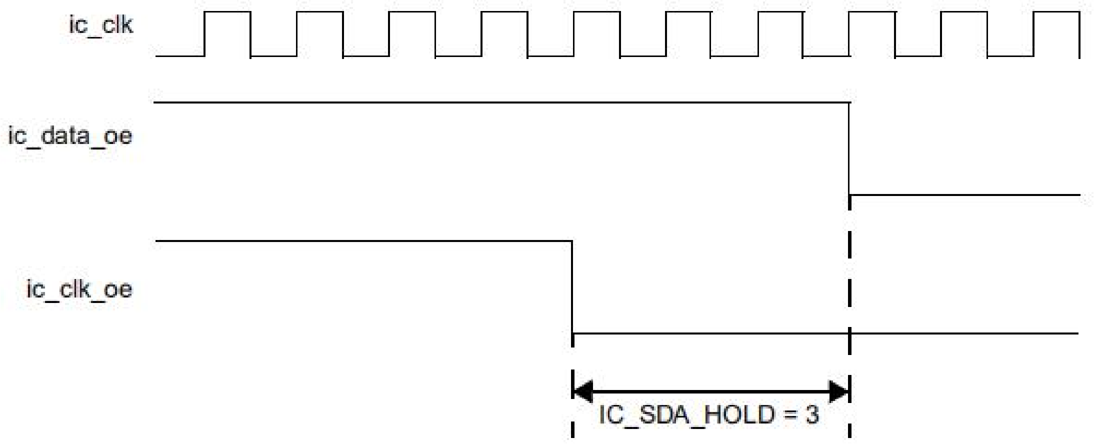


**Figure 8.45 Onmicro_i2c Master Implementing tHD;
DAT When IC_SDA_HOLD = 3**

8.10.4 I2Cx Register Map

Offset	Name	Description
0x0000	I2C_CON	I2C control
0x0004	I2C_TAR	I2C target address
0x0008	I2C_SAR	I2C slave address
0x000C	I2C_HS_MADDR	I2C HS Master Mode Code Address
0x0010	I2C_DATA_CMD	I2C Rx/Tx Data Buffer and Command
0x0014	I2C_SS_SCL_HCNT	Standard speed I2C Clock SCL High Count
0x0018	I2C_SS_SCL_LCNT	Standard speed I2C Clock SCL Low Count
0x001C	I2C_FS_SCL_HCNT	Fast speed I2C Clock SCL High Count
0x0020	I2C_FS_SCL_LCNT	Fast speed I2C Clock SCL Low Count
0x0024	I2C_HS_SCL_HCNT	High speed I2C Clock SCL High Count
0x0028	I2C_HS_SCL_LCNT	High speed I2C Clock SCL Low Count
0x002C	I2C_INTR_STAT	I2C Interrupt Status
0x0030	I2C_INTR_MASK	I2C Interrupt Mask
0x0034	I2C_RAW_INTR_STAT	I2C Raw Interrupt Status
0x0038	I2C_RX_TL	I2C Receive FIFO Threshold
0x003C	I2C_TX_TL	I2C Transmit FIFO Threshold
0x0040	I2C_CLR_INTR	Clear Combined and Individual Interrupts
0x0044	I2C_CLR_RX_UNDER	Clear RX_UNDER Interrupt
0x0048	I2C_CLR_RX_OVER	Clear RX_OVER Interrupt
0x004C	I2C_CLR_TX_OVER	Clear TX_OVER Interrupt
0x0050	I2C_CLR_RD_REQ	Clear RD_REQ Interrupt
0x0054	I2C_CLR_TX_ABRT	Clear TX_ABRT Interrupt

0x0058	I2C_CLR_RX_DONE	Clear RX_DONE Interrupt
0x005C	I2C_CLR_ACTIVITY	Clear ACTIVITY Interrupt
0x0060	I2C_CLR_STOP_DET	Clear STOP_DET Interrupt
0x0064	I2C_CLR_START_DET	Clear START_DET Interrupt
0x0068	I2C_CLR_GEN_CALL	Clear GEN_CALL Interrupt
0x006C	I2C_ENABLE	I2C Enable
0x0070	I2C_STATUS	I2C Status register
0x0074	I2C_TXFLR	Transmit FIFO Level Register
0x0078	I2C_RXFLR	Receive FIFO Level Register
0x007C	I2C_SDA_HOLD	SDA hold time length register
0x0080	I2C_TX_ABRT_SOURCE	I2C Transmit Abort Status Register
0x0084	I2C_SLV_DATA_NACK_ONLY	Generate SLV_DATA_NACK
0x0088	I2C_DMA_CR	DMA Control Register for transmit and receive handshaking interface
0x008C	I2C_DMA_TDLR	DMA Transmit Data Level
0x0090	I2C_DMA_RDLR	DMA Receive Data Level
0x0094	I2C_SDA_SETUP	I2C SDA Setup Register
0x0098	I2C_ACK_GENERAL_CALL	I2C ACK General Call Register
0x009C	I2C_ENABLE_STATUS	I2C Enable Status Register
0x00f4	I2C_COMP_PARAM_1	Component Parameter Register
0x00f8	I2C_COMP_VERSION	Component Version ID
0x00fc	I2C_COMP_TYPE	Design Ware Component Type Register
0x00a0	I2C_CON1	I2C control
0x00b0	I2C_TIMEOUT	I2C timeout control
0x00b4	I2C_CLR_TIME_OUT	I2C timeout interrupt clear

I2C_CON address offset: 0x0000

Bit	R/W	Reset	Name	Description
31:7	N/A	0x0	N/A	reserved
6	RW	0x1	IC_SLAVE_DIS	This bit controls whether I2C has its slave disabled 0: slave is enable 1: slave is disable
5	RW	0x1	IC_RESTART	Determines whether RESTART conditions may be sent when acting as a master 0: disable 1: enable
4	R	0x1	MASTER_10BIT	When acting as a master, the i2c responds only to 10-bit address
3	RW	0x1	SLAVE_10BIT	When acting as a slave, this bit controls

				whether the i2c responds to 7- or 10-bit addresses. 0: 7-bit addressing 1: 10-bit addressing
2:1	RW	0x3	SPEED	These bits control at which speed the i2c operates 1: standard mode (100 kbit/s) 2: fast mode (400kbit/s) 3: high speed mode (3.4 Mbit/s)
0	RW	0x1	MASTER_MODE	This bit controls whether the master is enabled. 0: master disabled 1: master enabled

I2C_TAR address offset: 0x0004

Bit	R/W	Reset	Name	Description
31:13	N/A	0x0	N/A	reserved
12	RW	0x1	MASTER_10BIT_SEL	This bit controls whether the i2c starts its transfers in 7- or 10-bit addressing mode when acting as a master. 0: 7-bit addressing 1: 10-bit addressing
11	RW	0x0	SPECIAL	This bit indicates whether software performs a General Call or START BYTE command. 0: ignore bit 10 GC_OR_START and use IC_TAR normally 1: perform special I2C command as specified in GC_OR_START bit
10	RW	0x0	GC_OR_START	If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a General Call or START byte command is to be performed by the i2c 0: General Call Address 1: START BYTE
9:0	RW	0x55	IC_TAR	This is the target address for any master transaction

I2C_SAR address offset: 0x0008

Bit	R/W	Reset	Name	Description
31:10	N/A	0x0	N/A	reserved
9:0	RW	0x55	IC_SAR	The IC_SAR holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only IC_SAR[6:0] is used.

I2C_HS_MADDR address offset: 0x000c

Bit	R/W	Reset	Name	Description
31:3	N/A	0x0	N/A	reserved
2:0	RW	0x1	IC_HS_MAR	This bit field holds the value of the I2C HS mode master code.

I2C_DATA_CMD address offset: 0x0010

Bit	R/W	Reset	Name	Description
31:9	N/A	0x0	N/A	reserved
8	RW	0x0	CMD	This bit controls whether a read or a write is performed. This bit does not control the direction when the i2c acts as a slave. It controls only the direction when it acts as a master. 1 = Read 0 = Write
7:0	RW	0x0	DAT	This register contains the data to be transmitted or received on the I2C bus.

I2C_SS_SCL_HCNT address offset: 0x0014

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x190	I2C_SS_SCL_HCNT	This register sets the SCL clock high-period count for standard speed.

I2C_SS_SCL_LCNT address offset: 0x0018

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x1d6	I2C_SS_SCL_LCNT	This register sets the SCL clock low-period count for standard speed.

I2C_FS_SCL_HCNT address offset: 0x001c

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x3c	I2C_FS_SCL_HCNT	This register sets the SCL clock high-period count for fast speed.

I2C_FS_SCL_LCNT address offset: 0x0020

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x82	I2C_FS_SCL_LCNT	This register sets the SCL clock low-period count for fast speed.

I2C_HS_SCL_HCNT address offset: 0x0024

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0xc	I2C_HS_SCL_HCNT	This register sets the SCL clock high-period count for high speed.

I2C_HS_SCL_LCNT address offset: 0x0028

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x20	I2C_HS_SCL_LCNT	This register sets the SCL clock low-period count for high speed.

I2C_INTR_STAT address offset: 0x002c

Bit	R/W	Reset	Name	Description
31:13	N/A	0x0	N/A	reserved
12	R	0x0	R_TIME_OUT	Set time out interrupt when i2c bus no response for a user defined time
11	R	0x0	R_GEN_CALL	Set only when a General Call address is received and it is acknowledged
10	R	0x0	R_START_DET	Indicates whether a START or RESTART condition has occurred
9	R	0x0	R_STOP_DET	Indicates whether a STOP condition has occurred
8	R	0x0	R_ACTIVITY	This bit captures I2C activity and stays set until it is cleared
7	R	0x0	R_RX_DONE	When the I2C is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte
6	R	0x0	R_TX_ABRT	This bit indicates if an I2C transmitter, is unable to complete the actions on the contents of the transmit FIFO.
5	R	0x0	R_RD_REQ	This bit is set to 1 when another I2C master is attempting to read data from one I2C slave
4	R	0x0	R_TX_EMPTY	This bit is set to 1 when the transmit buffer is at or below the threshold value
3	R	0x0	R_TX_OVER	Set during transmit if the transmit buffer is filled to the threshold value
2	R	0x0	R_RX_FULL	Set when the receive buffer reaches or goes above the threshold value
1	R	0x0	R_RX_OVER	Set if the receive buffer is completely filled

0	R	0x0	R_RX_UNDER	Set if the processor attempts to read the receive buffer when it is empty

I2C_INTR_MASK address offset: 0x0030

Bit	R/W	Reset	Name	Description
31:13	N/A	0x0	N/A	reserved
12	RW	0x0	M_TIME_OUT	TIME_OUT interrupt mask
11	RW	0x1	M_GEN_CALL	GEN_CALL interrupt mask
10	RW	0x1	M_START_DET	START_DET interrupt mask
9	RW	0x1	M_STOP_DET	STOP_DET interrupt mask
8	RW	0x1	M_ACTIVITY	ACTIVITY interrupt mask
7	RW	0x1	M_RX_DONE	RX_DONE interrupt mask
6	RW	0x1	M_TX_ABRT	TX_ABRT interrupt mask
5	RW	0x1	M_RD_REQ	RD_REQ interrupt mask
4	RW	0x1	M_TX_EMPTY	TX_EMPTY interrupt mask
3	RW	0x1	M_TX_OVER	TX_OVER interrupt mask
2	RW	0x1	M_RX_FULL	RX_FULL interrupt mask
1	RW	0x1	M_RX_OVER	RX_OVER interrupt mask
0	RW	0x1	M_RX_UNDER	RX_UNDER interrupt mask

I2C_RAW_INTR_STAT address offset: 0x0034

Bit	R/W	Reset	Name	Description
31:13	N/A	0x0	N/A	reserved
12	R	0x0	TIME_OUT	TIME_OUT raw interrupt status
11	R	0x0	GEN_CALL	GEN_CALL raw interrupt status
10	R	0x0	START_DET	START_DET raw interrupt status
9	R	0x0	STOP_DET	STOP_DET raw interrupt status
8	R	0x0	ACTIVITY	ACTIVITY raw interrupt status
7	R	0x0	RX_DONE	RX_DONE raw interrupt status
6	R	0x0	TX_ABRT	TX_ABRT raw interrupt status
5	R	0x0	RD_REQ	RD_REQ raw interrupt status
4	R	0x0	TX_EMPTY	TX_EMPTY raw interrupt status
3	R	0x0	TX_OVER	TX_OVER raw interrupt status
2	R	0x0	RX_FULL	RX_FULL raw interrupt status
1	R	0x0	RX_OVER	RX_OVER raw interrupt status
0	R	0x0	RX_UNDER	RX_UNDER raw interrupt status

I2C_RX_TL address offset: 0x0038

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	RW	0x0	RX_TL	Receive FIFO Threshold Level

I2C_TX_TL address offset: 0x003c

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	RW	0x0	TX_TL	Transmit FIFO Threshold Level

I2C_CLR_INTR address offset: 0x0040

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	CLR_INTR	Read this register to clear the combined interrupt, all individual interrupts

I2C_CLR_RX_UNDER address offset: 0x0044

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	CLR_RX_UNDER	Read this register to clear the RX_UNDER interrupt

I2C_CLR_RX_OVER address offset: 0x0048

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	CLR_RX_OVER	Read this register to clear the RX_OVER interrupt

I2C_CLR_TX_OVER address offset: 0x004c

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	CLR_TX_OVER	Read this register to clear the TX_OVER interrupt

I2C_CLR_RD_REQ address offset: 0x0050

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	CLR_RD_REQ	Read this register to clear the RD_REQ interrupt

I2C_CLR_TX_ABRT address offset: 0x0054

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	CLR_TX_ABRT	Read this register to clear the TX_ABRT interrupt

I2C_CLR_RX_DONE address offset: 0x0058

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	CLR_RX_DONE	Read this register to clear the RX_DONE interrupt

I2C_CLR_ACTIVITY address offset: 0x005c

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	CLR_ACTVITY	Read this register to clear the ACTIVITY interrupt

I2C_STOP_DET address offset: 0x0060

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	CLR_STOP_DET	Read this register to clear the STOP_DET interrupt

I2C_START_DET address offset: 0x0064

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	CLR_START_DET	Read this register to clear the START_DET interrupt

I2C_GEN_CALL address offset: 0x0068

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	CLR_GEN_CALL	Read this register to clear the GEN_CALL interrupt

I2C_ENABLE address offset: 0x006c

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	ENABLE	Controls whether the i2c is enabled. 0: Disables i2c (TX and RX FIFOs are held in an erased state) 1: Enables i2c

I2C_STATUS address offset: 0x0070

Bit	R/W	Reset	Name	Description
31:7	N/A	0x0	N/A	reserved
6	R	0x0	SLV_ACTIVITY	Slave FSM Activity Status 0: Slave FSM is in IDLE state

				1: Slave FSM is not in IDLE state
5	R	0x0	MST_ACTIVITY	Master FSM Activity Status 0: Master FSM is in IDLE state 1: Master FSM is not in IDLE state
4	R	0x0	RFF	Receive FIFO Completely Full. 0: Receive FIFO is not full 1: Receive FIFO is full
3	R	0x0	RFNE	Receive FIFO Not Empty 0: Receive FIFO is empty 1: Receive FIFO is not empty
2	R	0x1	TFE	Transmit FIFO Completely Empty 0: Transmit FIFO is not empty 1: Transmit FIFO is empty
1	R	0x1	TFNF	Transmit FIFO Not Full 0: Transmit FIFO is full 1: Transmit FIFO is not full
0	R	0x0	ACTIVITY	I2C Activity Status.

I2C_TXFLR address offset: 0x0074

Bit	R/W	Reset	Name	Description
31:6	N/A	0x0	N/A	reserved
5:0	R	0x0	TXFLR	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

I2C_RXFLR address offset: 0x0078

Bit	R/W	Reset	Name	Description
31:6	N/A	0x0	N/A	reserved
5:0	R	0x0	RXFLR	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO.

I2C_STA_HOLD address offset: 0x007c

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x1	I2C_SDA_HOLD	Sets the required SDA hold time

I2C_TX_ABRT_SOURCE address offset: 0x0080

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15	R	0x0	ABRT_SLVRD_INTX	1: When I2C slave is requested to transmit data to a master and user writes a 1 in CMD (bit 8) of IC_DATA_CMD register.
14	R	0x0	ABRT_SLV_ARBLOST	1: Slave lost the bus while

				transmitting data to a remote master
13	R	0x0	ABRT_SLVFLUSH_TXFIFO	1: Slave has received a read command and some data exists in the TX FIFO so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO.
12	R	0x0	ARB_LOST	1: I2C has lost arbitration
11	R	0x0	ABRT_MASTER_DIS	1: User tries to initiate a Master operation with the Master mode disabled.
10	R	0x0	ABRT_10B_RD_NORSTRT	1: The restart is disabled and the master sends a read command in 10-bit addressing mode.
9	R	0x0	ABRT_SBYTE_NORSTRT	1: The restart is disabled and the user is trying to send a START Byte
8	R	0x0	ABRT_HS_NORSTRT	1: The restart is disabled and the user is trying to use the master to transfer data in High Speed
7	R	0x0	ABRT_SBYTE_ACKDET	1: Master has sent a START Byte and the START Byte was acknowledged
6	R	0x0	ABRT_HS_ACKDET	1: Master is in High Speed mode and the High Speed Master code was acknowledged
5	R	0x0	ABRT_GCALL_READ	1: i2c in master mode sent a General Call but the user programmed byte following the General Call is a read
4	R	0x0	ABRT_GCALL_NOACK	1: i2c in master mode sent a General Call and no slave on the bus acknowledged the General Call.
3	R	0x0	ABRT_TXDATA_NOACK	1: This is a master-mode only bit. Master has received an acknowledgement for the address, but when it sent data following the address, it did not receive an acknowledge from the remote slave
2	R	0x0	ABRT_10ADDR2_NOACK	1: Master is in 10-bit addressing mode and the second 10-bit address byte was not acknowledged by any slave.
1	R	0x0	ABRT_10ADDR1_NOACK	1: Master is in 10-bit addressing

				mode and the first 10-bit address byte was not acknowledged by any slave.
0	R	0x0	ABRT_7B_ADDR_NOACK	1: Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave.

I2C_SLV_DATA_NACK_ONLY address offset: 0x0084

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	NACK	Generate NACK. This NACK generation only occurs when i2c is a slave-receiver. 1 = generate NACK after data byte received 0 = generate NACK/ACK normally

I2C_DMA_CR address offset: 0x0088

Bit	R/W	Reset	Name	Description
31:2	N/A	0x0	N/A	reserved
1	RW	0x0	TDMAE	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. 0 = Transmit DMA disabled 1 = Transmit DMA enabled
0	RW	0x0	RDMAE	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel. 0 = Receive DMA disabled 1 = Receive DMA enabled

I2C_DMA_TDLR address offset: 0x008c

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	RW	0x0	DMATDL	Transmit Data Level. This bit field controls the level at which a DMA request is triggered

I2C_DMA_RDLR address offset: 0x0090

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	RW	0x0	DMARDL	Receive Data Level. This bit field controls the level at which a DMA request is triggered

I2C_SDA_SETUP address offset: 0x0094

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	RW	0x64	SDA_SETUP	SDA setup time

I2C_ACK_GENERAL_CALL address offset: 0x0098

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x1	ACK_GEN_CALL	ACK General Call. When set to 1, i2c responds with a ACK (by asserting ic_data_oe) when it receives a General Call. When set to 0, the i2c does not generate General Call interrupts.

I2C_ENABLE_STATUS address offset: 0x009C

Bit	R/W	Reset	Name	Description
31:3	N/A	0x0	N/A	reserved
2	R	0x0	SLV_RX_DATA_LOST	Slave Received Data Lost.
1	R	0x0	SLV_DIS_IN_BUSY	Slave Disabled While Busy (Transmit, Receive).
0	R	0x0	I2C_EN	ic_en Status.

I2C_CON1 address offset: 0x00A0

Bit	R/W	Reset	Name	Description
31	W	0x0	I2C_EN_CLR	Write 1 to clear i2c_en, self clear
30:26	N/A	0x0	N/A	reserved
25:24	RW	0x0	TRIG_SRC_EDGE_SEL	Trigger source edge select: 0: posedge 1: negedge 3: both edge
23:20	RW	0x0	TRIG_SRC_SEL	Trigger source select
19:17	N/A	0x0	N/A	reserved
16	W	0x0	RD_DATA_NUM_UPDATE	Write 1 to update RD_DATA_NUM, self clear
15:13	N/A	0x0	N/A	reserved
12	RW	0x0	I2C_RX_EN	1: I2C RX 0: I2C TX
11:0	RW	0x0	RD_DATA_NUM	I2C read byte numbers

I2C_TIMEOUT address offset: 0x00B0

Bit	R/W	Reset	Name	Description
31	RW	0x0	I2C_EN_TIMEOUT	I2C timeout interrupt enable
30:0	RW	0x0	I2C_TIMEOUT_CNT	The timeout interrupt trigger level

I2C_CLR_TIME_OUT address offset: 0x00B4

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	R	0x0	CLR_TIME_OUT	Read this register to clear the TIME_OUT interrupt

8.11 TIMER

8.11.1 Introduction

The Timer includes three identical 32-bit Timer Counter channels. Each channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation. Each channel drives an internal interrupt signal which can be programmed to generate processor interrupts.

8.11.2 Main features

- 32-bit up,down,up/down auto-reload counter.
- 32-bit programmable prescaler.(allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 232-1).
- Up to 4 independent channels for:
  - Input Capture
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time.
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- Break input to put the timer’s output signals in reset state or in a known state.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input
- Support for incremental (orthogonal) encoders and Hall sensor circuits for positioning.
- Trigger input as external clock or periodic current management.

8.11.3 Function Description

8.11.3.1.Block Diagram

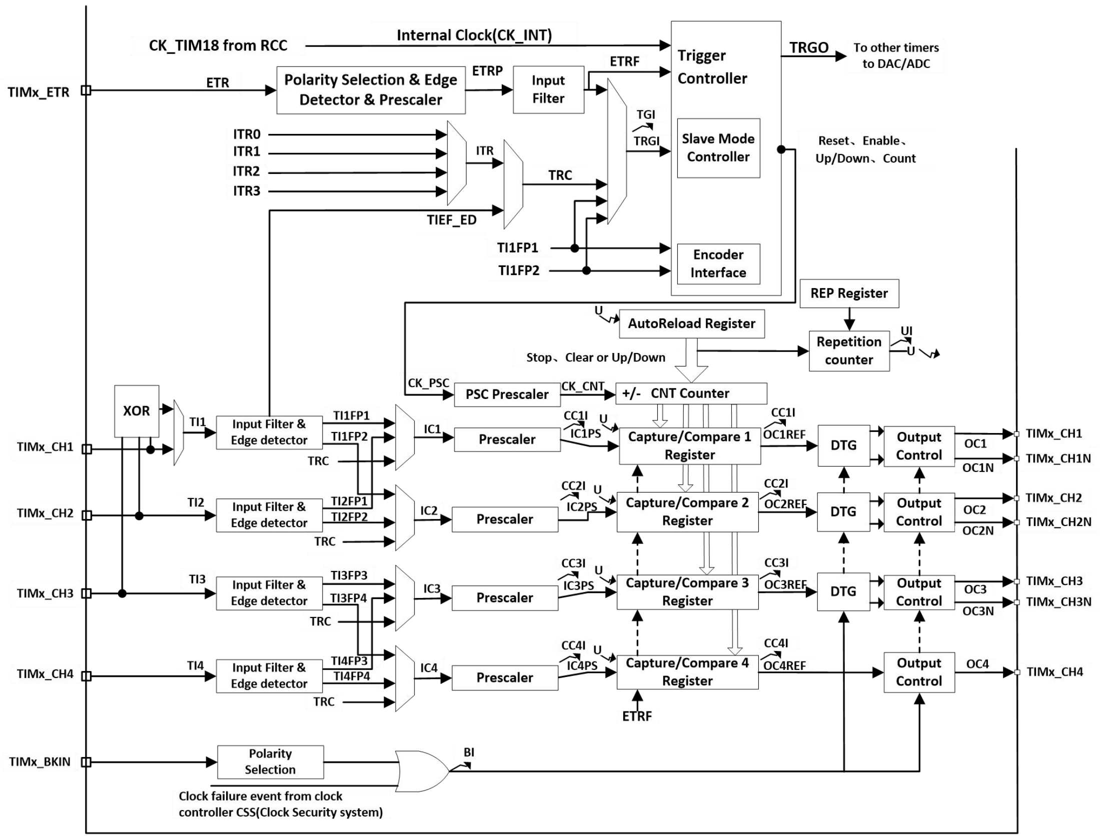


Figure 8.46 General timer block

Note: Interrupt and DMA output

Event

Reg According to the setting of the control bit, the contents of the preload register are transferred to the working register during the U event

8.11.3.2.Time-base unit

The main block of the programmable advanced-control timer is a 32-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software.This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded.Writing to or reading from the auto-reload register accesses the preload register.The content of the preload register are transferred into the shadow register permanently or at each update event (UEV),depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register.It can also be generated by software.The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set.

Note: That the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description:

The prescaler can divide the counter clock frequency by any factor between 1 and 2^{32}-1. It is based on a 32-bit counter controlled through a 32-bit register (in the TIMx_PSC register).It can be changed on the fly as this control register is buffered.The new prescaler ratio is taken into account at the next update event.

The following figures give some examples of the counter behavior when the prescaler ratio is changed on the fly:

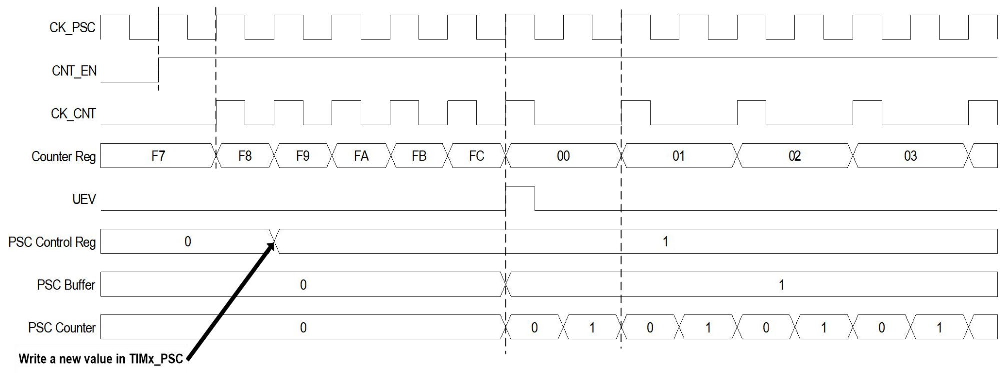


Figure 8.47 Counter timing diagram

with prescaler division change from 1 to 2

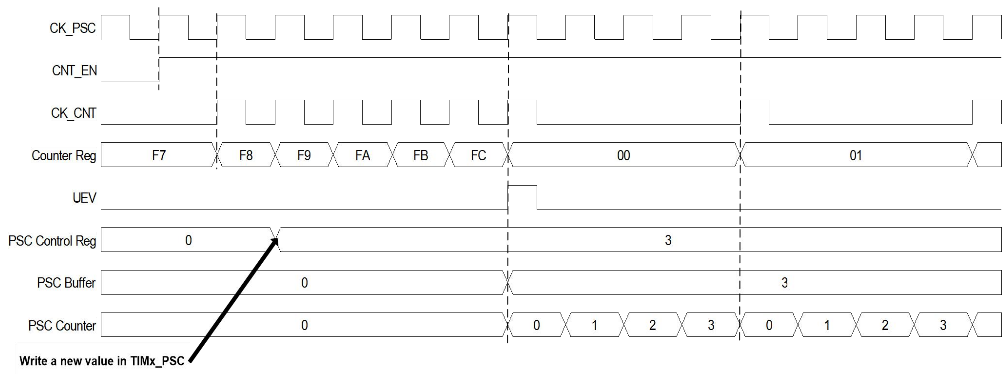


Figure 8.48 Counter timing diagram

with prescaler division change from 1 to 4

8.11.3.3.Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register.This is to avoid updating the shadow registers while writing new values in the preload registers.Then no update event occurs until the UDIS bit has been written to 0. However,the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The auto-reload shadow register is updated with the preload value (TIMx_ARR).
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

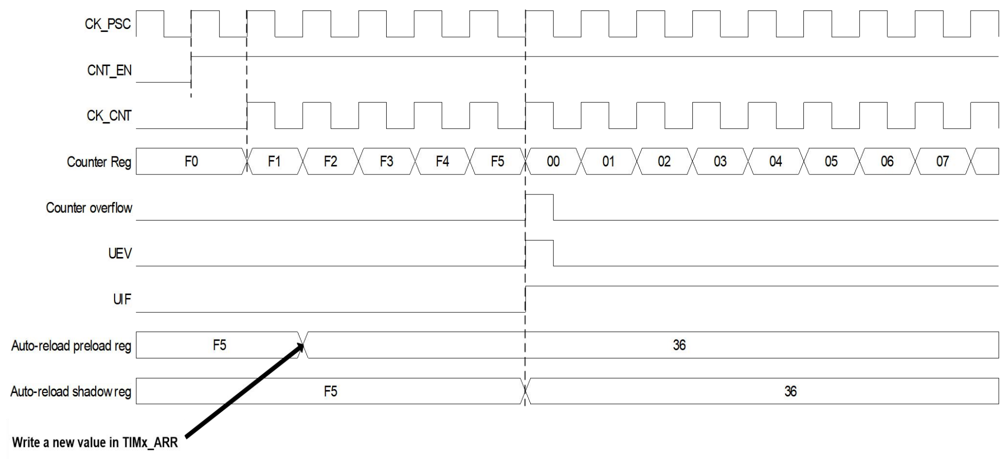


Figure 8.49 Counter timing diagram

update event when ARPE=1 (TIMx_ARR preloaded)

Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn’t change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

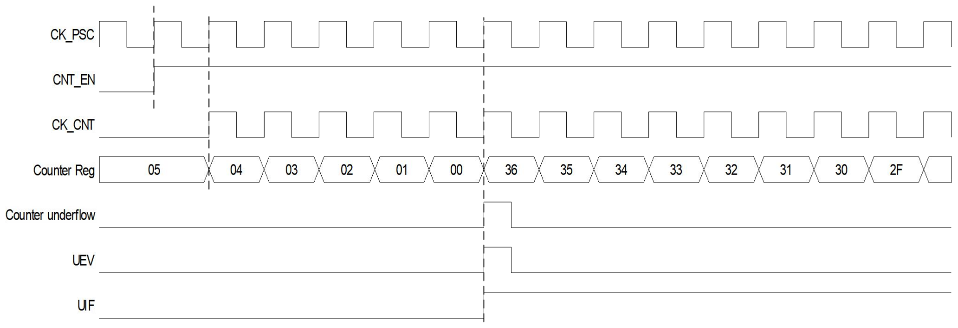


Figure 8.50 Counter timing diagram, internal clock divided by 1

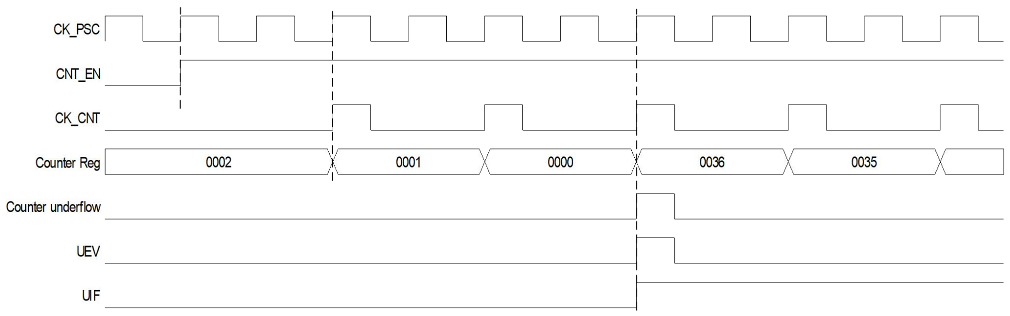


Figure 8.51 Counter timing diagram, internal clock divided by 2

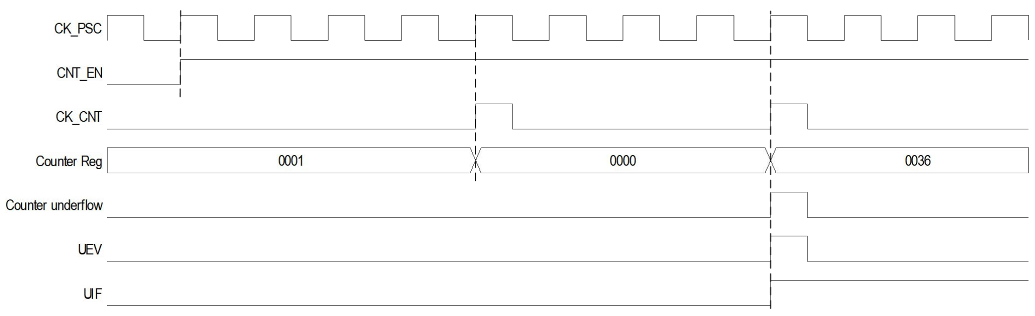


Figure 8.52 Counter timing diagram, internal clock divided by 4

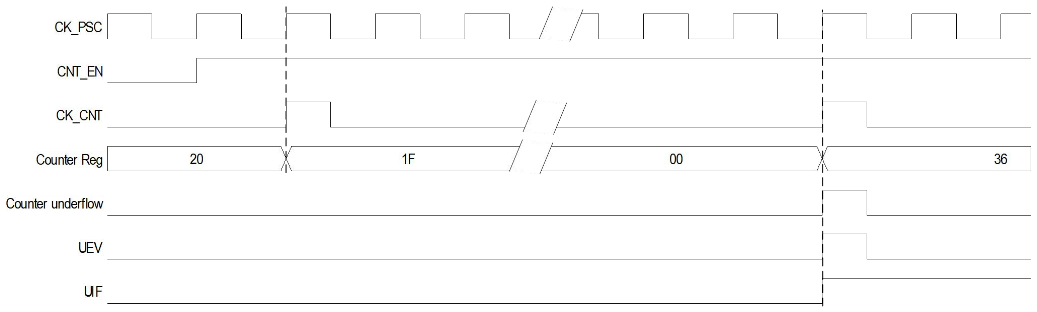


Figure 8.53 Counter timing diagram, internal clock divided by N

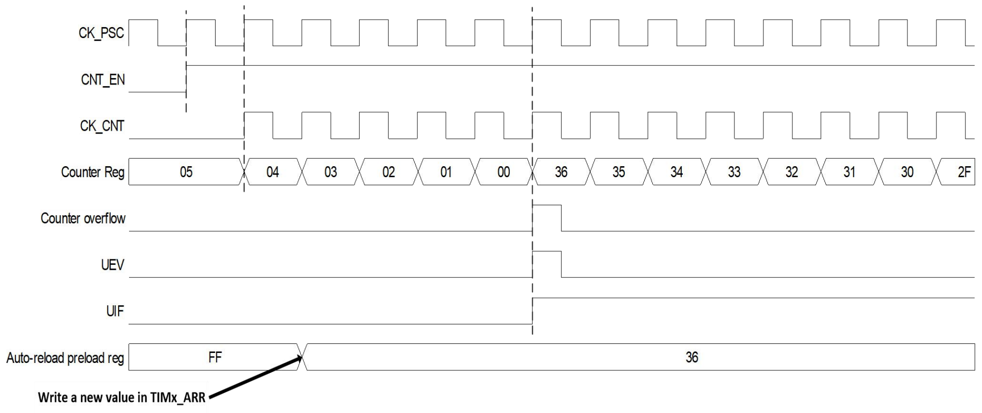


Figure 8.54 Counter timing diagram,

update event when repetition counter is not used

Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto- reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register).Note that if the update source is a counter overflow,the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

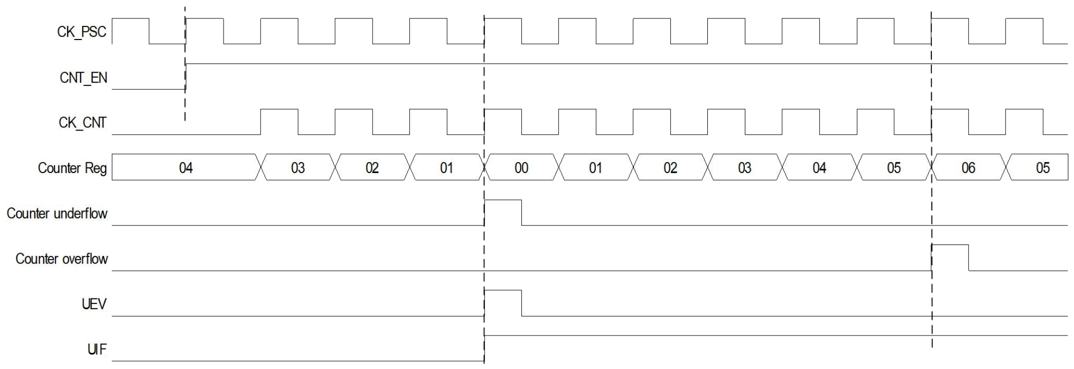


Figure 8.55 Counter timing diagram,

internal clock frequency division factor is 1, TIMx_ARR=0x6

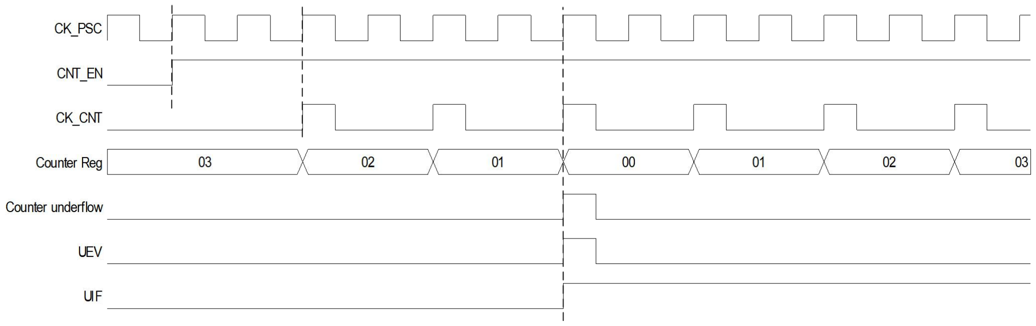


Figure 8.56 Counter sequence diagram,

internal clock frequency division factor is 2

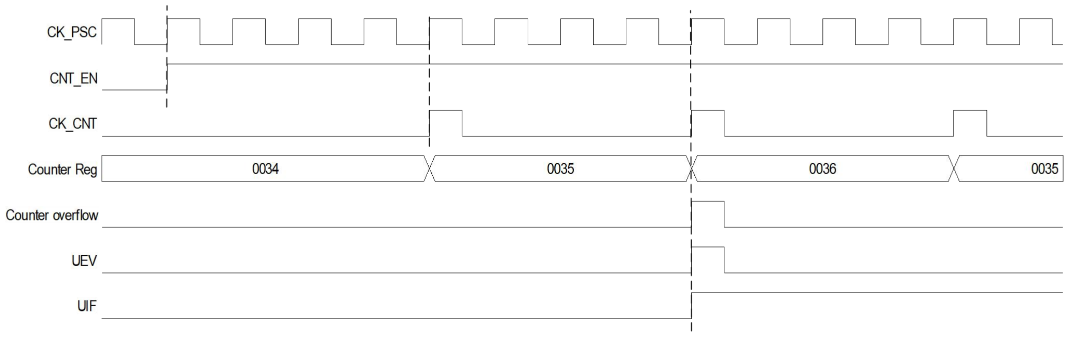


Figure 8.57 Counter timing diagram,

internal clock frequency division factor of 4, TIMx_ARR=0 x36

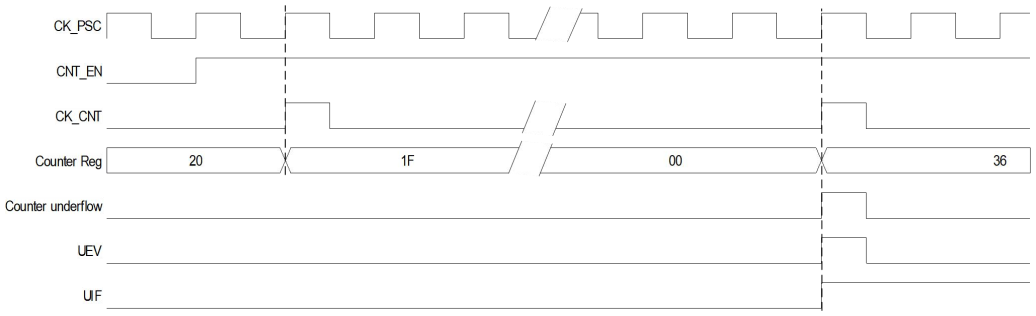


Figure 8.58 Counter timing diagram, internal clock divided by N

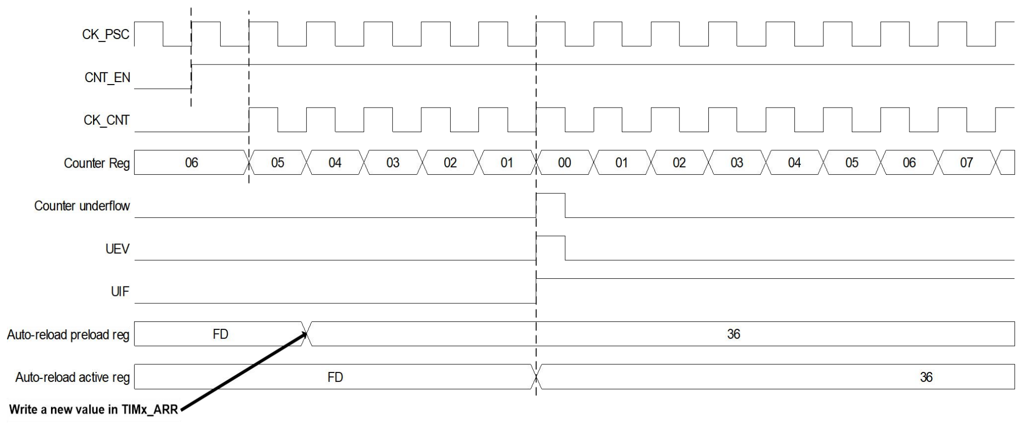


Figure 8.59 Counter timing diagram, update event at ARPE=1(counter underflow)

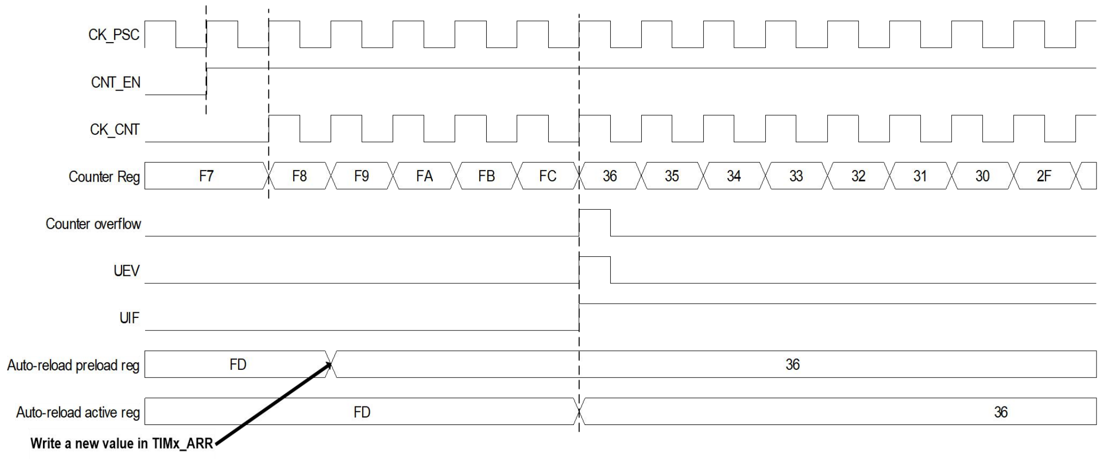


Figure 8.60 Counter timing diagram, update event at ARPE=1(counter overflow)

8.11.3.4.Repetition counter

Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

- At each counter overflow in upcounting mode.
- At each counter underflow in downcounting mode.
- At each counter overflow and at each counter underflow in center-aligned mode.

Although this limits the maximum number of repetition to 128 PWM cycles, it makes it possible to update the duty cycle twice per PWM period.When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is 2xTck, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value. When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In center-aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was started. If the RCR was written before starting the counter, the UEV occurs on the overflow. If the RCR was written after starting the counter, the UEV occurs on the underflow. For example for RCR = 3, the UEV is generated on each 4th overflow or underflow event depending on when RCR was written.

8.11.3.5.Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Internal trigger inputs (ITRx):using one timer as prescaler for another timer,for example, you can configure Timer 1 to act as a prescaler for Timer 2.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

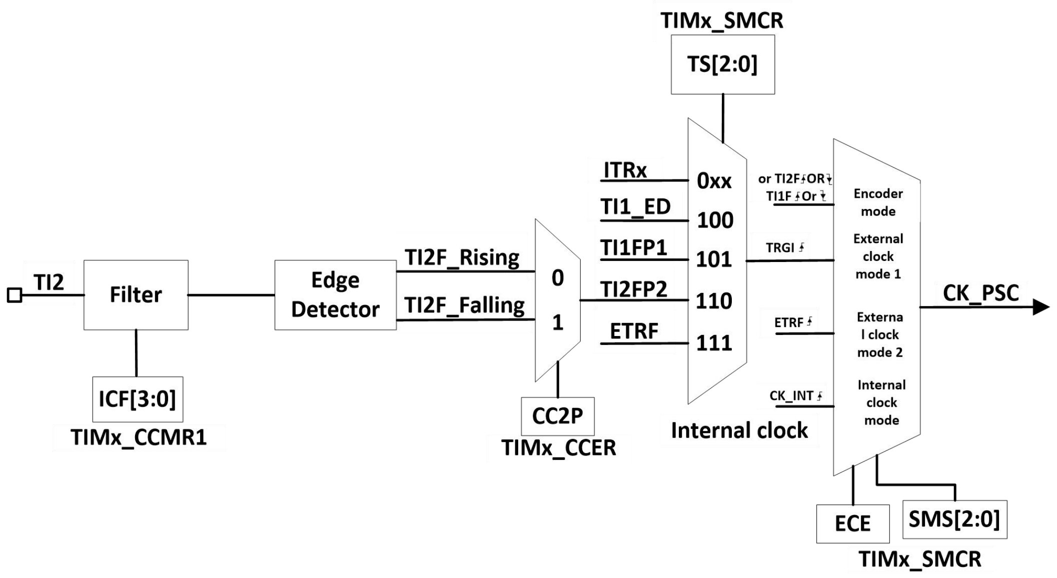


Figure 8.61 TI2 external clock connection example

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

- Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
- Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
- Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
- Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
- Select TI2 as the trigger input source by writing TS=110 in the TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering,so you don’t need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

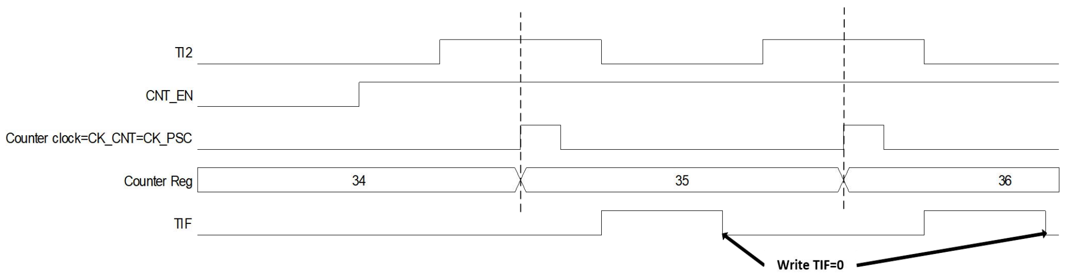


Figure 8.62 Control circuit in external clock mode 1

External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The following figure gives an overview of the external trigger input block.

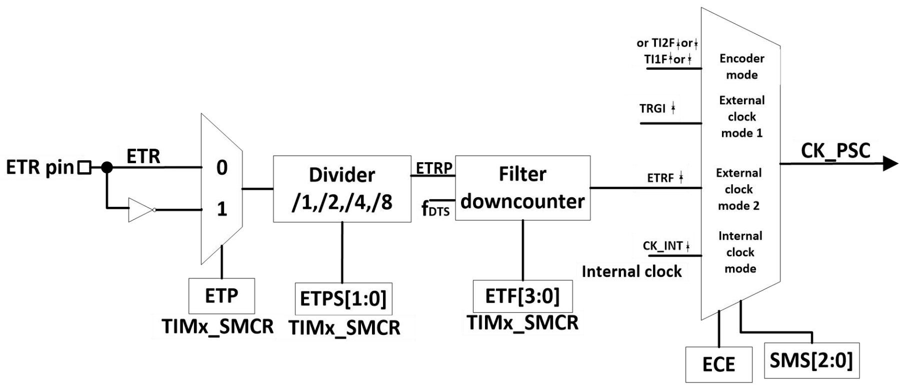


Figure 8.63 External trigger input block

For example,to configure the upcounter to count each 2 rising edges on ETR,use the following procedure:

- As no filter is needed in this example,write ETF[3:0]=0000 in the TIMx_SMCR register.
- Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register.
- Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register.
- Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

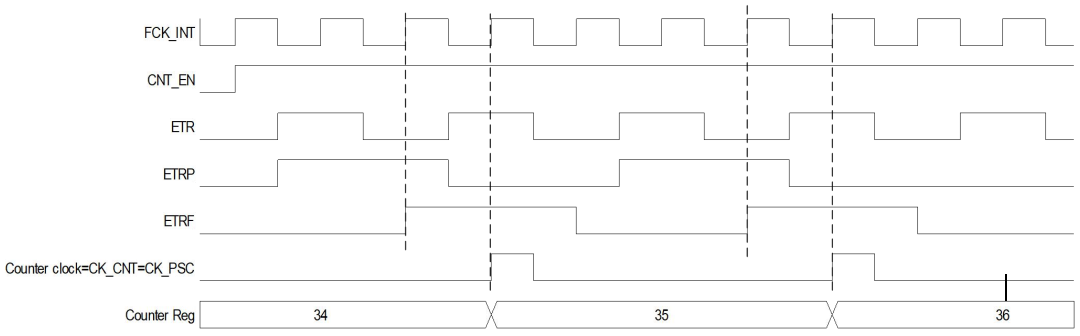


Figure 8.64 Control circuit in external clock mode 2

8.11.3.6.Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 8.65 to Figure 8.68 give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

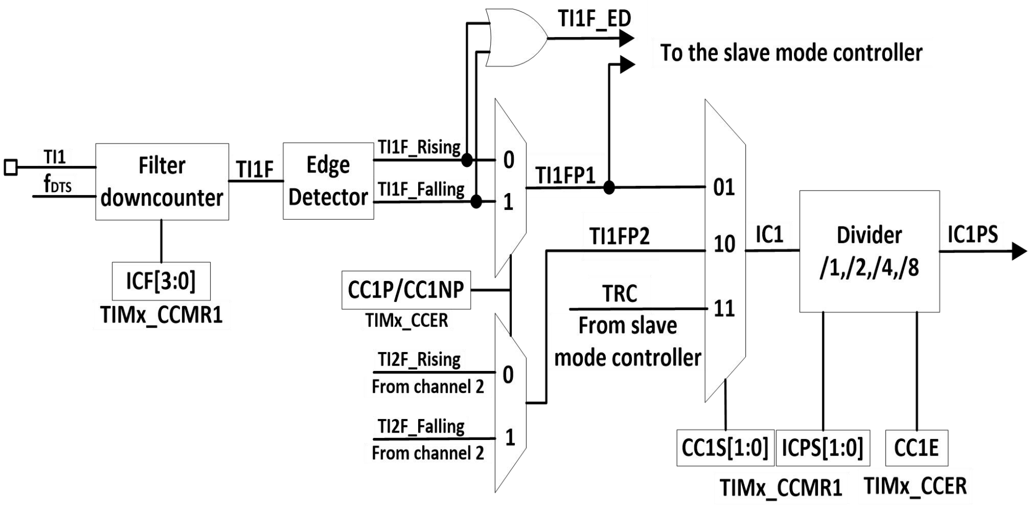


Figure 8.65 Capture/compare channel

(example: channel 1 input stage)

Note: The output part produces an intermediate waveform OCxRef(high efficiency) as the reference, and the end of the chain determines the polarity of the final output signal.

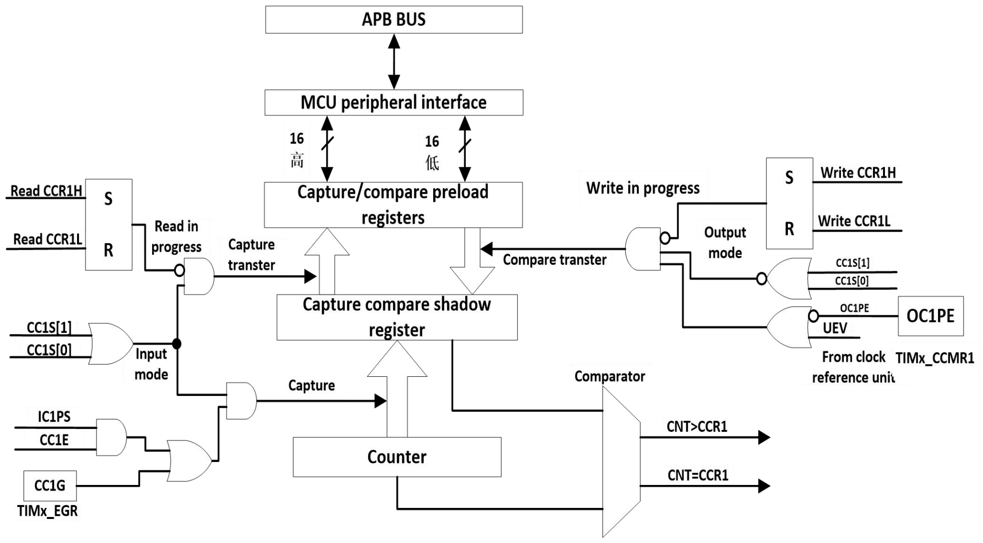


Figure 8.66 Main circuit of capture/compare channel 1

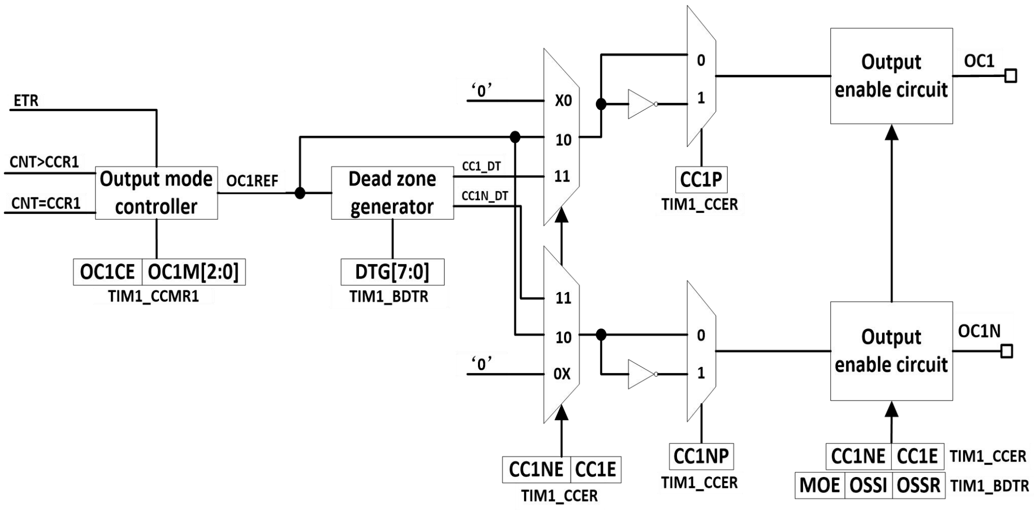


Figure 8.67 Capture/compare the output portion of the channel (channels 1 to 3)

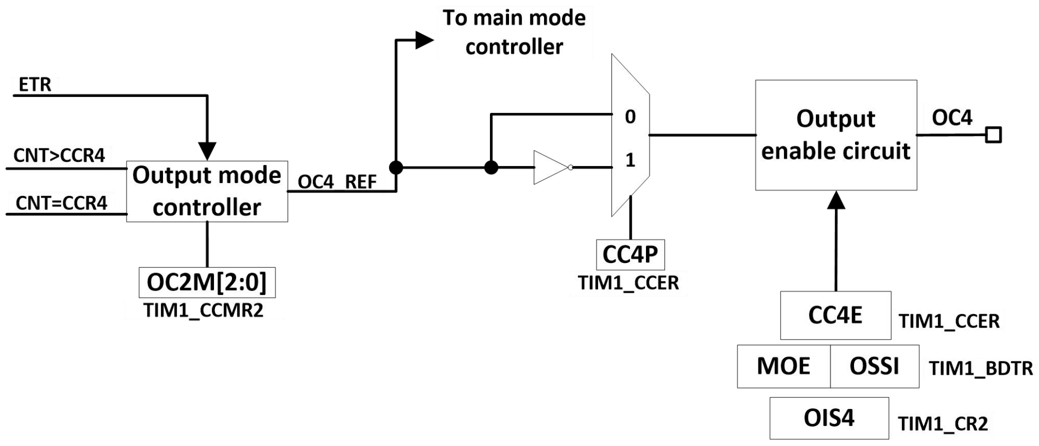


Figure 8.68 Capture/compare the output portion of the channel (channel 4)

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

8.11.3.7.Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to ‘0’ or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to ‘0’.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises.To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the TI1 input,so write the CC1S bits to 01 in the TIMx_CCMR1 register.As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing CC1P and

CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).

- Program the input prescaler.In our example,we wish the capture to be performed at each valid transition,so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.
- When an input capture occurs:
- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag).CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture,it is recommended to read the data before the overcapture flag.This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note:IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

8.11.3.8.PWM input mode

This mode is a particular case of input capture mode.The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

- Select the active input for TIMx_CCR1:write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
- Select the active input for TIMx_CCR2:write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2Pand CC2NP bits to '1' (active on falling edge).
- Select the valid trigger input:write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode:write the SMS bits to 100 in the TIMx_SMCR register.

- Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

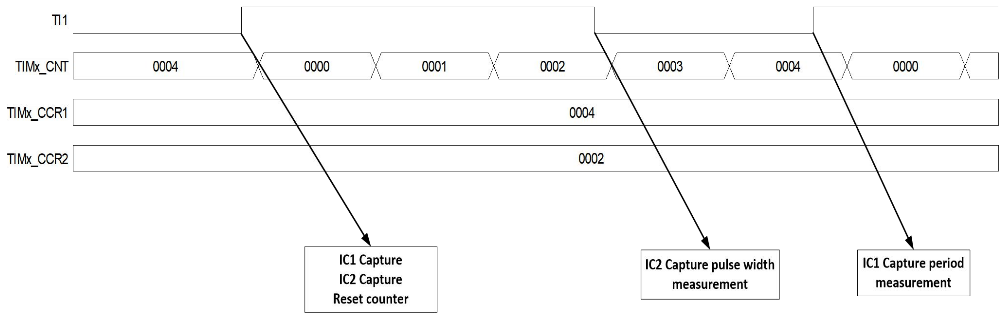


Figure 8.69 PWM Input Mode Timing

8.11.3.9.Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register.Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example:CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set.Interrupt and DMA requests can be sent accordingly.This is described in the output compare mode section below.

8.11.3.10.Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register).The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register,CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure:

- Select the counter clock (internal, external, prescaler).
- Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
- Set the CCxIE bit if an interrupt request is to be generated.
- Select the output mode.For example:
  - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx.
  - Write OCxPE = 0 to disable preload register.
  - Write CCxP = 0 to select active high polarity.
  - Write CCxE = 1 to enable the output.
- Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV).A example is given in the following figure.

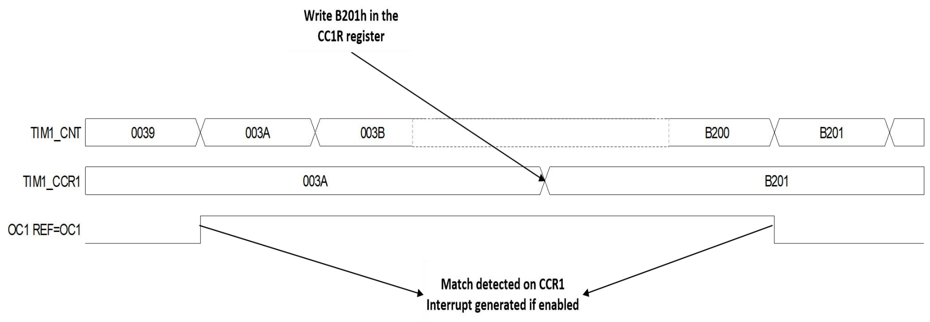


Figure 8.70 Output compare mode, toggle on OC1

8.11.3.11.PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low.OCx output is enabled by a combination of the CCxE,CCxNE, MOE,OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers).Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2),TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

Upcounting configuration:

Upcounting is active when the DIR bit in the TIMx_CR1 register is low.

In the following example,we consider PWM mode 1.The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low.If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’.

The following figure shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

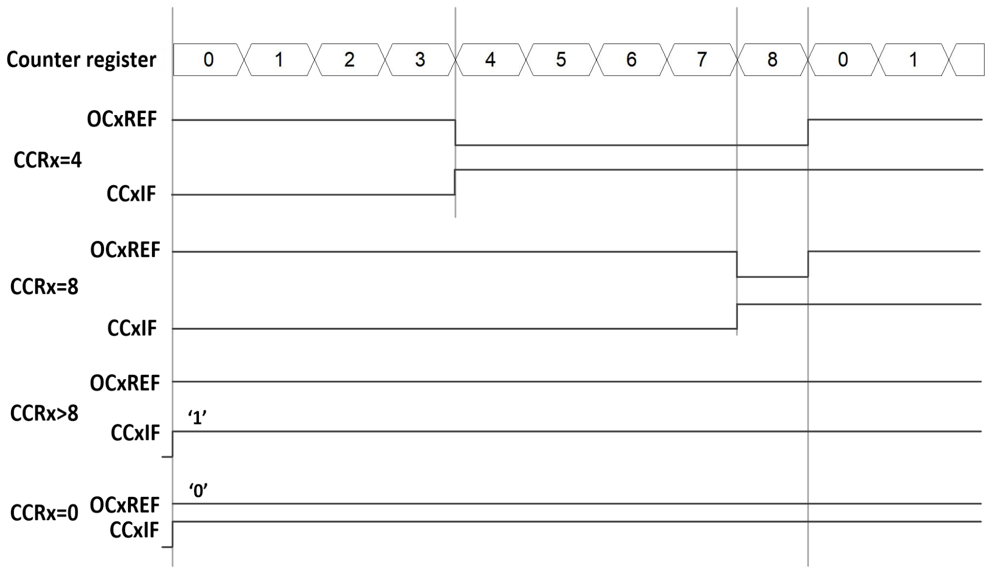


Figure 8.71 Edge-aligned PWM waveforms (ARR=8)

Downcounting configuration:

Downcounting is active when DIR bit in TIMx_CR1 register is high.

In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at ‘1’.0% PWM is not possible in this mode.

PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from '00' (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software.

The following figure shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8.
- PWM mode is the PWM mode 1.
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

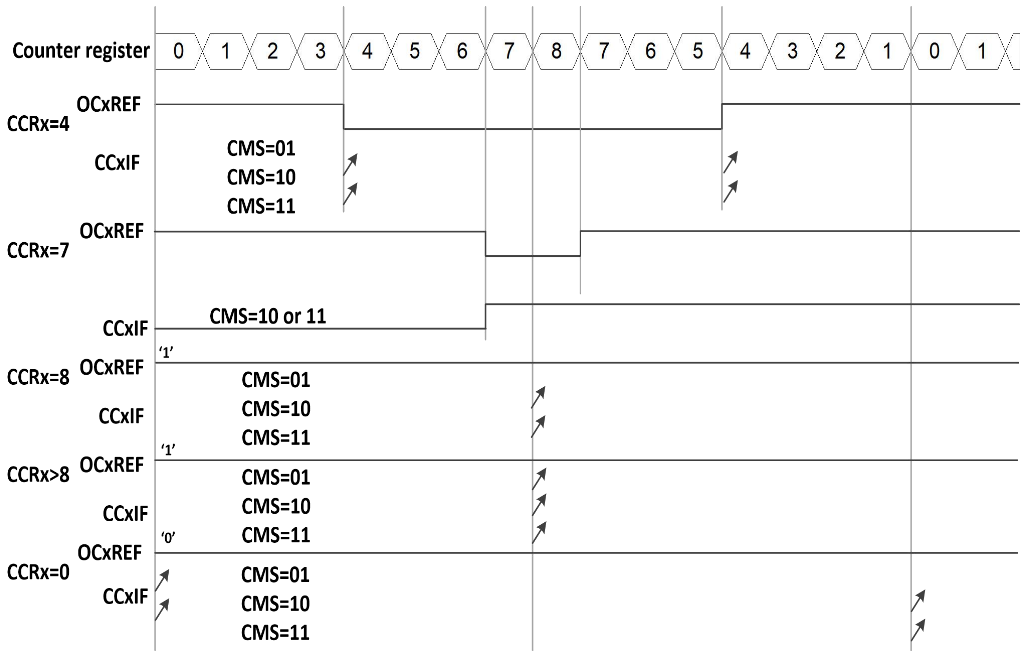


Figure 8.72 Center-aligned PWM waveforms (ARR=8)

Hints on using center-aligned mode

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while run ning in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if you write a value in the counter that is greater than the auto-reload value (TIMx_CNT>TIMx_ARR).For example,if the counter was counting up,it continues to count up.
  - The direction is updated if you write 0 or write the TIMx_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and

not to write the counter while it is running.

8.11.3.12.Complementary outputs and dead-time insertion

The advanced-control timers (TIMER) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and you have to adjust it depending on the devices you have connected to the outputs and their characteristics (intrinsic delays of level- shifters, delays due to power switches...).

You can select the polarity of the outputs (main output OCx or complementary OCxN) independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers.In particular, the dead-time is activated when switching to the IDLE state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF.(we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples).

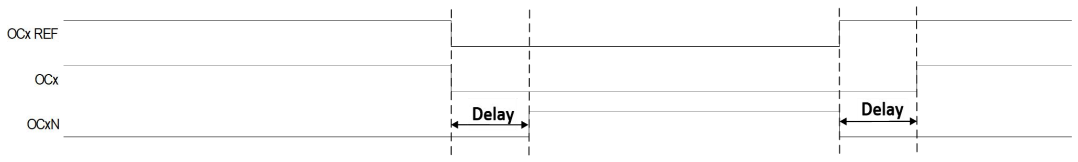


Figure 8.73 Complementary output with dead-time insertion

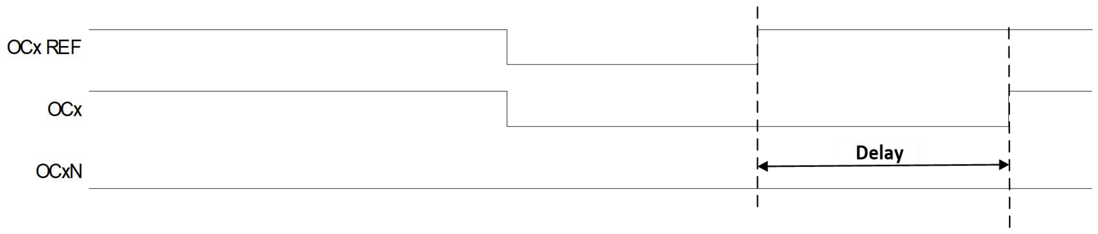


Figure 8.74 Dead-time waveforms with delay greater than the negative pulse

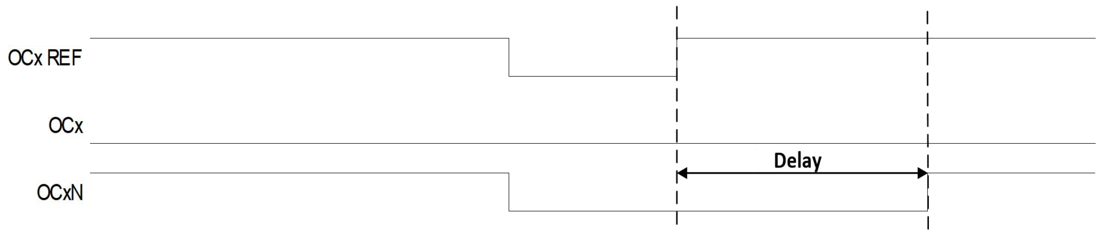


Figure 8.75 Dead-time waveforms with delay greater than the positive pulse

The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows you to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

8.11.3.13.Using the break function

When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSI and OSSR bits in the TIMx_BDTR register, OISx and OISxN bits in the TIMx_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time.

The break source can be either the break input pin or a clock failure event, generated by the Clock Security System (CSS), from the Reset Clock Controller.

When exiting from reset, the break circuit is disabled and the MOE bit is low. You can enable the break function by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register).It results in some delays between the asynchronous and the synchronous signals. In particular, if you write MOE to 1 whereas it was low, you must insert a delay (dummy instruction) before reading it correctly. This is because you write the asynchronous signal and read the synchronous signal.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or in reset state (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0 then the timer releases the enable output else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity).
  - This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time.Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).
  - If OSSI=0 then the timer releases the enable outputs else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until you write it to ‘1’ again. In this case, it can be used for security and you can connect the break input to an alarm from power drivers, thermal sensors or any security components.

Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows you to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). You can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. The LOCK bits can be written only once after an MCU reset.

The following figure shows an example of behavior of the outputs in response to a break.

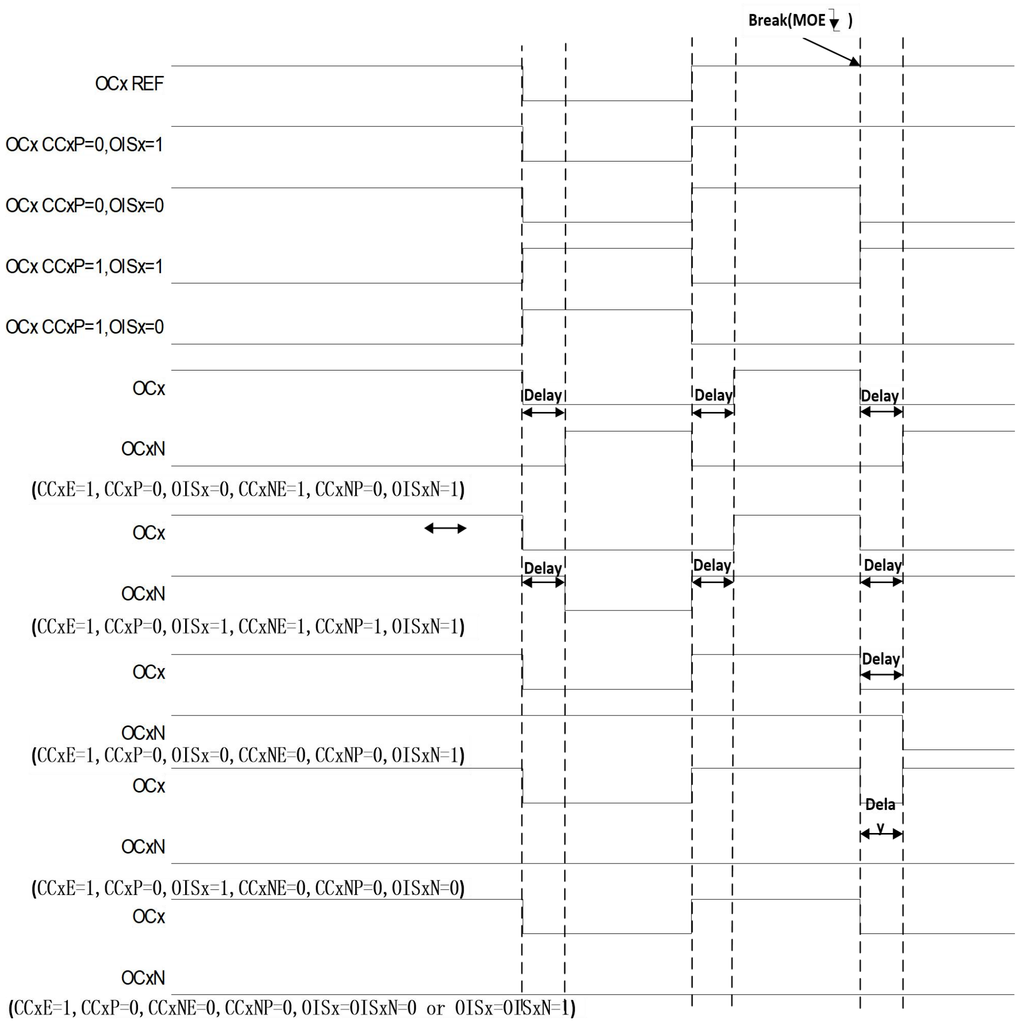

Break(MOE ↓)
OCx REF
OCx CCxP=0,OISx=1
OCx CCxP=0,OISx=0
OCx CCxP=1,OISx=1
OCx CCxP=1,OISx=0
OCx
OCxN
(CCxE=1, CCxP=0, OISx=0, CCxNE=1, CCxNP=0, OISxN=1)
OCx
OCxN
(CCxE=1, CCxP=0, OISx=1, CCxNE=1, CCxNP=1, OISxN=1)
OCx
OCxN
(CCxE=1, CCxP=0, OISx=0, CCxNE=0, CCxNP=0, OISxN=1)
OCx
OCxN
(CCxE=1, CCxP=0, OISx=1, CCxNE=0, CCxNP=0, OISxN=0)
OCx
OCxN
(CCxE=1, CCxP=0, CCxNE=0, CCxNP=0, OISx=OISxN=0 or OISx=OISxN=1)
Delay

Figure 8.76 Output behavior in response to a break

8.11.3.14.Clearing the OCxREF signal on an external event

The OCxREF signal for a given channel can be driven Low by applying a High level to the ETRF input (OCxCE enable bit of the corresponding TIMx_CCMRx register set to ‘1’). The OCxREF signal remains Low until the next update event(UEV)occurs.

This function can only be used in output compare and PWM modes, and does not work in forced mode.

For example, the OCxREF signal) can be connected to the output of a comparator to be used for current handling. In this case, the ETR must be configured as follow:

- The External Trigger Prescaler should be kept off:bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
- The external clock mode 2 must be disabled:bit ECE of the TIMx_SMCR register set to ‘0’.
- The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

The following figure shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

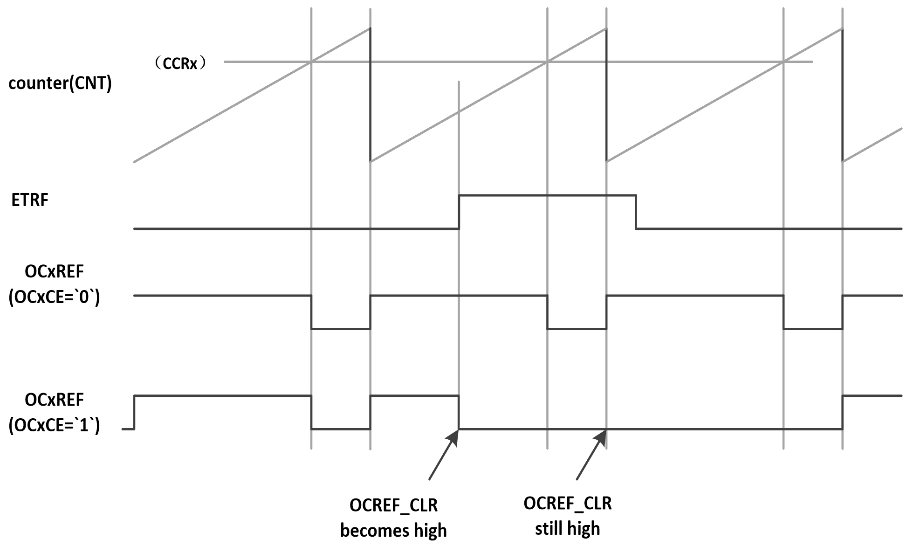


Figure 8.77 Clearing TIMx OCxREF

Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR),then OCxREF is enabled again at the next counter overflow.

8.11.3.15.6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus you can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The following figure describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

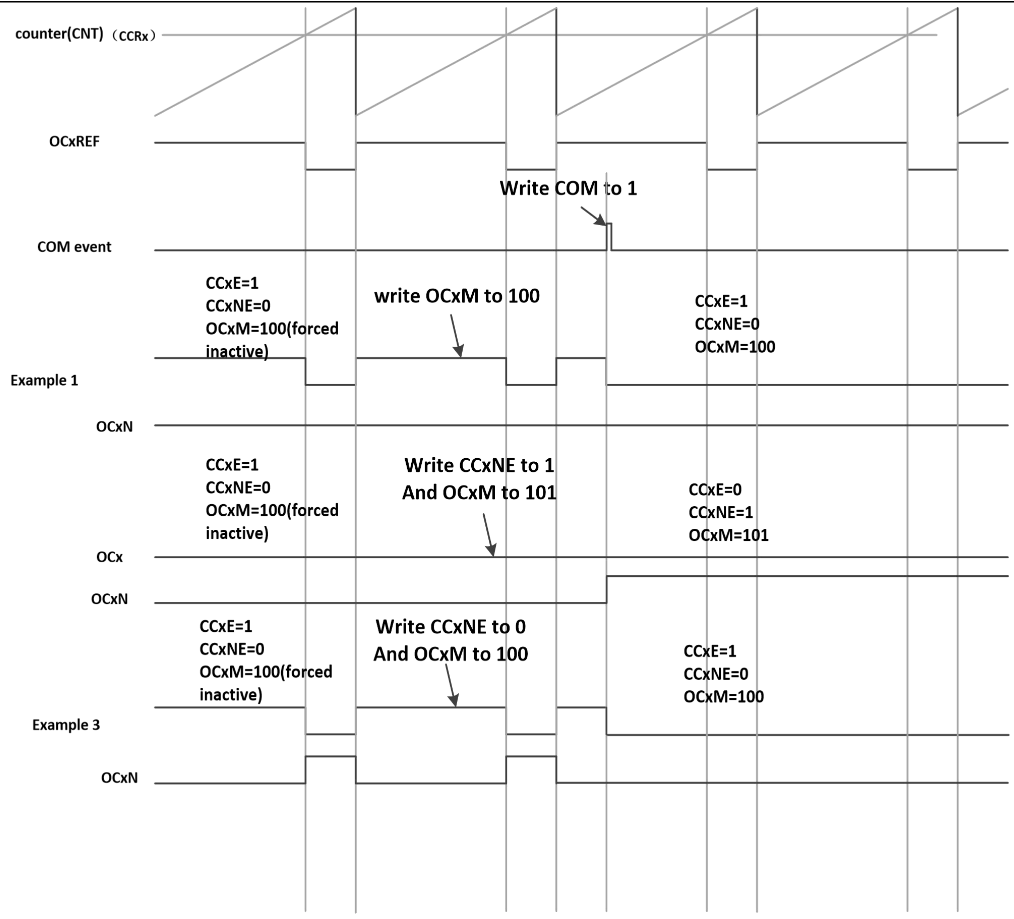


Figure 8.78 step generation, COM example (OSSR=1)

8.11.3.16.One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value.Before starting (when the timer is waiting for the trigger),the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx).
- In downcounting: CNT > CCRx.

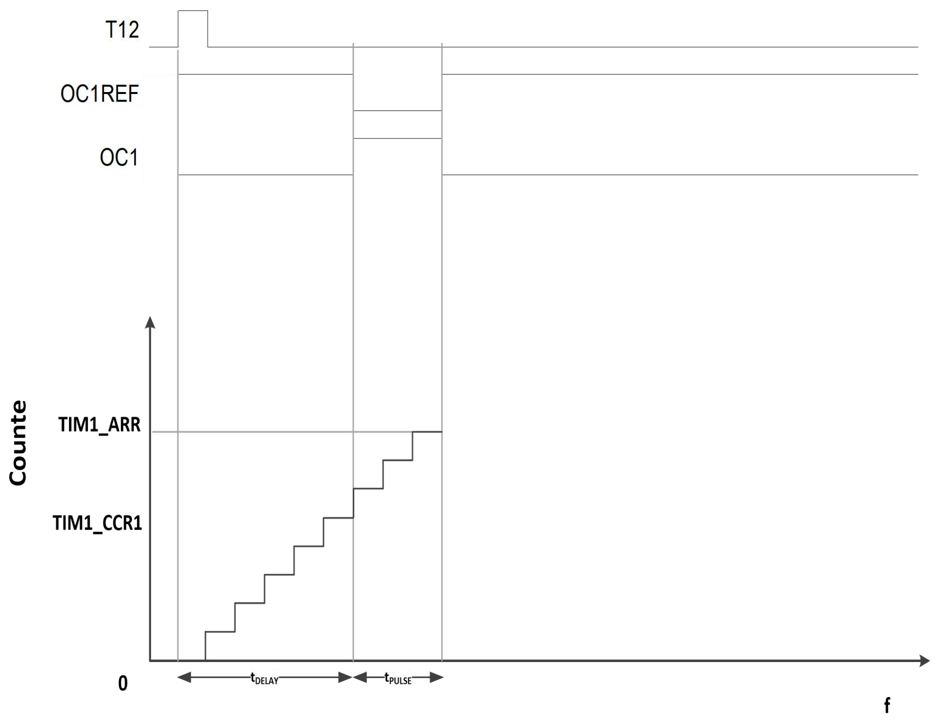


Figure 8.79 Example of one pulse mode

For example you may want to generate a positive pulse on OC1 with a length of tPULSE and after a delay of tDELAY as soon as a positive edge is detected on the TI2 input pin.Let’s use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S=’01’ in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P=’0’ and CC2NP=’0’ in the TIMx_CCERregister.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=’110’ in the TIMx_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to ‘110’ in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The tDELAY is defined by the value written in the TIMx_CCR1 register.
- The tPULSE is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let’s say you want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE=’1’ in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

You only want 1 pulse (Single mode), so you write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay tDELAY min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

8.11.3.17.Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

8.11.3.18.TIMx and external trigger synchronization

The TIMx timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don’t need any filter, so we keep IC1F=0000).The capture prescaler is not used for triggering, so you don’t need to configure it.The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=’0’ in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36.The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

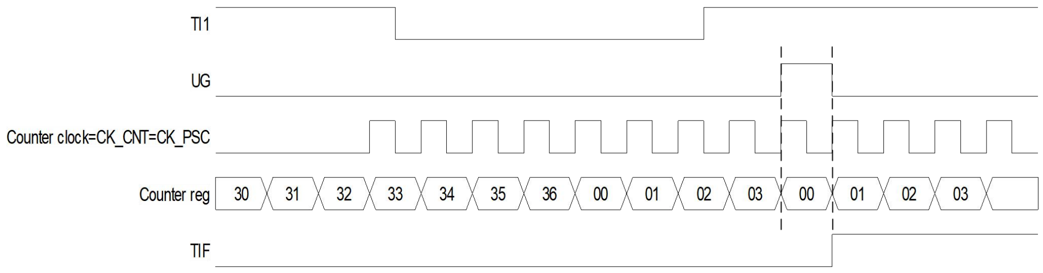


Figure 8.80 Control circuit in reset mode

Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1.Configure the input filter duration (in this example, we don’t need any filter,so we keep IC1F=0000).The capture prescaler is not used for triggering, so you don’t need to configure it.The CC1S bits select the input capture source only,CC1S=01 in TIMx_CCMR1 register.Write CC1P=1 and CC1NP=’0’ in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

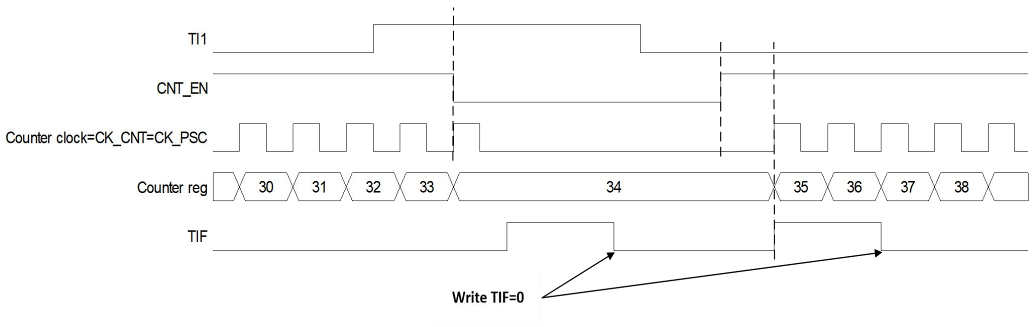


Figure 8.81 Control circuit in gated mode

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2.Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000).The capture prescaler is not used for triggering, so you don't need to configure it.The CC2S bits are configured to select the input capture source only,CC2S=01 in TIMx_CCMR1 register.Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

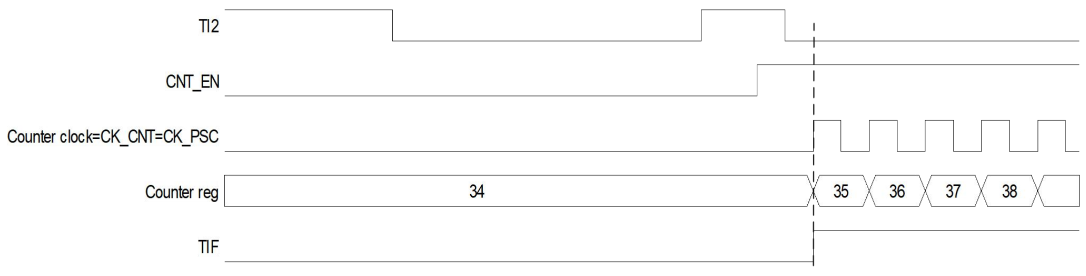


Figure 8.82 Control circuit in trigger mode

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

- Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
  - ETF = 0000: no filter
  - ETPS=00: prescaler disabled
  - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
- Configure the channel 1 as follows, to detect rising edges on TI:
  - IC1F=0000: no filter.

  - The capture prescaler is not used for triggering and does not need to be configured.
  - CC1S=01in TIMx_CCMR1 register to select only the input capture source
  - CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edge only).
- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

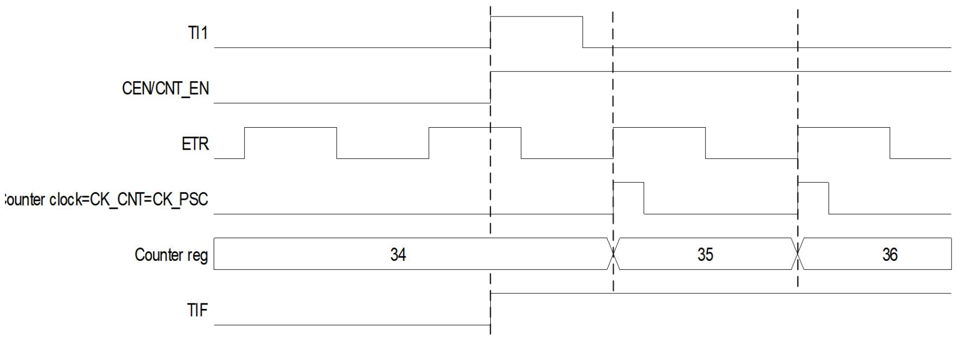


Figure 8.83 Control circuit in external clock mode2 + trigger mode

8.11.4 TIMER Register Map

Offset	Name	Description
0x0000	TIM_CR1	Timer control register 1
0x0004	TIM_CR2	Timer control register 2
0x0008	TIM_SMCR	Timer slave mode control register
0x000c	TIM_DIER	Timer DMA/interrupt enable register
0x0010	TIM_SR	Timer status register
0x0014	TIM_EGR	Timer event generation register
0x0018	TIM_CCMR1	Timer capture/compare mode register1
0x001c	TIM_CCMR2	Timer capture/compare mode register2
0x0020	TIM_CCER	Timer capture/compare enable register
0x0024	TIM_CNT	Timer counter
0x0028	TIM_PSC	Timer prescaler
0x002c	TIM_ARR	Timer auto-reload register
0x0030	TIM_RCR	Timer reperepetition counter register
0x0034	TIM_CCR1	Timer capture/compare register1
0x0038	TIM_CCR2	Timer capture/compare register2
0x003c	TIM_CCR3	Timer capture/compare register3

0x0040	TIM_CCR4	Timer capture/compare register4
0x0044	TIM_BDTR	Timer break and dead-time register
0x0048	TIM_DCR	Timer dma control register
0x004c	TIM_DMAR	Timer dma address for full transfer

TIM_CR1 address offset: 0x0000

Bit	R/W	Reset	Name	Description
31:10	N/A	0x0	N/A	reserved
9:8	RW	0x0	CKD	Clock division This bit-field indicates the division ratio between the timer clock (tCK_INT) frequency and the dead-time and sampling clock (tDTS)used by the dead-time generators and the digital filters
7	RW	0x0	ARPE	Auto-reload preload enable 0: TIM_ARR register is not buffered 1: TIM_ARR register is buffered
6:5	RW	0x0	CMS	Center-aligned mode selection 00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR). 01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels are set only when the counter is counting down. 10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels are set only when the counter is counting up. 11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels are set both when the counter is counting up or down.
4	RW	0x0	DIR	Direction 0: Counter used as upcounter 1: Counter used as downcounter
3	RW	0x0	OPM	One pulse mode 0: Counter is not stopped at update event 1: Counter stops counting at the next update event (clearing the bit CEN)
2	RW	0x0	URS	Update request source This bit is set and cleared by software to select the UEV event sources.
1	RW	0x0	UDIS	Update disable, This bit is set and cleared by software to enable/disable UEV event generation.

0	RW	0x0	CEN	Counter enable 0: Counter disabled 1: Counter enabled

TIM_CR2 address offset: 0x0004

Bit	R/W	Reset	Name	Description
31:15	N/A	0x0	N/A	reserved
14	RW	0x0	OIS4	Output Idle state 4 (OC4 output) 0:OC4=0 (after a dead-time if OC4N is implemented) when MOE=0 1:OC4=1 (after a dead-time if OC4N is implemented) when MOE=0
13	RW	0x0	OIS3N	Output Idle state 3 (OC3N output) 0: OC3N=0 after a dead-time when MOE=0 1: OC3N=1 after a dead-time when MOE=0
12	RW	0x0	OIS3	Output Idle state 3 (OC3 output) 0: OC3=0 (after a dead-time if OC3N is implemented) when MOE=0 1: OC3=1 (after a dead-time if OC3N is implemented) when MOE=0
11	RW	0x0	OIS2N	Output Idle state 2 (OC2N output) 0: OC2N=0 after a dead-time when MOE=0 1: OC2N=1 after a dead-time when MOE=0
10	RW	0x0	OIS2	Output Idle state 2 (OC2 output) 0: OC2=0 (after a dead-time if OC2N is implemented) when MOE=0 1: OC2=1 (after a dead-time if OC2N is implemented) when MOE=0
9	RW	0x0	OIS1N	Output Idle state 1 (OC1N output) 0: OC1N=0 after a dead-time when MOE=0 1: OC1N=1 after a dead-time when MOE=0
8	RW	0x0	OIS1	Output Idle state 1 (OC1 output) 0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0 1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0
7	RW	0x0	TI1S	TI1 selection 0: The TIM_CH1 pin is connected to TI1 input 1: The TIM_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)
6:4	RW	0x0	MMS	Master mode selection
3	RW	0x0	CCDS	Capture/compare DMA selection 0: CC DMA request sent when CC event occurs 1: CC DMA requests sent when update event

				occurs
2	RW	0x0	CCUS	Capture/compare control update selection 0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only 1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI
1	N/A	0x0	N/A	reserved
0	RW	0x0	CCPC	Capture/compare preloaded control 0: CCE, CCNE and OCM bits are not preloaded 1: CCE, CCNE and OCM bits are preloaded

TIM_SMCR address offset: 0x0008

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15	RW	0x0	ETP	External trigger polarity This bit selects whether ETR or ETR is used for trigger operations 0: ETR is non-inverted, active at high level or rising edge. 1: ETR is inverted, active at low level or falling edge.
14	RW	0x0	ECE	External clock enable This bit enables External clock mode 2. 0: External clock mode 2 disabled 1: External clock mode 2 enabled.
13:12	RW	0x0	ETPS	External trigger prescaler
11:8	RW	0x0	ETF	External trigger filter
7	RW	0x0	MSM	Master/slave mode
6:4	RW	0x0	TS	Trigger selection
3	N/A	0x0	N/A	reserved
2:0	RW	0x0	SMS	Slave mode selection

TIM_DIER address offset: 0x000c

Bit	R/W	Reset	Name	Description
31:15	N/A	0x0	N/A	reserved
14	RW	0x0	TDE	Trigger DMA request enable 0: Trigger DMA request disabled 1: Trigger DMA request enabled
13	RW	0x0	COMDE	COM DMA request enable 0: COM DMA request disabled 1: COM DMA request enabled

12	RW	0x0	CC4DE	Capture/Compare 4 DMA request enable 0: CC4 DMA request disabled 1: CC4 DMA request enabled
11	RW	0x0	CC3DE	Capture/Compare 3 DMA request enable 0: CC3 DMA request disabled 1: CC3 DMA request enabled
10	RW	0x0	CC2DE	Capture/Compare 2 DMA request enable 0: CC2 DMA request disabled 1: CC2 DMA request enabled
9	RW	0x0	CC1DE	Capture/Compare 1 DMA request enable 0: CC1 DMA request disabled 1: CC1 DMA request enabled
8	RW	0x0	UDE	Update DMA request enable 0: Update DMA request disabled 1: Update DMA request enabled
7	RW	0x0	BIE	Break interrupt enable 0: Break interrupt disabled 1: Break interrupt enabled
6	RW	0x0	TIE	Trigger interrupt enable 0: Trigger interrupt disabled 1: Trigger interrupt enabled
5	RW	0x0	COMIE	COM interrupt enable 0: COM interrupt disabled 1: COM interrupt enabled
4	RW	0x0	CC4IE	Capture/Compare 4 interrupt enable 0: CC4 interrupt disabled 1: CC4 interrupt enabled
3	RW	0x0	CC3IE	Capture/Compare 3 interrupt enable 0: CC3 interrupt disabled 1: CC3 interrupt enabled
2	RW	0x0	CC2IE	Capture/Compare 2 interrupt enable 0: CC2 interrupt disabled 1: CC2 interrupt enabled
1	RW	0x0	CC1IE	Capture/Compare 1 interrupt enable 0: CC1 interrupt disabled 1: CC1 interrupt enabled
0	RW	0x0	UIE	Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

TIM_SR address offset: 0x0010

Bit	R/W	Reset	Name	Description
31:13	N/A	0x0	N/A	reserved
12	RW	0x0	CC4OF	Capture/Compare 4 overcapture flag

11	RW	0x0	CC3OF	Capture/Compare 3 overcapture flag
10	RW	0x0	CC2OF	Capture/Compare 2 overcapture flag
9	RW	0x0	CC1OF	Capture/Compare 1 overcapture flag
8	N/A	0x0	N/A	reserved
7	RW	0x0	BIF	Break interrupt flag
6	RW	0x0	TIF	Trigger interrupt flag
5	RW	0x0	COMIF	COM interrupt flag
4	RW	0x0	CC4IF	Capture/Compare 4 interrupt flag
3	RW	0x0	CC3IF	Capture/Compare 3 interrupt flag
2	RW	0x0	CC2IF	Capture/Compare 2 interrupt flag
1	RW	0x0	CC1IF	Capture/Compare 1 interrupt flag
0	RW	0x0	UIF	Update interrupt flag This bit is set by hardware on an update event. It is cleared by software. 0: No update occurred. 1: Update interrupt pending.

TIM_EGR address offset: 0x0014

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7	W	0x0	BG	Break generation
6	W	0x0	TG	Trigger generation
5	W	0x0	COMG	Capture/Compare control update generation
4	W	0x0	CC4G	Capture/Compare 4 generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: A capture/compare event is generated on channel 4:
3	W	0x0	CC3G	Capture/Compare 3 generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: A capture/compare event is generated on channel 3:
2	W	0x0	CC2G	Capture/Compare 2 generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: A capture/compare event is generated on channel 2:
1	W	0x0	CC1G	Capture/Compare 1 generation This bit is set by software in order to generate an

				event, it is automatically cleared by hardware. 0: No action 1: A capture/compare event is generated on channel 1:
0	W	0x0	UG	Update generation This bit can be set by software, it is automatically cleared by hardware. 0: No action 1: Reinitialize the counter and generates an update of the registers.

TIM_CCMR1 address offset: 0x0018 (Output compare mode)

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15	RW	0x0	OC2CE	Output Compare 2 clear enable
14:12	RW	0x0	OC2M	Output Compare 2 mode
11	RW	0x0	OC2PE	Output Compare 2 preload enable
10	RW	0x0	OC2FE	Output Compare 2 fast enable
9:8	RW	0x0	CC2S	Capture/Compare 2 Selection
7	RW	0x0	OC1CE	Output Compare 1 clear enable
6:4	RW	0x0	OC1M	Output Compare 1 mode
3	RW	0x0	OC1PE	Output Compare 1 preload enable
2	RW	0x0	OC1FE	Output Compare 1 fast enable
1:0	RW	0x0	CC1S	Capture/Compare 1 Selection

TIM_CCMR1 address offset: 0x0018 (Input capture mode)

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:12	RW	0x0	IC2F	Input capture 2 filter
11:10	RW	0x0	IC2PSC	Input capture 2 prescaler
9:8	RW	0x0	CC2S	Capture/Compare 2 Selection
7:4	RW	0x0	IC1F	Input capture 1 filter
3:2	RW	0x0	IC1PSC	Input capture 1 prescaler
1:0	RW	0x0	CC1S	Capture/Compare 1 Selection

TIM_CCMR2 address offset: 0x001c (Output compare mode)

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15	RW	0x0	OC4CE	Output Compare 4 clear enable
14:12	RW	0x0	OC4M	Output Compare 4 mode
11	RW	0x0	OC4PE	Output Compare 4 preload enable
10	RW	0x0	OC4FE	Output Compare 4 fast enable
9:8	RW	0x0	CC4S	Capture/Compare 4 Selection

7	RW	0x0	OC3CE	Output Compare 3 clear enable
6:4	RW	0x0	OC3M	Output Compare 3 mode
3	RW	0x0	OC3PE	Output Compare 3 preload enable
2	RW	0x0	OC3FE	Output Compare 3 fast enable
1:0	RW	0x0	CC3S	Capture/Compare 3 Selection

TIM_CCMR2 address offset: 0x001c (Input capture mode)

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:12	RW	0x0	IC4F	Input capture 4 filter
11:10	RW	0x0	IC4PSC	Input capture 4 prescaler
9:8	RW	0x0	CC4S	Capture/Compare 4 Selection
7:4	RW	0x0	IC3F	Input capture 3 filter
3:2	RW	0x0	IC3PSC	Input capture 3 prescaler
1:0	RW	0x0	CC3S	Capture/Compare 3 Selection

TIM_CCER address offset: 0x0020

Bit	R/W	Reset	Name	Description
31:14	N/A	0x0	N/A	reserved
13	RW	0x0	CC4P	Capture/Compare 4 output polarity
12	RW	0x0	CC4E	Capture/Compare 4 output enable
11	RW	0x0	CC3NP	Capture/Compare 3 complementary output polarity
10	RW	0x0	CC3NE	Capture/Compare 3 complementary output enable
9	RW	0x0	CC3P	Capture/Compare 3 output polarity
8	RW	0x0	CC3E	Capture/Compare 3 output enable
7	RW	0x0	CC2NP	Capture/Compare 2 complementary output polarity
6	RW	0x0	CC2NE	Capture/Compare 2 complementary output enable
5	RW	0x0	CC2P	Capture/Compare 2 output polarity
4	RW	0x0	CC2E	Capture/Compare 2 output enable
3	RW	0x0	CC1NP	Capture/Compare 1 complementary output polarity
2	RW	0x0	CC1NE	Capture/Compare 1 complementary output enable
1	RW	0x0	CC1P	Capture/Compare 1 output polarity
0	RW	0x0	CC1E	Capture/Compare 1 output enable

TIM_CNT address offset: 0x0024

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved

15:0	RW	0x0	CNT	Counter value

TIM_PSC address offset: 0x0028

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x0	PSC	Prescaler value

TIM_ARR address offset: 0x002c

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x0	ARR	Prescaler value ARR is the value to be loaded in the actual auto-reload register.

TIM_RCR address offset: 0x0030

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	RW	0x0	REP	Repetition counter value

TIM_CCR1 address offset: 0x0034

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x0	CCR1	Capture/Compare 1 value

TIM_CCR2 address offset: 0x0038

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x0	CCR2	Capture/Compare 2 value

TIM_CCR3 address offset: 0x003c

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x0	CCR3	Capture/Compare 3 value

TIM_CCR4 address offset: 0x0040

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x0	CCR4	Capture/Compare 4 value

TIM_BDTR address offset: 0x0044

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15	RW	0x0	MOE	Main output enable

14	RW	0x0	AOE	Automatic output enable
13	RW	0x0	BKP	Break polarity
12	RW	0x0	BKE	Break enable
11	RW	0x0	OSSR	Off-state selection for Run mode
10	RW	0x0	OSSI	Off-state selection for Idle mode
9:8	RW	0x0	LOCK	Lock configuration
7:0	RW	0x0	DTG	Dead-time generator setup

TIM_DCR address offset: 0x0048

Bit	R/W	Reset	Name	Description
31:13	N/A	0x0	N/A	reserved
12:8	RW	0x0	DBL	DMA burst length
7:5	N/A	0x0	N/A	reserved
4:0	RW	0x0	DBA	DMA base address

TIM_DMAR address offset: 0x004c

Bit	R/W	Reset	Name	Description
31:16	N/A	0x0	N/A	reserved
15:0	RW	0x0	DMAB	DMA register for burst accesses

8.12 RTC

8.12.1 Introduction

The Real Time Calendar (RTC) works as a generic 32-bit low power timer running on a low frequency clock. It provides an alarm signal to either wake up the chip or interrupt the MCU. It also generates an interrupt when it wraps around.

Since it is in AON domain, the timer can keep running when the MCU is off.

8.12.2 Main features

- Embedded 32768Hz oscillator for 32k clock generation with an external 32k crystal.
- RTCLK selectable from the oscillator or from the divided clock of EXCLK, so that 32k crystal can be absent if the hibernating mode is not needed.
- 32-bits second counter.
- Programmable and adjustable counter to generate accurate 1 Hz clock.
- Alarm interrupt, 1Hz interrupt.
- Stand alone power supply, work in hibernating mode.
- Power down controller.
- Alarm wakeup.
- External pin wakeup with up to 2s glitch filter.

8.12.3 Function Description

8.12.3.1.Block Diagram

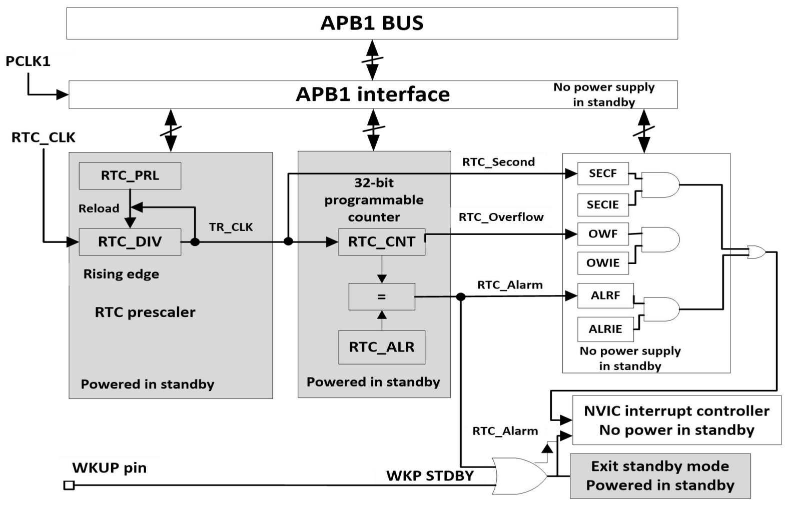


Figure 8.84 RTC block diagram

8.12.3.2.Clocks

There could be two clock input to RTC internal clock called RTCLK.One is OSC32k clock;the other is EXCLK/512.

The software MUST make sure the RTC run in valid clock configuration.

RTCCR.SELEXC	CPM.ERCS	Description	Valid
0	0	RTC use OSC32K clock	OK
0	1		OK
1	0	RTC use EXCLK/512 clock	OK
1	1	RTC will lost clock(Not Valid)	NO

Table 8.8 Clock select registers

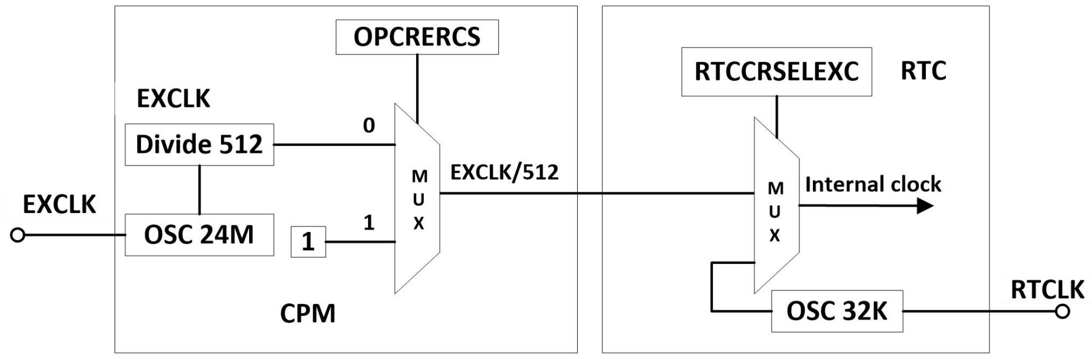


Figure 8.85 RTC clock selection path

Changing RTCLK sequence:

- There are both 32KHz crystal and 24Mhz EXCLK crystal connected, so RTCLK input path has 32Khz clock.

In this case, there is no need to change internal clock, so do NOT change SELEXC all the time.

- There is no 32KHz crystal connected but only 24Mhz EXCLK crystal connected, so RTCLK input path has no clock.

In this case, should flow the sequence below to change internal clock:

- Set OPCR.ERCS of CPM to 1; close EXCLK/512 to RTC;
- Set CLKGR.RTC of CPM to 1; close PCLK to RTC;
- Set RTCCR.SELEXC to 1; change internal clock to EXCLK/512;
- Wait two clock period of clock;
- Clear OPCR.ERCS of CPM to 0; open EXCLK/512 to RTC;
- Clear CLKGR.RTC of CPM to 0; open PCLK to RTC;
- Configure all RTC registers but RTCCR.SELEXC;
- Check RTCCR.SELEXC == 1;
- IF YES, finish this sequence; IF NO, do step (1) again.

Note: If using HIBERNATE mode,MUST have both 32KHz crystal (or input 32Khz clock) and 24Mhz EXCLK crystal connected, or RTC time will be insignificant.

8.12.3.4.Counter load

- To load value to up-counter,program desired value to VAL then set write-only register **VAL_LOAD = 1**.It takes effect in 1~3 configured clock cycles as well. MCU can check readonly register VAL_RD to confirm (optional).
- To load value to alarm register,program desired value to VAL then set write-only register **ALARM_VAL_LOAD = 1**.It takes effect in 1~3 configured clock cycles as well. MCU can check read-only register VAL_RD to confirm (optional).

8.12.3.5.Timer Enable and Disable

- To Enable the clock,set EN = 1;
- To Disable the clock,set EN = 0.

Note: Before enabling the block,make sure the value loads successfully.

8.12.3.6.Counter

The up-counter starts to count up once the block is enabled. WRAP_CNT increments when the counter wraps around. An interrupt is generated as well.

8.12.3.7.Alarm

The calendar timer generates an alarm interrupt when the up-counter equal the value of alarm register.

8.12.3.8.Reading Timer Value

Set TIMER_READ_SEL with its values in the table below,and read the corresponding values of the counter.

Value	Option
0	Calendar timer up-counter value
1	Always on watchdog down-counter value
2	Sleep timer counter value
3	Calendar timer alarm value

Table 8.9 TIMER_READ_SEL Description

8.12.4 RTC Register Map

Offset	Name	Description
0x0000	RTCCR	RTC control register

0x0004	RTCSR	RTC second register
0x0008	RTCSAR	RTC second alarm register
0x000c	RTCGR	RTC regulator register
0x0010	RTC_SR	RTC one second counter register
0x0014	RTCWDT_CR	RTC watchdog control register
0x0018	RTCWDT_SR	RTC watchdog second register

RTCCR address offset: 0x0000

Bits	RW	Reset	Name	Description
31:20	N/A	0x0	N/A	reserved
19	R	0x0	rtc_aie_sync	The sync bit of rtc_aie
18	R	0x0	rtc_ae_sync	The sync bit of rtc_ae
17	N/A	0x0	N/A	reserved
16	R	0x0	rtc_en_sync	The sync bit of bit0
15:10	N/A	0x0	N/A	reserved
9	R	0x0	rtc_1hz_wake	1hz function wakeup flag
8	R	0x0	rtc_af_wake	alarm function wakeup flag
7	R	0x0	rtcsr_ini_sync	RTCSR register update flag
6	RW	0x0	rtc_1hz_clr	1Hz function interrupt clear, write 1 clear Read this bit to get clear status, self-clear
5	RW	0x0	rtc_1hzie	1hz function interrupt enable
4	RW	0x0	rtc_af_clr	alarm function interrupt clear, write 1 clear Read this bit to get clear status, self-clear
3	RW	0x0	rtc_aie	alarm function interrupt enable
2	RW	0x0	rtc_ae	alarm function enable
1	N/A	0x0	N/A	reserved
0	RW	0x0	rtcce	rtc enable

RTCSR address offset: 0x0004

Bits	RW	Reset	Name	Description
31:0	RW	0x0	rtcsr	Second counter register

RTCSAR address offset: 0x0008

Bits	RW	Reset	Name	Description
31:0	RW	0x0	rtcsar	Second alarm register

RTCGR address offset: 0x000c

Bits	RW	Reset	Name	Description
31	RW	0x0	lock	0:the value of RTCGR[25:0] take effect, 1:the value of RTCGR[25:0] don’t take effect
30	R	0x0	rtc_lock_sync	The sync bit of bit31
29:26	N/A	0x0	N/A	reserved
25:16	RW	0x0	adjc	The adjusment cycle number of each 1024

				second
15:0	RW	0x7fff	nc1hz	The cycle number of one second

RTC1HZ address offset: 0x0010

Bits	RW	Reset	Name	Description
31:10	N/A	0x0	N/A	reserved
9:0	RW	0x3ff	rtc1hz_reg	one second counter

RTCWDT address offset: 0x0014

Bits	RW	Reset	Name	Description
31:6	N/A	0x0	N/A	reserved
5	R	0x0	rtc_wdt_intr	RTC WDT interrupt status
4	R	0x0	rtc_wdt_sync	The sync bit of rtc_wdt
3	RW	0x0	rtc_wdt_upd	Write 1 to update RTCWDT settings Read this bit to get update status
2	RW	0x0	rtc_wdt_intr_clr	RTC WDT interrupt clear, write 1 clear Read this bit to get clear status, self-clear
1	RW	0x0	wdt_intr_en	RTC WDT interrupt enable 1: enable 0: disable
0	RW	0x1	rtc_wdt	RTC WDT enable 1: enable 0: disable

RTCWDT_TRIG address offset: 0x0018

Bits	RW	Reset	Name	Description
31:0	RW	0x100000	rtc_wdt_trig	The trigger value for RTC watchdog

9 Communication Subsystem

9.1 Supported Features

HS6621Cx on-chip Bluetooth system compliant with version 5.1, support all of Bluetooth standard 5.1 feature.

9.2 Radio Transceiver

The Radio Transceiver implements the RF part of the Bluetooth Low Energy protocol. Together with the Bluetooth 5.1 PHY layer, this provides a reliable wireless communication. All RF blocks are supplied by on-chip low-drop out-regulators (LDO's). The Bluetooth LE radio comprises the Receiver, Transmitter, Synthesizer, Rx/Tx combiner block, and Biasing LDO's.

9.2.1 Bluetooth Radio Receiver

The HS6621Cx receiver is a low IF down conversion architecture.The RF signal passes first through an integrated transformer, which is shared between receiver and transmitter. The transformer drives a differential variable-gain LNA, which amplifies the signal before it passes through a low-IF down conversion mixer stage. Following the mixer is a third-order complex BPF, which performs channel selection and image rejection. The IF signal is then digitized by two noise-shaping SAR ADCs before further signal processing in the digital domain.

9.2.2 Bluetooth Radio Transmitter

The HS6621Cx transmitter is a direct modulating architecture. The digital base band signals directly modulate VCO and divider of PLL, which is called two-point modulation. After a 3-stage B-class power amplifier, the radio signal is output through antenna.

9.2.3 Frequency Synthesizer

The HS6621Cx Frequency synthesizer is fully integrated sigma delta fractional-N PLL to lock the VCO to a reference crystal oscillator. The synthesizer uses a number of integrated linear regulators for better isolation to the blocks respectively.

9.3 Bluetooth Base band Unit

The BLE (Bluetooth Low Energy) core is a qualified Bluetooth 5.1 base-band controller compatible with Bluetooth Smart specification and it is in charge of packet encoding/decoding and frame scheduling.

9.3.1 Main Features

- All device classes support (Broadcaster, Central, Observer, Peripheral).
- All packet types (Advertising / Data / Control).
- Encryption (AES / CCM).
- Bit stream processing (CRC, Whitening).
- Frequency Hopping calculation.
- Low power modes supporting 32.768kHz.

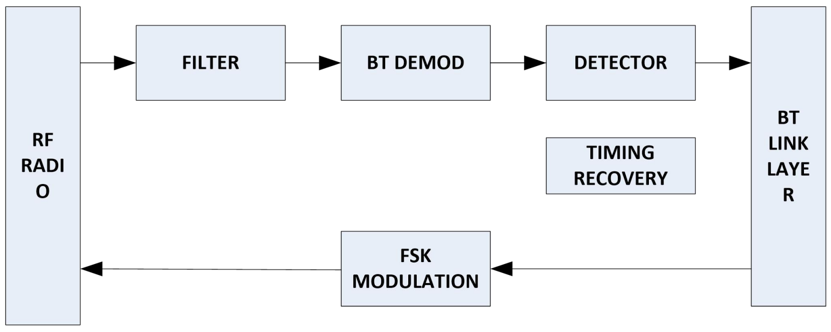


Figure 9.1 HS6621Cx BT Base band

9.4 Performance

9.4.1 BLE Receiver Performance

[Supply Voltage = 3.3V @ 25℃]

Parameter	Min	Typ	Max	Unit
Sensitivity, uncoded data at 1 Ms/s		-95		dBm
Sensitivity, uncoded data at 2 Ms/s		-93		dBm
Senstivity, LE Coded (S=2)		-96		dBm
Sensitibity, LE Coded (S=8)		-100		
Maximum received signal , uncoded data at 1 Ms/s	-		-1.5	dBm
Maximum received signal , uncoded data at 2 Ms/s			-1.5	dBm
Maximum received signal , uncoded data at 500 Ks/s			-1.5	dBm
Maximum received signal , uncoded data at 125 Ks/s			-1.5	dBm
C/I co-channel Sensitivity, uncoded data at 1 Ms/s	-		3	dB
C/I co-channel Sensitivity, uncoded data at 2 Ms/s			2.6	dB

Parameter		Min	Typ	Max	Unit
C/I co-channel Sensitivity, uncoded data at 500 Ks/s				1.2	dB
C/I co-channel Sensitivity, uncoded data at 125 Ks/s				0.5	dB
Adjacent channel selectivity C/I Note: F0=2440MHz	F = F0+1MHz , uncoded data at 1 Ms/s	-	-	-38	dB
	F = F0 -1MHz , uncoded data at 1 Ms/s	-	-	-17	dB
	F = F0+2MHz , uncoded data at 1 Ms/s	-	-	-41	dB
	F = F0-2MHz , uncoded data at 1 Ms/s	-	-	-19	dB
	F = F0+3MHz , uncoded data at 1 Ms/s	-	-	-45	dB
	F = F0-3MHz , uncoded data at 1 Ms/s	-	-	-42	dB
	F = F0+2MHz , uncoded data at 2 Ms/s			-39	dB
	F = F0-2MHz , uncoded data at 2 Ms/s			-17	dB
	F = F0+4MHz , uncoded data at 2 Ms/s			-43	dB
	F = F0-4MHz , uncoded data at 2 Ms/s			-32	dB
	F = F0+6MHz , uncoded data at 2 Ms/s			-46	dB
	F = F0-6MHz , uncoded data at 2 Ms/s			-42	dB
	F = F0+1MHz , uncoded data at 500 Ks/s	-	-	-38	dB
	F = F0 -1MHz , uncoded data at 500 Ks/s	-	-	-26	dB
	F = F0+2MHz , uncoded data at 500 Ks/s	-	-	-51	dB
	F = F0-2MHz , uncoded data at 500 Ks/s	-	-	-24	dB
	F = F0+3MHz , uncoded data at 500 Ks/s	-	-	-53	dB
	F = F0-3MHz , uncoded data at 500 Ks/s	-	-	-53	dB
	F = F0+1MHz , uncoded data at 125 Ks/s	-	-	-36	dB
	F = F0 -1MHz , uncoded data at 125 Ks/s	-	-	-31	dB
	F = F0+2MHz , uncoded data at 125 Ks/s	-	-	-53	dB
	F = F0-2MHz , uncoded data at 125Ks/s	-	-	-29	dB
	F = F0+3MHz , uncoded data at 125 Ks/s	-	-	-57	dB
	F = F0-3MHz , uncoded data at 125 Ks/s	-	-	-55	dB

Table 9.1 HS6621CG/HS6621CM/HS6621CQBLE Receiver architecture

9.4.2 BLE Transmitter Performance

[Supply Voltage = 3.3V @ 25℃]

Parameter		Min	Typ	Max	Unit
Maximum RF transmit power		-	7	-	dBm
RF power control range		-20	-	7	dBm
RF power range control resolution		1.7	2	2.7	dB
ACP Note:F0=2440MHz	F = F0±2MHz	-	-		dBm
	F = F0±>3MHz	-	-		dBm
Δf1avg maximum modulation (uncoded data at 1 Ms/s)		225	250	275	kHz
Δf1avg maximum modulation		450	500	550	kHz

Parameter	Min	Typ	Max	Unit
(uncoded data at 2 Ms/s)				
Δf1avg maximum modulation (LE Coded (S=8))	225	250	275	kHz
Δf2max maximum modulation (uncoded data at 1 Ms/s)	100%			
Δf2max maximum modulation (uncoded data at 2 Ms/s)	100%			
Δf2max maximum modulation (LE Coded (S=8))	100%			
Δf2avg/Δf1avg (uncoded data at 1 Ms/s)	0.84			
Δf2avg/Δf1avg (uncoded data at 2 Ms/s)	0.84			
Frequency Accuracy (uncoded data at 1 Ms/s)		4.03		kHz
Frequency Accuracy (uncoded data at 2 Ms/s)		9.08		kHz
Frequency Accuracy (LE Coded (S=8))		6.08		kHz
Frequency Offset (uncoded data at 1 Ms/s)		4.02		KHz
Frequency Offset (uncoded data at 2 Ms/s)		4.02		KHz
Frequency Offset (LE Coded (S=8))		4.02		KHz
Frequency Drift (uncoded data at 1 Ms/s)		-3.31		KHz
Frequency Drift (uncoded data at 2 Ms/s)		-3.31		KHz
Frequency Drift (LE Coded (S=8))		-3.31		KHz
Frequency Drift rate (uncoded data at 1 Ms/s)		-3.13		KHz/50us
Frequency Drift rate (uncoded data at 2 Ms/s)		-3.13		KHz/50us
Frequency Drift rate (LE Coded (S=8))		-3.13		KHz/50us
Initial Frequency Drift (uncoded data at 1 Ms/s)		-2.25		KHz
Initial Frequency Drift (uncoded data at 2 Ms/s)		-2.25		KHz
Initial Frequency Drift (LE Coded (S=8))		-2.25		KHz
2nd harmonic content			-50	dBm
3rd harmonic content	-		-50	dBm

Table 9.2 HS6621CG/HS6621CM/HS6621CQBLE Transceiver architecture

10 Package Information

HS6621Cx offers QFN48 and QFN32 packages to support different environmental requirements.

10.1 QFN48

HS6621CG is a 48-pin 6x6mm QFN48 package.

SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	0.80	0.85	0.90
A1	0	0.02	0.05
b	0.15	0.20	0.25
c	0.18	0.20	0.23
D	5.90	6.00	6.10
D2	4.10	4.20	4.30
e	0.40BSC		
Ne	4.40BSC		
Nd	4.40BSC		
E	5.90	6.00	6.10
E2	4.10	4.20	4.30
L	0.35	0.40	0.45
h	0.30	0.35	0.40
L/F 载体尺寸（MIL）	177*177		

Table 10.1 HS6621CG QFN48 package information

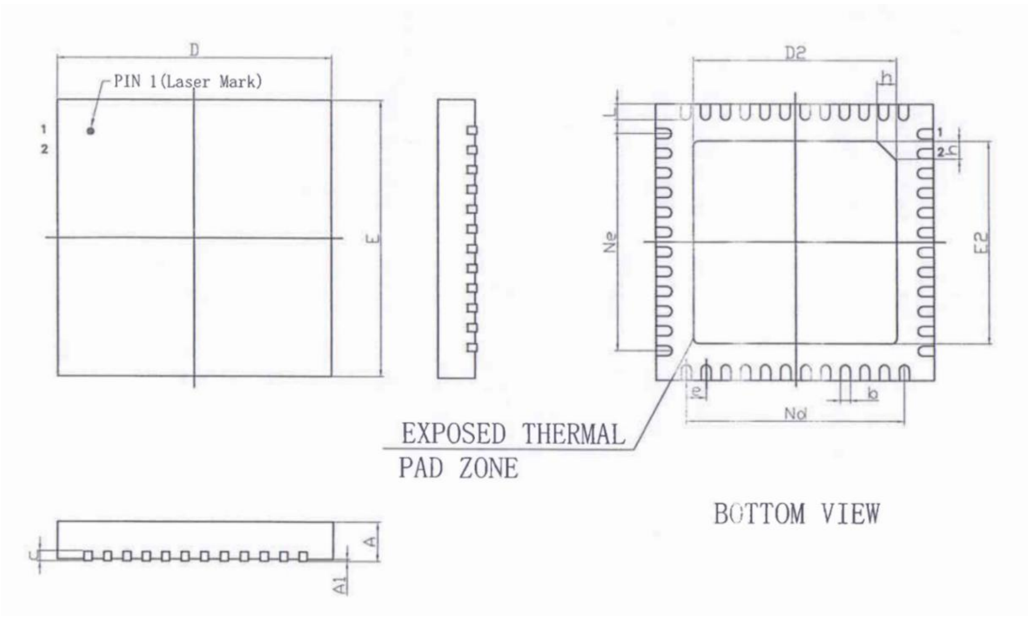


Figure 10.1 HS6621CG QFN48 package outlines

10.2 QFN32

HS6621CM and HS6621CQ is a 32-pin 4x4mm QFN32 package.

SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	0.80	0.85	0.90
A1	0	0.02	0.05
b	0.15	0.20	0.25
b1	0.14REF		
c	0.18	0.20	0.25
D	3.90	4.00	4.10
D2	2.70	2.80	2.90
e	0.40BSC		
Ne	2.80BSC		
Nd	2.80BSC		
E	3.90	4.00	4.10
E2	2.70	2.80	2.90
L	0.25	0.30	0.35
h	0.30	0.35	0.40
L/F 载体尺寸（MIL）	122*122		

Table 10.2 HS6621CQ/HS6621CM QFN32 package information

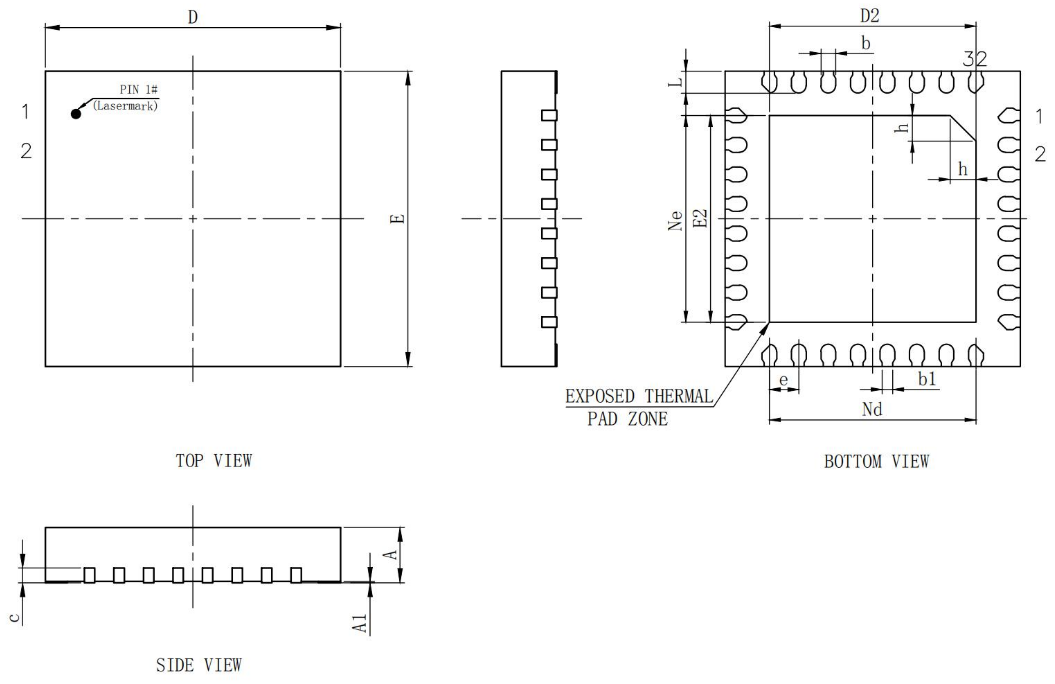


Figure 10.2 HS6621CQ QFN32 package outlines

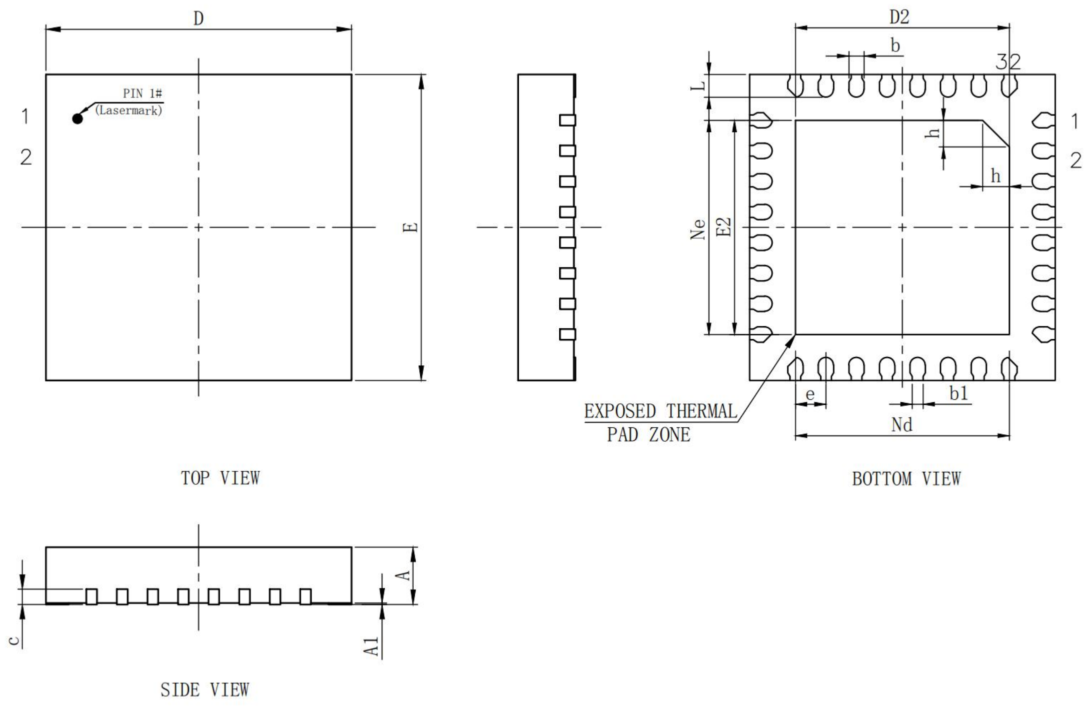


Figure 10.3 HS6621CM QFN32 package outlines

11 Ordering Information

HS6621Cx offers devices below for different application requirement.

Parameter	HS6621CG	HS6621CQ	HS6621CM
CPU	Cortex-M4F	Cortex-M4F	Cortex-M4F
RAM	64KB	64KB	64KB
Flash	8Mb	4Mb	4Mb
Package	QFN48	QFN32	QFN32
Temperature range	-40~85℃	-40~85℃	-40~85℃

Table 11.1 HS6621Cx ordering information

12 Glossary and Abbreviations

Name	Description
ADC	Analog to Digital Converter
AGC	Automatic Gain Control
AON	Always-on
APB	Advanced Peripheral Bus
BB	Base band
BLE	Bluetooth Low Energy
BOD	Brown-out Detector
IFS	Inter Frame Spacing
LDO	Low Dropout
LNA	Low Noise Amplifier
LPD	Low Power Domain
NVM	Non-volatile memory
PLL	Phase Locked Loop
PMU	Power Management Unit
RNG	RING Oscillator
SOC	System-on-chip
TPMS	Tire pressure monitor system
W1C	Write 1 to clear
XO	Crystal Oscillator
Typ	Typical
SNR	Signal to Noise Ratio
PA	Power Amplifier
IRQ	Interrupt Request
LSB	Least Significant Bit
MSB	Most Significant Bit
DFE	Digital Front End

Table 12.1 Glossary and Abbreviations

13 Reference Documents

Below documents are referred in this datasheet:

Documents	Description
HS6621Cx Hardware guideline	Introduce HS6621Cx hardware design.
HS6621Cx Developer User Guide	Introduce HS6621Cx Register and its related resources to developers.

Table 13.1 Reference Documents