

OM6621Dx

Bluetooth Low Energy Compliant & 2.4-GHz Proprietary

System-on-Chip

V1.3

Datasheet

Table of Contents

Version History.....	6
1 OM6621Dx Overview.....	7
1.1 Description.....	7
1.2 Features.....	7
1.3 System Function Block Diagram.....	8
1.4 Applications.....	9
2 Pinout.....	10
2.1 QFN20.....	10
2.2 QFN32.....	11
2.3 Pin Description.....	12
3 MCU Subsystem.....	14
3.1 MCU Debug.....	14
3.2 Interrupts Vector.....	14
3.3 Electrical Specifications.....	15
3.4 Module Address Mapping.....	16
4 Memory.....	17
4.1 Memory Introduction.....	17
4.2 Memory Map.....	17
4.3 APB Address Space.....	17
5 PMU.....	19
5.1 Power Management.....	19
5.2 Digital LDO.....	19
5.3 POR/BOD.....	19
6 Peripherals.....	20
6.1 Pin Mux.....	20
6.2 DMA.....	23
6.3 Control SPI Interface.....	42
6.4 GPIO.....	55
6.5 UARTx.....	65
6.6 SPIx.....	102
6.7 TIMER.....	121
6.8 OTP.....	158
6.9 GPADC.....	167
7 Communication Subsystem.....	169
7.1 Supported Features.....	169
7.2 Radio Transceiver.....	169
7.3 Bluetooth Base band Unit.....	170
7.4 Performance.....	170
8 Package Information.....	173
9 Ordering Information.....	175

10 Tape and reel information.....	176
10.1 Tape orientation.....	176
10.2 Tape and reel dimensions.....	176
11 Glossary and Abbreviations.....	178
12 Reference Documents.....	179

Figure 1.1	OM6621Dx block diagram.....	9
Figure 1.2	OM6621Dx block diagram.....	9
Figure 2.1	OM6621DB chip pin definition.....	10
Figure 2.2	OM6621DQ chip pin definition.....	11
Figure 3.1	Micro-controller Subsystem.....	14
Figure 4.1	Memory Map.....	17
Figure 4.2	APB memory map.....	18
Figure 6.1	Example of DMA Data Transfers.....	24
Figure 6.2	Example of Hardware Handshaking.....	24
Figure 6.3	Linked List Structure for Chain Transfers.....	25
Figure 6.4	Data Order at the Destination	26
Figure 6.5	Data Order at the Destination.....	27
Figure 6.6	Data Order at the Destination	27
Figure 6.7	Specific SPI Device Configuration.....	44
Figure 6.8	Basic structure of a standard I/O port bit.....	55
Figure 6.9	Input floating/pull up/pull down configurations.....	57
Figure 6.10	Output configuration.....	58
Figure 6.11	High impedance-analog configuration.....	58
Figure 6.12	Serial Data Format.....	67
Figure 6.13	Receiver Serial Data Sample Points.....	67
Figure 6.14	Baud Clock Reference Timing Diagram.....	68
Figure 6.15	IrDA SIR Data Format.....	68
Figure 6.16	Timing for RAM Reads.....	70
Figure 6.17	Timing for RAM Writes.....	70
Figure 6.18	RTL Diagram of Data Synchronization Module.....	71
Figure 6.19	Timing Diagram for Data Synchronization Module.....	72
Figure 6.20	Auto Flow Control Block Diagram.....	73
Figure 6.21	Auto RTS Timing.....	74
Figure 6.22	Auto CTS Timing.....	75
Figure 6.23	Flowchart of Interrupt Generation	76
Figure 6.24	Flowchart of Interrupt generation	77
Figure 6.25	Clock Gate Enable Timing.....	78
Figure 6.26	Resuming Clock(s) After Low Power Mode Timing.....	79
Figure 6.27	SPI block diagram.....	103
Figure 6.28	Single master/ single slave application.....	104
Figure 6.29	Data clock timing diagram.....	106
Figure 6.30	General timer block.....	122
Figure 6.31	Counter timing diagram	124
Figure 6.32	Counter timing diagram	124
Figure 6.33	Counter timing diagram	125
Figure 6.34	Counter timing diagram, internal clock divided by 1.....	126
Figure 6.35	Counter timing diagram, internal clock divided by 2.....	126
Figure 6.36	Counter timing diagram, internal clock divided by 4.....	126
Figure 6.37	Counter timing diagram, internal clock divided by N.....	126

Figure 6.38	Counter timing diagram,	127
Figure 6.39	Counter timing diagram,	128
Figure 6.40	Counter sequence diagram,.....	128
Figure 6.41	Counter timing diagram,.....	128
Figure 6.42	Counter timing diagram, internal clock divided by N.....	129
Figure 6.43	Counter timing diagram,	129
Figure 6.44	Counter timing diagram,	129
Figure 6.45	TI1 external clock connection example.....	131
Figure 6.46	Control circuit in external clock mode 1.....	131
Figure 6.47	External trigger input block.....	132
Figure 6.48	Control circuit in external clock mode 2.....	132
Figure 6.49	Capture/compare channel	133
Figure 6.50	Main circuit of capture/compare channel 1.....	133
Figure 6.51	Capture/compare the output portion of the channel (channel 4).....	134
Figure 6.52	PWM Input Mode Timing	136
Figure 6.53	Output compare mode, toggle on OC1.....	137
Figure 6.54	Edge-aligned PWM waveforms (ARR=8).....	138
Figure 6.55	Center-aligned PWM waveforms (ARR=8).....	139
Figure 6.56	Complementary output with dead-time insertion.....	140
Figure 6.57	Dead-time waveforms with delay greater than the negative pulse.....	140
Figure 6.58	Dead-time waveforms with delay greater than the positive pulse.....	141
Figure 6.59	Output behavior in response to a break.....	143
Figure 6.60	Clearing TIMx OCxREF.....	144
Figure 6.61	step generation, COM example (OSSR=1).....	145
Figure 6.62	Example of one pulse mode.....	146
Figure 6.63	Control circuit in reset mode.....	148
Figure 6.64	Control circuit in gated mode.....	149
Figure 6.65	Control circuit in trigger mode.....	149
Figure 6.66	Control circuit in external clock mode2 + trigger mode.....	150
Figure 6.67	OTP controller block diagram.....	159
Figure 6.68	OTP mem state transfer block.....	161
Figure 7.1	OM6621Dx BT Base band.....	170
Figure 8.1	OM6621DB QFN20 package.....	173
Figure 8.2	OM6621DQ QFN32 package.....	174
Figure 10.1	OM6621DQ Tape Orientation.....	176
Figure 10.2	OM6621DQ Tape and Reel Dimensions.....	176

Table 2.1	OM6621DB and OM6621DQ pin definition.....	13
Table 3.1	The MCU interrupt vector.....	15
Table 3.2	OM6621Dx absolute maximum ratings.....	15
Table 3.3	OM6621Dx recommend operating conditions.....	16
Table 3.4	ESD Characteristic.....	16
Table 3.5	Module Address Mapping.....	16
Table 5.1	Digital Core LDO Specifications.....	19
Table 5.2	POR/BOD specifications.....	19
Table 6.1	Peripheral pinmux.....	21
Table 6.2	Format of Linked List Descriptor.....	25
Table 6.3	Detection of SCLK values for narrow pulses is not allowed.....	69
Table 6.4	OTP read timing set.....	166
Table 6.5	OTP read speed.....	167
Table 7.1	OM6621Dx BLE Receiver architecture.....	171
Table 7.2	OM6621Dx BLE Transceiver architecture.....	172
Table 8.1	OM6621DB QFN20 package.....	174
Table 8.2	OM6621DQ QFN32 package.....	174
Table 9.1	OM6621DB and OM6621DQ ordering information.....	175
Table 10.1	OM6621DQ Common Size.....	177
Table 10.2	OM6621DQ Bag Size.....	177
Table 11.1	Glossary and Abbreviations.....	178
Table 12.1	Reference Documents.....	179

Version History

Version	Revision	Date	Author	Reviewer
V1.0	Initial version	2021/05/09	ZJ	
V1.1	Update timer and sflash descriptions	2021/06/08	ZJ	
V1.2	Update QFN20 information	2021/06/29	ZJ	
V1.3	Update Figure3.1 and Figure 6.32	2021/08/04	ZJ	

1 OM6621Dx Overview

1.1 Description

The OM6621Dx is a power-optimized true system-on-chip (SOC) solution for both Bluetooth low energy and proprietary 2.4-GHz applications. It integrates a high performance and low power RF transceiver with Bluetooth base band and rich peripheral IO extension. OM6621Dx also integrates a power management to provide high-efficient power management. It targets 2.4-GHz Bluetooth low energy systems, proprietary 2.4-GHz systems, Human-Interface Devices (keyboard, mouse, and remote control), sports and leisure equipment, mobile phone accessories and consumer electronics.

OM6621Dx on-chip Bluetooth system compliant with version 5.0.

The chip integrates up to 64MHz high-performance MCU, DMA, GPIO, SPI, UART, TIMER, watch dog, supports 32MHz external crystal, integrates multi-purpose 10bit ADC.

The OM6621Dx integrates on chip 24KB ROM, 24K SRAM, 16K OTP and supports user defined IDE system on chip SFLASH MCU development and JTAG software upgrade.

1.2 Features

- RF transceiver
 - -95dBm at 1Mbps sensitivity Bluetooth® low energy
 - TX Power -20 to +10dBm
 - 14.1mA peak RX, 12.8mA peak TX (0dBm)
- CPU
 - ARM® Cortex™-M4, max 64MHz
 - Serial Wire Debug (SWD)
- Memory
 - 16KB OTP
 - 24KB SRAM
 - 24KB ROM
 - Serial Flash
 - QFN20:2Mb
 - QFN32:4Mb
 - Icache RAM 2KB
- Clocks
 - 32MHz crystal, 32MHz RC, 32.768KHz RC
- Link Controller
 - BT 5.0 LE PHY, link controller
 - Proprietary 2.4-GHz link controller
- Power Management

- Deep sleep power 2.5uA
- Supply voltage range 1.8V~3.6V
- Integrated Charger(100mA/200mA)
- Software
 - compliant with BLE version 5.0
 - Supports mesh network
 - Network processor interface for applications running on an external micro controller
 - Sample applications and profiles
 - Supports 6LowPAN
 - Supports OTA
 - SWD interface
- Peripherals
 - DMA x 4
 - UART x 1
 - Flexible general-purpose I/O GPIO 19GPIO(max)
 - SPI master or slave interface x 1
 - PWM x 6
 - Watchdog to prevent system dead lock
 - 32bit timer x 3
 - single-end 10bits GPADC x 4
 - AES HE encryption
- Package
 - OM6621DB: 20-pin 3x3mm QFN20
 - OM6621DQ: 32-pin 4x4mm QFN32

1.3 System Function Block Diagram

OM6621Dx is a low power Bluetooth wireless transceiver chip. The chip integrates Bluetooth base band, PHY and proprietary 2.4GHz protocol. The MCU accesses system hardware resource by AHB bus, ROM, RAM, DMA, SFLASH, GPIO exchange data through AHB bus, and all other peripheral is accessed through AHB to APB Bridge and APB bus.

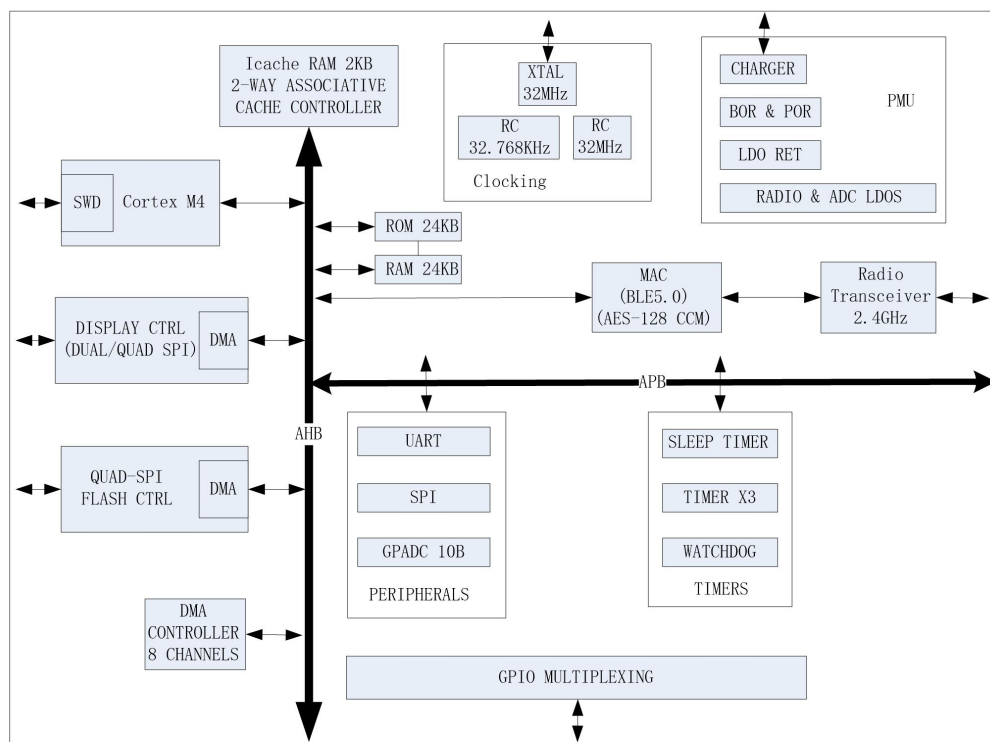


Figure 1.1 OM6621Dx block diagram

1.4 Applications

The OM6621Dx integrated circuit has a fully integrated radio transceiver and base band processor for Bluetooth® Smart. It can be used as an application processor as well as a data pump in fully hosted systems.



Figure 1.2 OM6621Dx block diagram

2 Pinout

2.1 QFN20

The OM6621DB is in the 3mmx3mm QFN20 package. The chip pin definition is as below:

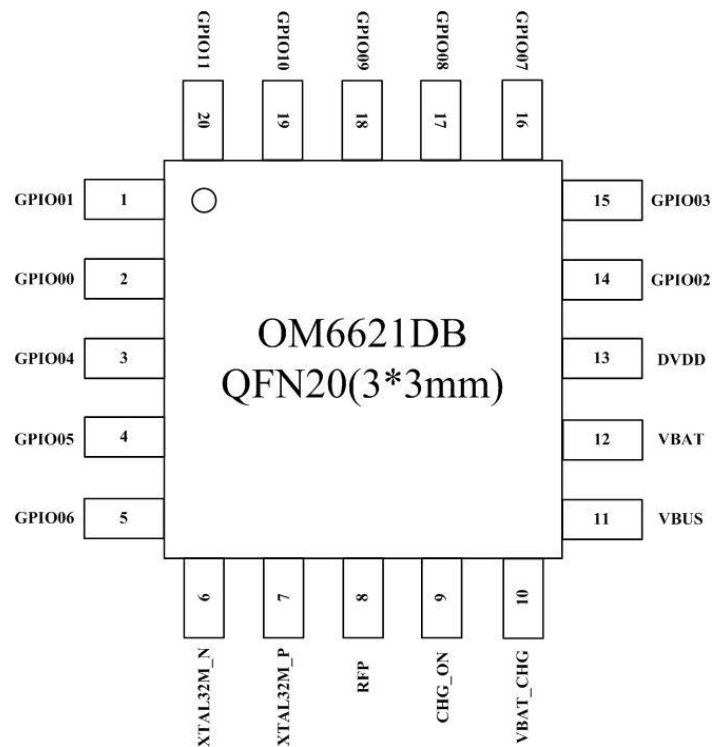


Figure 2.1 OM6621DB chip pin definition

2.2 QFN32

The OM6621DQ is in the 4mmx4mm QFN32 package. The chip pin definition is as below:

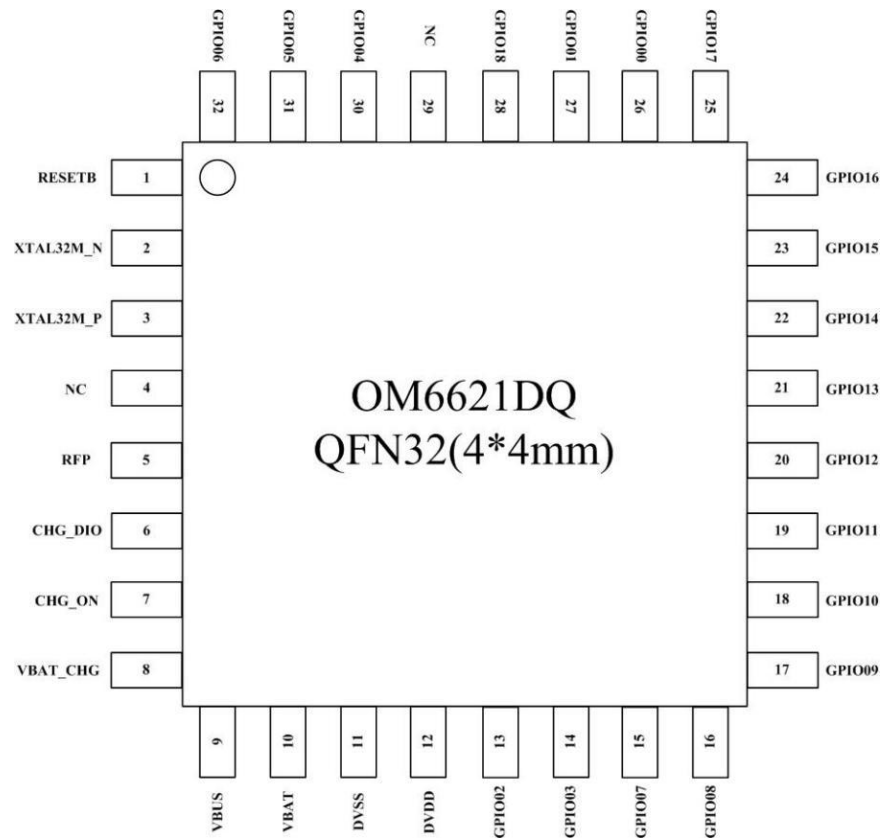


Figure 2.2 OM6621DQ chip pin definition

2.3 Pin Description

Name	OM6621DB	OM6621DQ	Type	Description	Note
RESETB	-	1	Digital	Reset signal	
XTAL32M_N	6	2	Analog	32M crystal oscillator P input	
XTAL32M_P	7	3	Analog	32M crystal oscillator N input	
NC	-	4			
RFP	8	5	RF	RF input/output	
CHG_DIO	-	6	Analog	Power supply from charger	
CHG_ON	9	7	Analog	Charge indicator	
VBAT_CHG	10	8	Power	Battery charge pad	
VBUS	11	9	Power	Charger power	
VBAT	12	10	Power	Power supply 1.8V~3.6V	
DVSS	-	11	Power	Digital ground	
DVDD	13	12	Power	Digital Circuit Power	
GPIO02	14	13	Digital/ Analog	Digital GPIO/GP-ADC input	Note 1
GPIO03	15	14	Digital/ Analog	Digital GPIO/GP-ADC input	Note 1
GPIO07	16	15	Digital/ Analog	Digital GPIO/GP-ADC input	Note 1
GPIO08	17	16	Digital/ Analog	Digital GPIO/GP-ADC input	Note 1
GPIO09	18	17	Digital/ Analog	Digital GPIO	Note 1
GPIO10	19	18	Digital/ Analog	Digital GPIO	Note 1
GPIO11	20	19	Digital/ Analog	Digital GPIO	Note 1
GPIO12	-	20	Digital/ Analog	Digital GPIO	Note 1
GPIO13	-	21	Digital/ Analog	Digital GPIO	Note 1
GPIO14	-	22	Digital/ Analog	Digital GPIO	Note 1
GPIO15	-	23	Digital/ Analog	Digital GPIO	Note 1

GPIO16	-	24	Digital/ Analog	Digital GPIO	Note 1
GPIO17	-	25	Digital/ Analog	Digital GPIO	Note 1
GPIO00	2	26	Digital/ Analog	Digital GPIO	Note 1
GPIO01	1	27	Digital/ Analog	Digital GPIO	Note 1
GPIO18	-	28	Digital/ Analog	Digital GPIO	Note 1
NC	-	29			
GPIO04	3	30	Digital/ Analog	Digital GPIO	Note 1
GPIO05	4	31	Digital/ Analog	Digital GPIO	Note 1
GPIO06	5	32	Digital/ Analog	Digital GPIO	Note 1

Note 1: All digital peripheral pins can be programmed to any GPIO

Table 2.1 OM6621DB and OM6621DQ pin definition

3 MCU Subsystem

OM6621Dx has an MCU subsystem that contains an ARM Cortex-M4 processor, its corresponding buses and peripherals, including all the multiplexing options for the GPIOs, as illustrated in the following figure.

The processor has 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-M4:

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

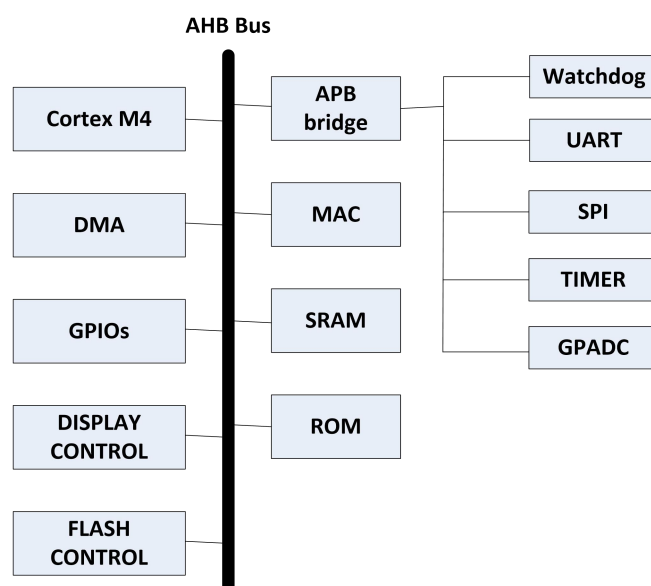


Figure 3.1 Micro-controller Subsystem

3.1 MCU Debug

Serial Wire Debug (SWD) is used for debug.

3.2 Interrupts Vector

The following table shows the MCU interrupt vector table for OM6621Dx.

Number	Interrupt name	bit	note
0	BT BB combo	1	
1	ceva_native_int	1	ceva sleep wake
2	DMA combo	1	
3	GPIO combo	1	
4	Timer combo	1	
5	6200_rf_irq	1	
6	6200_rf_spi_irq	1	
7	PMU_timer	1	
8	Lcd_spi_int	1	
9	UART1 combo	1	
10	otp_int	1	
11	pin_wakeup_int	1	
12	ADC	1	
13	SPI master 0 combo	1	
14	SFLASH int	1	
15-22	soft_int	1	soft interrupt
23	vtrack_int	1	vtrack interruption indicates AFC need to be redone
24	cry32m_dig_ready	1	32M crystal oscillator can give digital flag
26	GPIO	1	GPIO0 COMBO
27	cc_intr		
28-30	Timer	3	Timers
31	charger_int	1	Charging and plug interruption

Table 3.1 The MCU interrupt vector

3.3 Electrical Specifications

3.3.1 Absolute Maximum Ratings

Parameter	Minimum	Maximum	Units
Supply voltage(VBAT)	-0.3	3.9	V
Charger voltage(VBUS)	-0.3	6.0	V
Maximum Junction Temperature	-40	125	°C
Storage Temperature	-40	125	°C

Table 3.2 OM6621Dx absolute maximum ratings

3.3.2 Recommend Operating Conditions

Rating	Min	Typ	Max	Unit
Operation Temperature	-40	-	85	°C
Digital Core supply voltage	0.9	1.0	1.2	V
RF supply voltage	1.1	1.25	1.4	V
I/O voltage (Vsupply>3.3)	3.1	3.3	3.5	V
I/O voltage (Vsupply<3.3)	Vsupply	Vsupply	Vsupply	V
Supply voltage(VBAT)	1.8	3.3	3.6	V
Charger voltage(VBUS)	4.5	5.0	5.5	V

Table 3.3 OM6621Dx recommend operating conditions

3.3.3 ESD Characteristic

Parameter	Condition	Minimum	Typical	Maximum
HBM	Test method: ESDA/JEDEC JS-001-2017	-	±4000 V	±8000 V
MM	All pins, test method: JESD22 -A115C	-	-	±200 V
CDM	All pins, test method: ESDA/JEDEC JS-002-2018	-	-	±1000 V

Table 3.4 ESD Characteristic

3.4 Module Address Mapping

Base Address	Module
0x40000000	SYS_REG
0x40001000	CPM_PSO
0x40004000	RNG
0x40008000	CPM_ROM_P
0x4000B000	6200
0x40020000	BT_PHY
0x40040000	UART1
0x40050000	SPI0
0x400A0000	DA_IF
0x400C0000	TIMER
0x400E0000	PMU

Table 3.5 Module Address Mapping

4 Memory

4.1 Memory Introduction

OM6621Dx 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

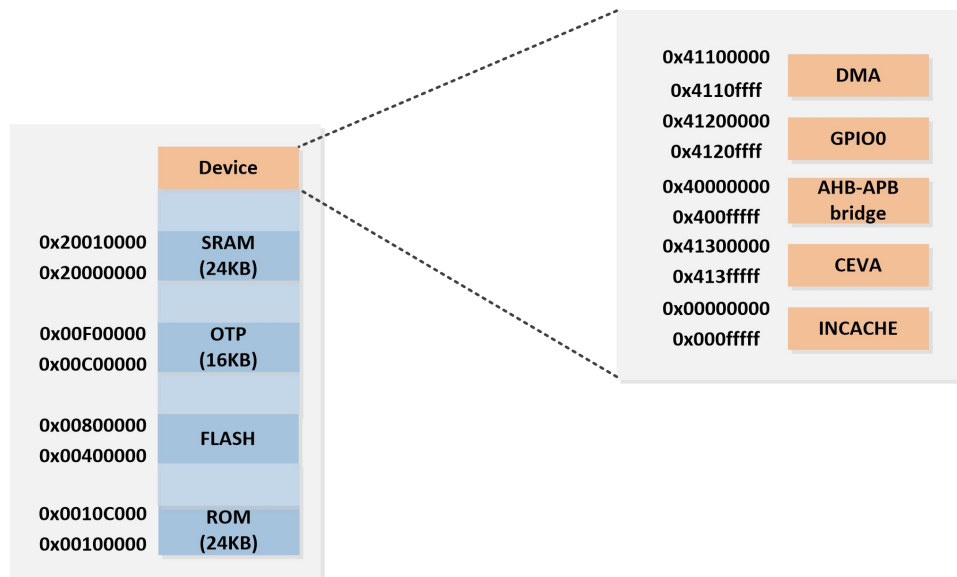


Figure 4.1 Memory Map

4.3 APB Address Space

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

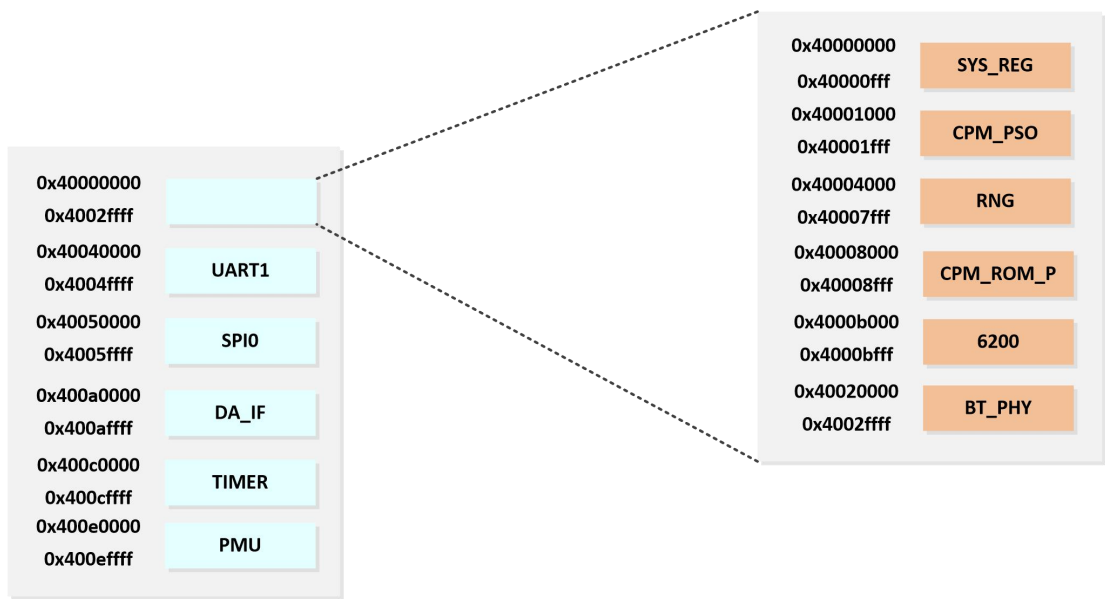


Figure 4.2 APB memory map

5 PMU

5.1 Power Management

There are three different power modes in the OM6621Dx:

- Active Mode: System is active and operates at full speed.
- Sleep Mode: No power gating has been programmed; the CPU is idle, waiting for an interrupt. 32M crystal is on, 32K RC is on. Peripherals is depending on the programmed enabled value.
- Extended Sleep Mode: All power domains are off except for the always on power domain, the programmed Bluetooth timer module, 32M crystal is off, 32K RC is on. The data retention SRAM retains its data and other SRAM is power off. It is wake by timer or GPIOs.

5.2 Digital LDO

Digital LDO regulates the DC-DC converter to supply power to all the Digital Logic and Memory blocks.

Parameter	Symbol	Min	Typ	Max	Unit	Comment
Input Voltage			3.3		V	
Output Voltage			1.0		V	
External Load Cap			1.0		μF	

Table 5.1 Digital Core LDO Specifications

5.3 POR/BOD

Power-on Reset (POR) circuit holds the system at reset while the supply 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
Brown Out Threshold			1.4		V	

Table 5.2 POR/BOD specifications

6 Peripherals

6.1 Pin Mux

6.1.1 Introduction

OM6621Dx has a configurable pin multiplexing module (Pin MUX) which can bring different peripherals on different GPIOs.

6.1.2 Main Features

pinmux has the following features:

- There are 41 mux choices for the Peripheral pinmux.
- Peripheral pinmux can be config by REG_GPIOX_MUX.

6.1.3 Function Description

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

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

NAME	NUMBER
PINMUX_JTAG_MODE_CFG	0
PINMUX_DBG_MODE_CFG	1
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_SFLASH_SI_CFG	8
PINMUX_SFLASH_SO_CFG	9
PINMUX_SFLASH_HD_CFG	10
PINMUX_SFLASH_WP_CFG	11
PINMUX_SFLASH_CK_CFG	12
PINMUX_SFLASH_CSN_CFG	13
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_TX_EXT_PD_CFG	19
PINMUX_RX_EXT_PD_CFG	20
PINMUX_GPIO_ADC_CFG	26
PINMUX_GPIO_MODE_CFG	28
PINMUX_SFLASH1_CSN_1_CFG	24
PINMUX_SFLASH1_SI_CFG	25
PINMUX_SFLASH1_SO_CFG	28
PINMUX_SFLASH1_HD_CFG	30
PINMUX_SFLASH1_WP_CFG	31
PINMUX_SFLASH1_CK_CFG	32
PINMUX_SFLASH1_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_TOGGLE_N_0_CFG	44
PINMUX_TIMER1_IO_0_CFG	47
PINMUX_TIMER1_IO_1_CFG	48
PINMUX_TIMER1_TOGGLE_N_0_CFG	51
PINMUX_TIMER2_IO_0_CFG	54
PINMUX_TIMER2_IO_1_CFG	55
PINMUX_TIMER2_TOGGLE_N_0_CFG	58

Table 6.1 Peripheral pinmux

6.1.4 Pin MUX Register Map

Address	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

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
31:23	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

6.2 DMA

6.2.1 Introduction

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

6.2.2 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.

6.2.3 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.

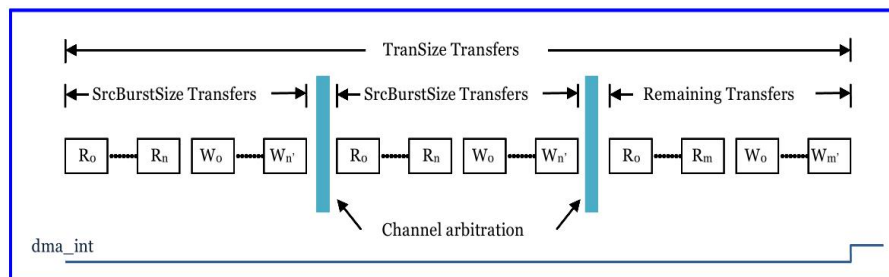


Figure 6.1 Example of DMA Data Transfers

6.2.3.1 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.

6.2.3.2 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.

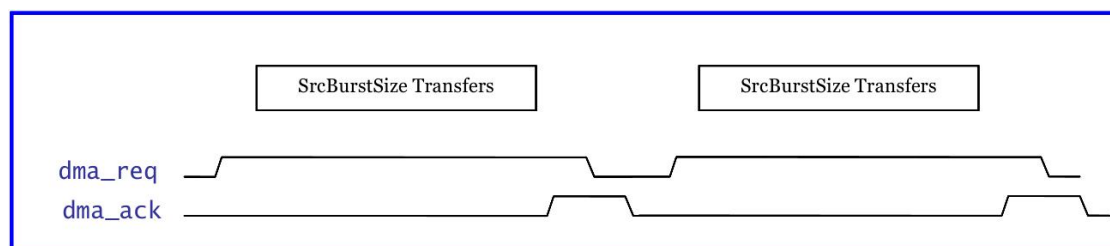


Figure 6.2 Example of Hardware Handshaking

6.2.3.3 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.

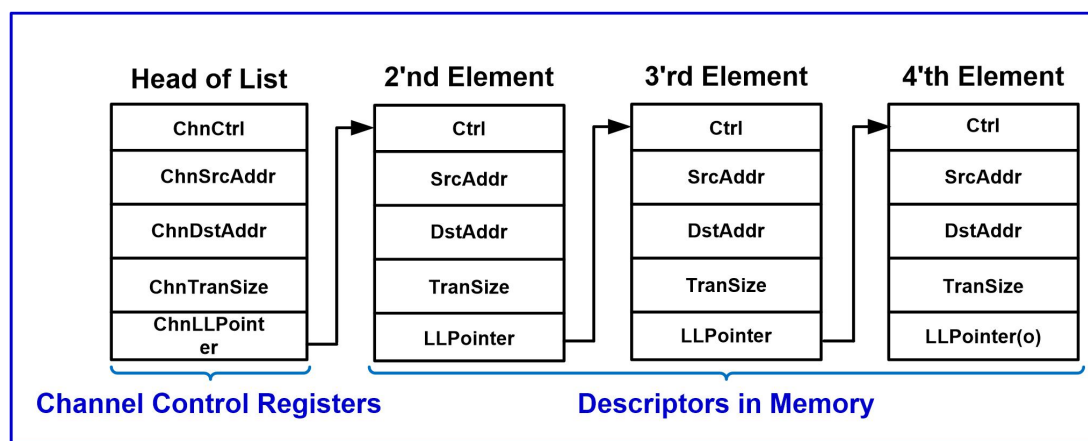


Figure 6.3 Linked List Structure for Chain Transfers

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.

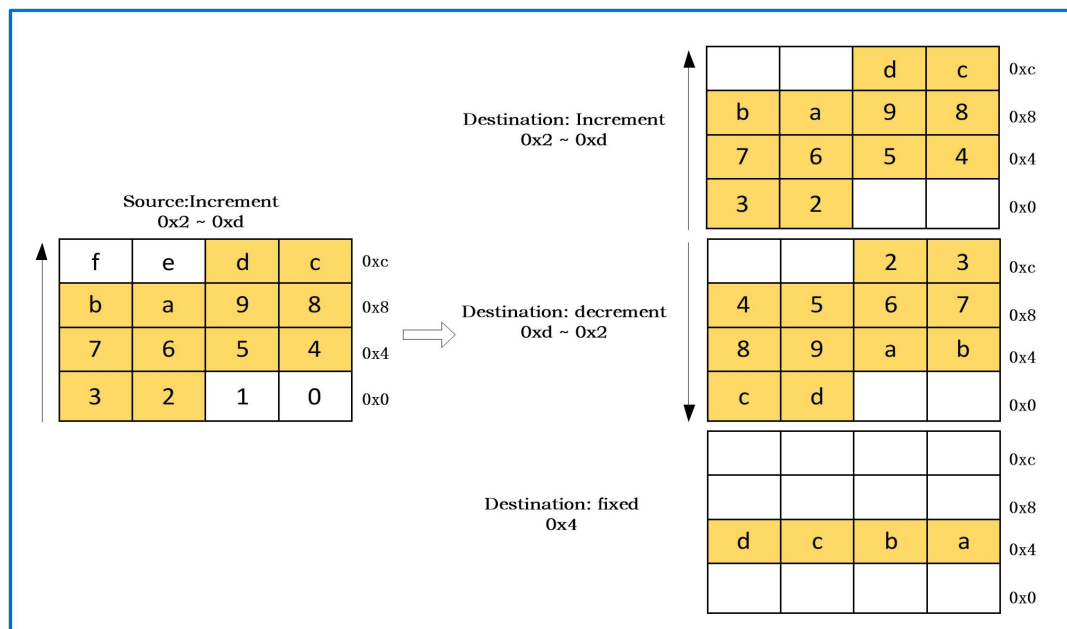
Name	Offset	Description	Format
Ctrl	0x00	Channel control	See DMA Register Map
SrcAddr	0x04	Source address	See DMA Register Map
DstAddr	0x08	Destination address	See DMA Register Map
TranSize	0x0c	Total transfer size	See DMA Register Map
LLPointer	0x10	Linked list pointer	See DMA Register Map

Table 6.2 Format of Linked List Descriptor

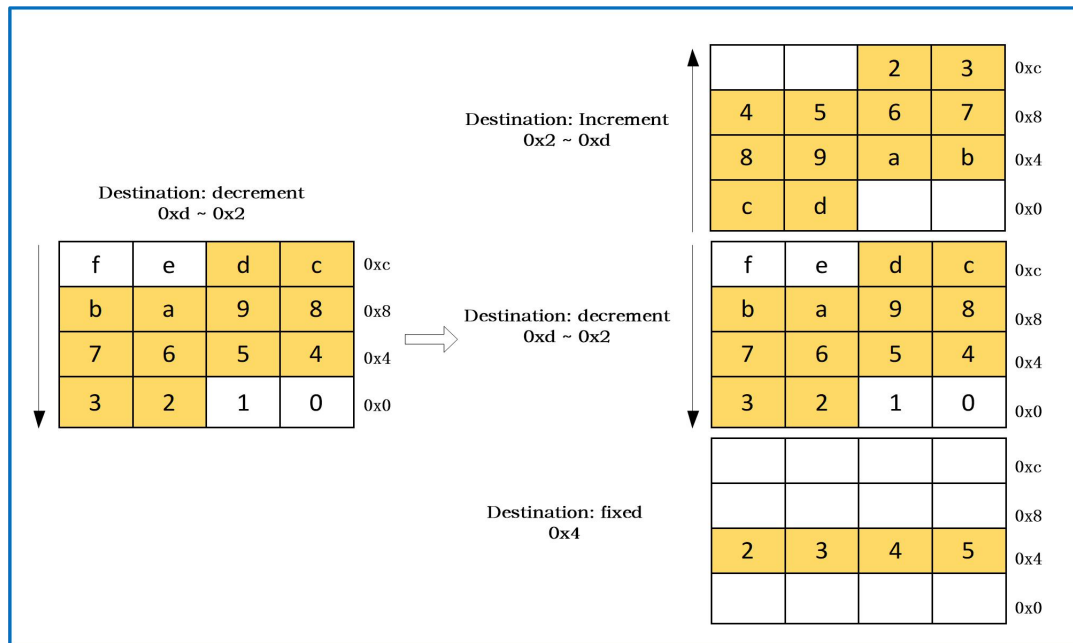
6.2.3.4 Data Order

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.

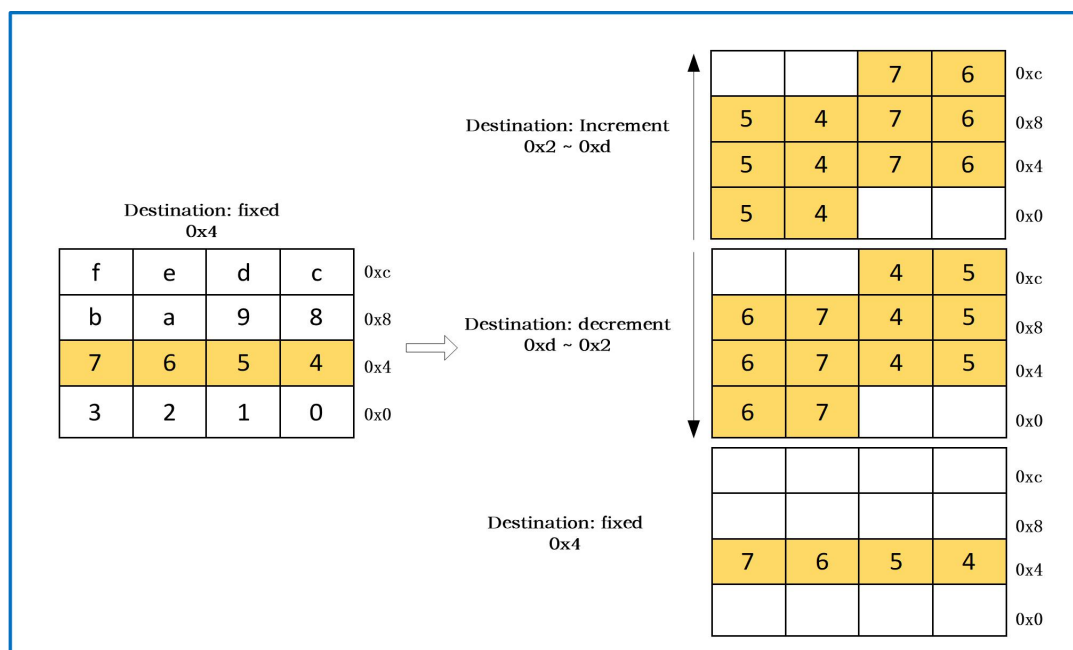
When the address control mode of the source is the same as that of the destination, the DMA controller maintains the same byte order of the data between the source and the destination. When the address control mode of the source is opposite to that of the destination, the data written to the destination 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. Figure 6.4, Figure 6.5 and Figure 6.6 illustrate the byte order of the data at the destination when the source address mode is increment, decrement, and fixed respectively.



**Figure 6.4 Data Order at the Destination
when the Source Address Mode is the Increment Mode**



**Figure 6.5 Data Order at the Destination
when the Source Address Mode is the Decrement Mode**



**Figure 6.6 Data Order at the Destination
when the Source Address Mode is the Fixed Mode**

6.2.4 DMA Register Map

offset	Name	Description
0x00	IdRev	ID and revision register
0x04	NA	Reserved
0x08	NA	Reserved

0x0C	NA	Reserved
0x10	DMACfg	DMAC configuration register
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 Configuration Register(offset 0x10)

Bit	R/W	Reset	Name	Description
31	R	Configuration dependent	ChainXfr	Chain transfer 0=chain transfer is not configured 1=chain transfer is configured

30	R	Configuration dependent	RevMajor	Major revision number
29:15	NA	Configuration dependent	ReqSync	DMA request synchronization. The DMA request synchronization should be configured to avoid signal integrity problems when the request signal is not clocked by the system bus clock, which the DMA control logic operates in. If the request synchronization is not configured, the request signal is sampled directly without synchronization. 0=request synchronization is not configured 1=request synchronization is configured
14:10	R	Configuration dependent	ReqNum	Request/acknowledge number
9:4	R	Configuration dependent	FIFO Depth	FIFO depth
3:0	R	Configuration dependent	ChannelNum	Channel 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	<p>The abort status of channel, one bit per channel.</p> <p>The abort status is asserted when a channel transfer is aborted.</p> <p>0=channel N has no abort status</p> <p>1=channel N has abort status</p>
7:0	R/W ₁ C	0x0	Error	<p>The error status, one bit per channel. The error status is asserted when a channel transfer encounters the following error events:</p> <ul style="list-style-type: none"> Bus error Unaligned address Unaligned transfer width Reserved configuration <p>0=channel N has no error status</p> <p>1=channel N has error status</p>

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	<p>Write 1 to this field to stop the channel transfer.</p> <p>The bits can only be set when the corresponding channels are enabled. Otherwise, the writes will be ignored for channels that are not enabled.</p>

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	<p>Channel priority level.</p> <p>0=lower priority</p> <p>1=higher priority</p>
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 triggered 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

				<p>completes, its value is updated to the ending address + sizeof(DstWidth).</p> <p>This address must be aligned to the destination transfer size; otherwise the error event will be triggered.</p>
--	--	--	--	---

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	<p>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.</p> <p>If a channel is enabled with zero total transfer size, the error event will be triggered and the transfer will be terminated.</p>

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 SSTATx 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 DSTATx 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 CFGx.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 CFGx.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 SSTATx 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 DSTATx

				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	Channel 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

6.3 Control SPI Interface

6.3.1 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.

6.3.2 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

6.3.3 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:

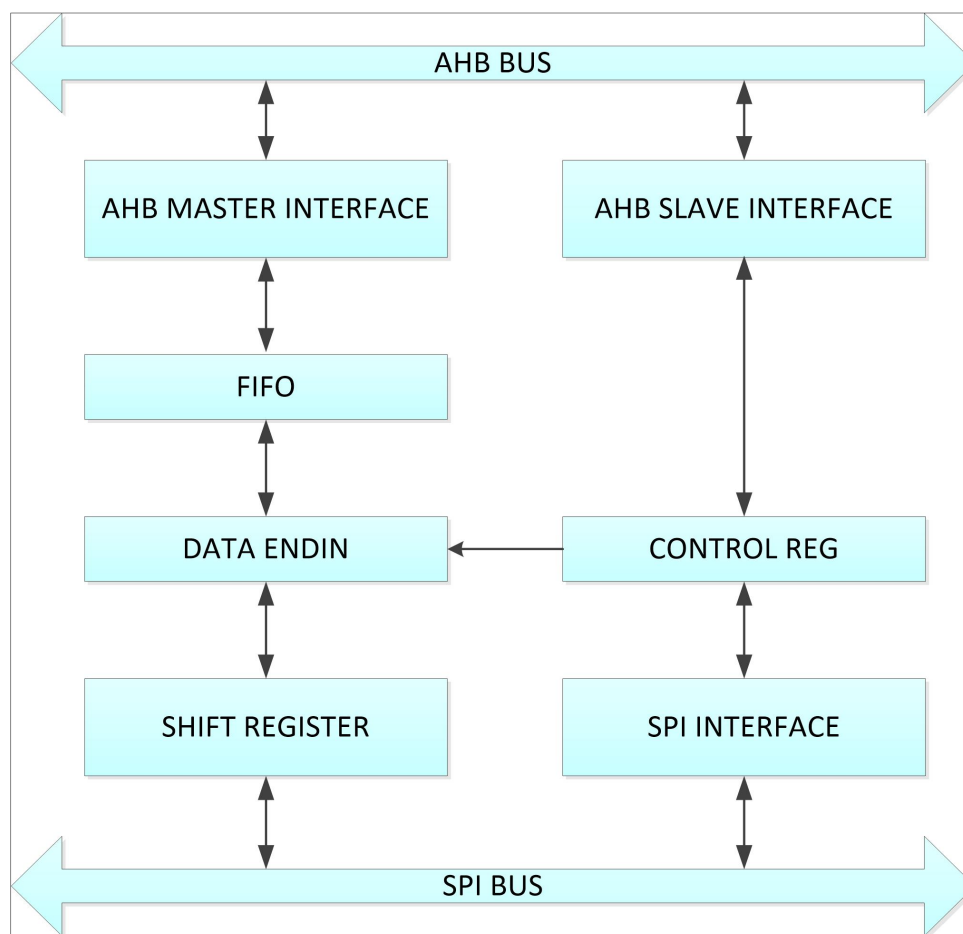


Figure 6.7 Specific SPI Device Configuration

6.3.4 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

6.3.5 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	RO	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-bit registers. 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 first 32 SPI clock cycles, depending on the CMD_BITS field. It is sent MSB first (data is left-shifted out of bit 31). 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 first 32 SPI clock cycles, depending on the CMD_BITS field. It is sent MSB first (data is left-shifted out of bit 31). 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 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.
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 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.
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_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.

6.3.6 SFLASH software program guide

6.3.6.1. 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.

6.3.6.2. 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.

6.3.6.3. 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 Data1 register 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.

6.3.6.4.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 write 0x0b000000 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.

- ## 6.4 GPIO

6.4.1 Introduction

The diagram illustrates the internal architecture of an I/O pin. On the left, external signals are shown: 'Analog Input' and 'Alternate Function Input' pointing to the 'Input data register'; 'Read' pointing to the 'Input data register'; 'Write' pointing to the 'Bit set/reset registers'; and 'Read/write' pointing to the 'Output data register'. The 'Bit set/reset registers' and 'Output data register' are connected to an 'Output driver' block. The 'Output driver' contains an 'Output control' block and a 'Push-pull, open-drain or disabled' switch controlled by V_{DD} and V_{SS} . The 'Input driver' block contains a 'Schmitt trigger' and a switch controlled by 'on/off' signals. The 'Input data register' is connected to the 'Schmitt trigger' and the 'on/off' switch. The 'Output driver' and 'Input driver' are connected to the 'I/O pin' on the right. The 'I/O pin' is protected by two 'Protection diode's connected to V_{DD} and V_{SS} .

Figure 6.8 Basic structure of a standard I/O port bit

6.4.2 Main Features

- OM6621Dx: 19 general-purpose I/O GPIOs (max)

6.4.3 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

6.4.3.1 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

6.4.3.2 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.

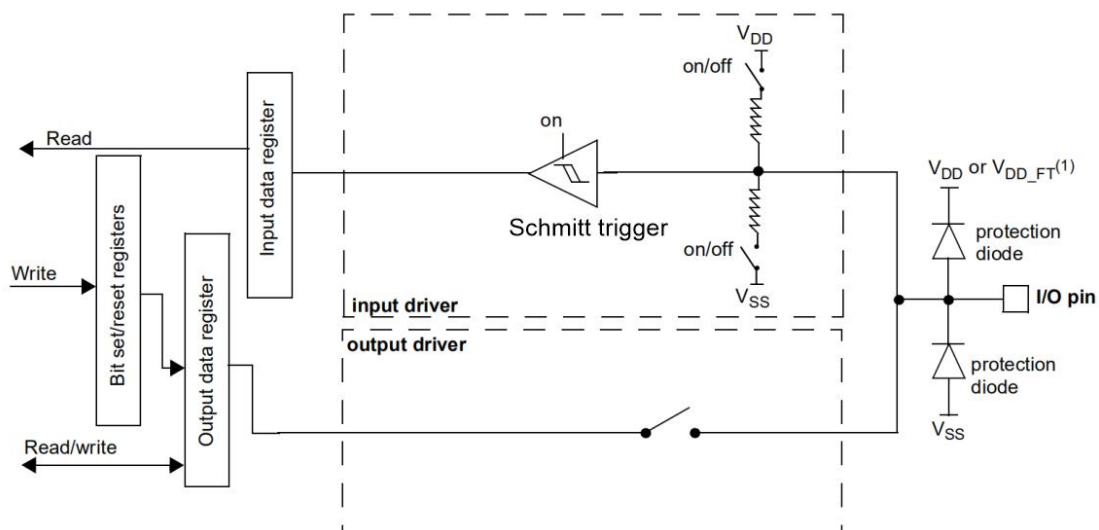


Figure 6.9 Input floating/pull up/pull down configurations

6.4.3.3 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.

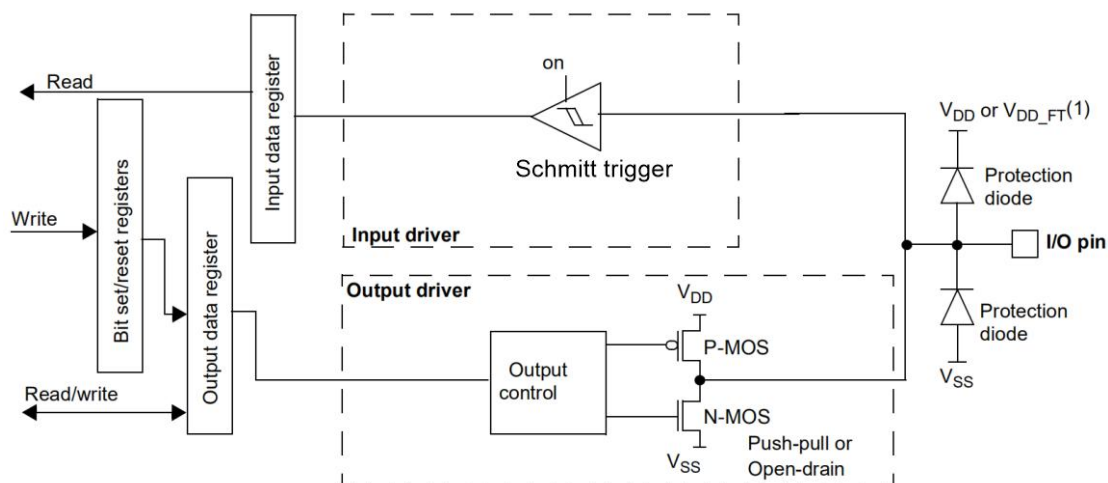


Figure 6.10 Output configuration

6.4.3.4 Analog configuration

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

- The Output Buffer is disabled.
- The Schmitt Trigger Input is deactivated 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.

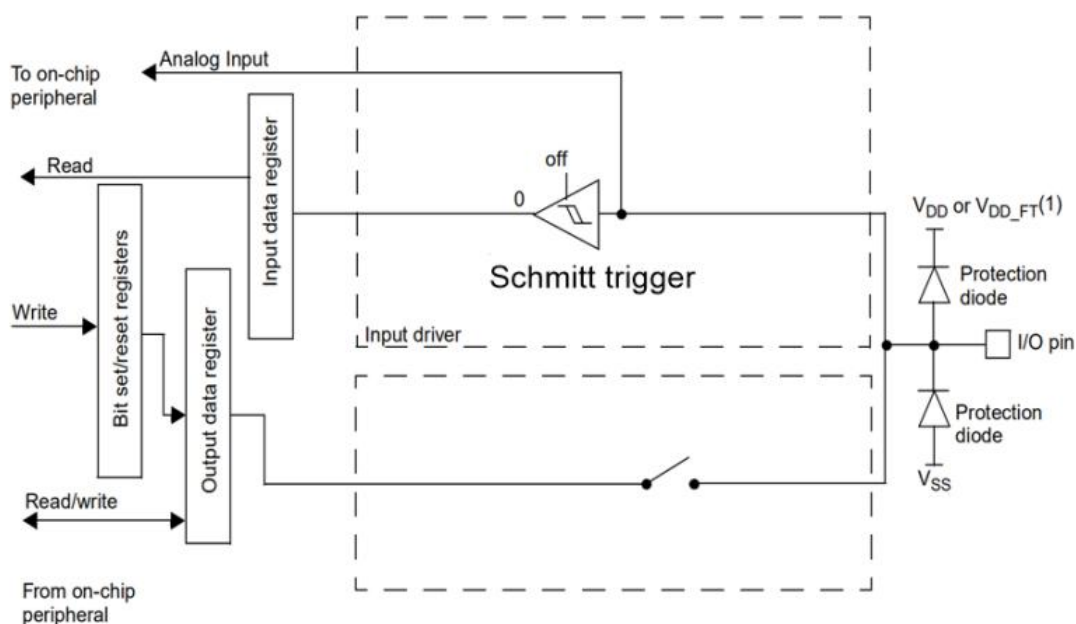


Figure 6.11 High impedance-analog configuration

6.4.4 GPIO Register Map

Offset	Name	Description
0x00000	GPIO_DATA	Data
0x00004	GPIO_DATAOUT	Data output latch
0x00010	GPIO_OUTENSET	Output enable set
0x00014	GPIO_OUTENCLR	Output enable clear
0x00018	GPIO_ALTFUNCSET	Alternative function set
0x0001c	GPIO_ALTFUNCCLR	Alternative function set
0x00020	GPIO_INTENSET	Interrupt enable set
0x00024	GPIO_INTENCLR	Interrupt enable clear
0x00028	GPIO_INTTYPESET	Interrupt type set
0x0002c	GPIO_INTTYPECLR	Interrupt type clear
0x00030	GPIO_INTPOLSET	Interrupt polarity set
0x00034	GPIO_INTPOLCLR	Interrupt polarity clear
0x00038	GPIO_INTSTATUS	Interrupt status
0x00040	GPIO_INTBOTHSET	Interrupt type1 set
0x00044	GPIO_INTBOTHCLR	Interrupt type1 clear
0x01000-0x13FC	GPIO_MASKBYTE0	Byte 0 masked access
0x01400-0x17FC	GPIO_MASKBYTE1	Byte 1 masked access
0x01800-0x1BFC	GPIO_MASKBYTE2	Byte 2 masked access
0x01c00-0x1FFC	GPIO_MASKBYTE3	Byte 3 masked access
0x400E002C	GPIO_OE_CTRL	GPIO output enable control
0x400E0034	GPIO_PU_CTRL	GPIO pull up 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
0x400E00CC	GPIO_DRV_CTRL_2	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. 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
19	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
18:0	RW	0x7fff	gpio_oeb_reg[18:0]	Only valid when gpio_oeb_sel=1 GPIO[18:0] output enable control 0: output enable

GPIO_PU_CTRL address offset: 0x0034

Bit	R/W	Reset	Name	Description
18:0	RW	0x010	gpio_pu_reg[18:0]	GPIO[18:0] pull up control, high active

GPIO_ODA_CTRL address offset: 0x0040

Bit	R/W	Reset	Name	Description
18:0	RW	0x0	gpio_oda_ctrl[18:0]	GPIO[18:0] output data (Only valid when gpio_oeb_sel=1)

GPIO_IE_CTRL address offset: 0x0054

Bit	R/W	Reset	Name	Description
18:0	RW	0x1f	gpio_ie_ctrl[18:0]	GPIO[18:0] input enable, high active

GPIO_ODE_CTRL address offset: 0x00a0

Bit	R/W	Reset	Name	Description
18:0	RW	0x0	gpio_ode_reg[18:0]	gpio[18:0] open drain control, high active

GPIO_PD_CTRL address offset: 0x00a8

Bit	R/W	Reset	Name	Description
18:0	RW	0x0	gpio_pd_reg[18:0]	gpio[18:0] pull down control, high active

GPIO_DRV_CTRL_0 address offset: 0x00c4

Bit	R/W	Reset	Name	Description
18:0	RW	0x0	gpio_drv_ctrl_0[18:0]	GPIO driver strength

GPIO_DRV_CTRL_2 address offset: 0x00cc

Bit	R/W	Reset	Name	Description
18:0	RW	0x7fff	gpio_drv_ctrl_1[18:0]	GPIO driver strength

6.5 UARTx

6.5.1 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 design ware 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 Onmicro_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 .

6.5.2 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 design ware 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 = $3/16 \times$ 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
- Loop back mode that enables greater testing of Modem Control and Auto Flow Control features (Loop back 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 \times 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

6.5.3 Function Description

6.5.3.1 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.

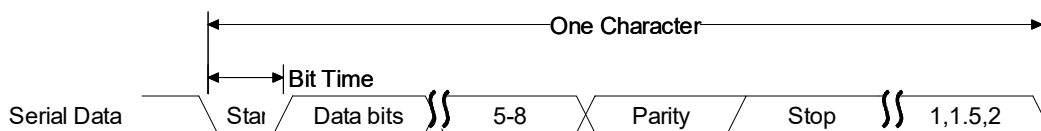


Figure 6.12 Serial Data Format

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 .

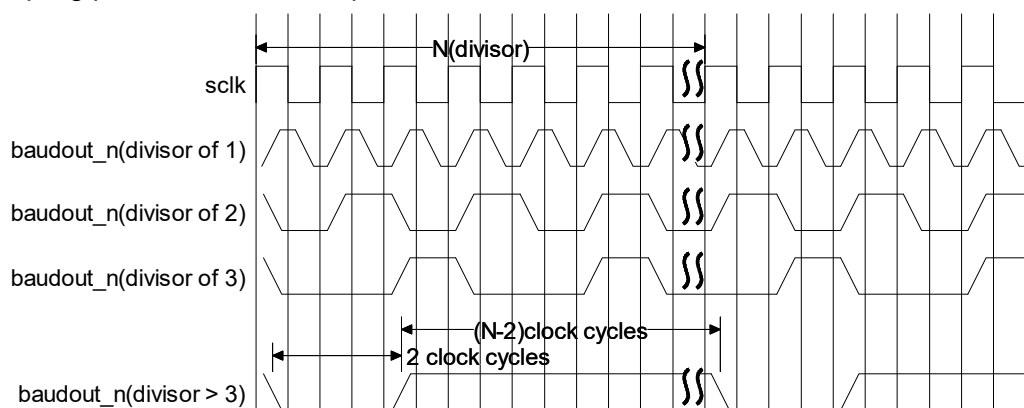


Figure 6.13 Receiver Serial Data Sample Points

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.

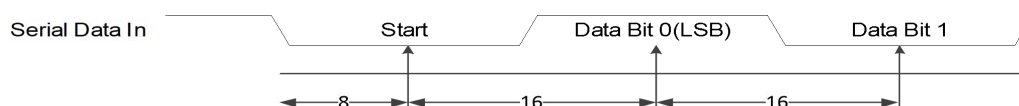


Figure 6.14 Baud Clock Reference Timing Diagram

6.5.3.2 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.

6.5.3.3 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.

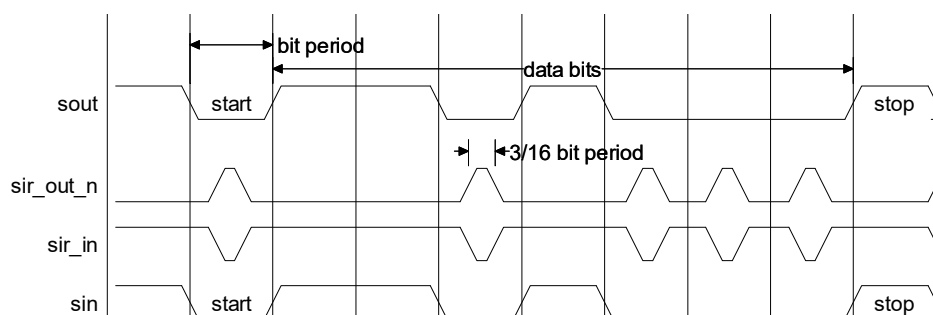


Figure 6.15 IrDA SIR Data Format

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 6.3 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.

6.5.3.4 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.

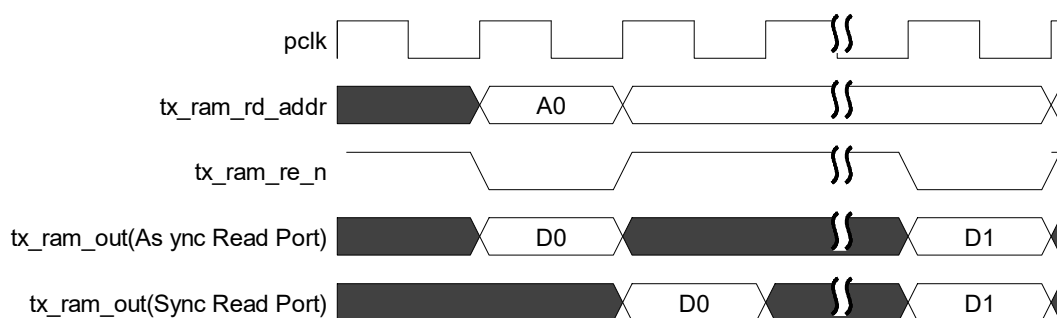


Figure 6.16 Timing for RAM Reads

Note: This timing diagram illustrated in the following figure 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.

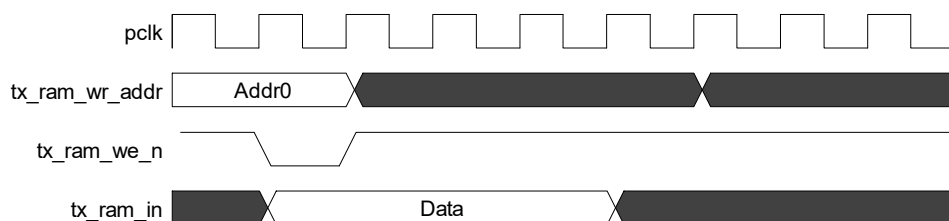


Figure 6.17 Timing for RAM Writes

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

6.5.3.5 Clock Support

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 the Figure 6.18. The data synchronization module can have pending data capability. The timing diagram shown in the Figure 6.19 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.

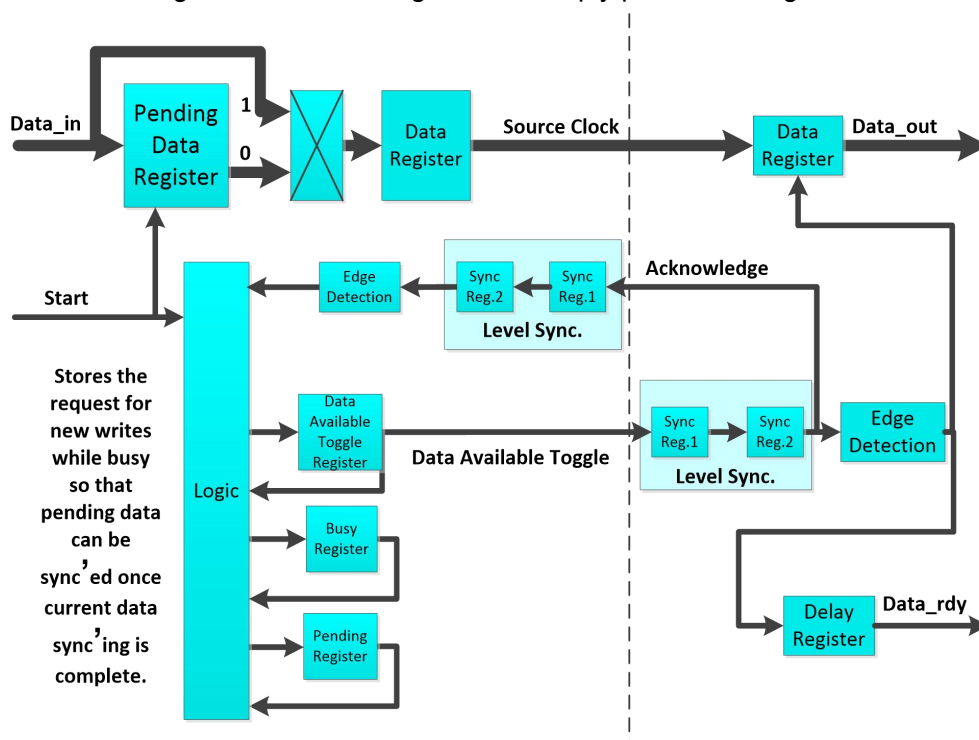


Figure 6.18 RTL Diagram of Data Synchronization Module

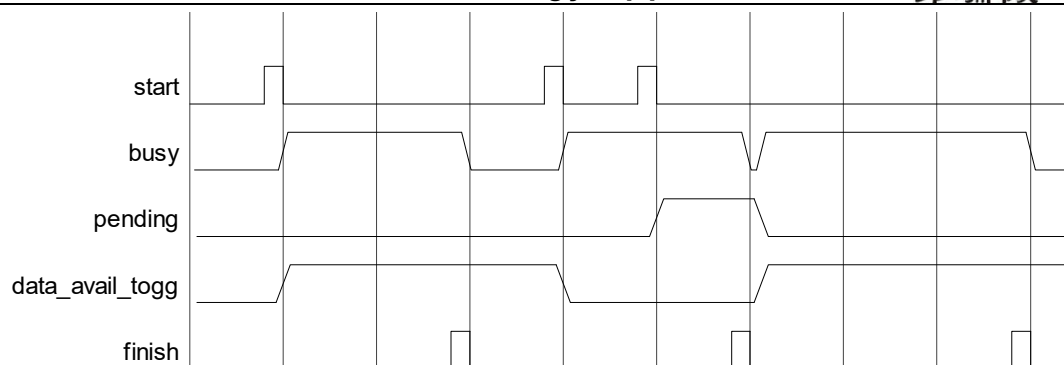


Figure 6.19 Timing Diagram for Data Synchronization Module

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.

6.5.3.6 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:

- Configurable parameters for the following:
 - 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

6.5.3.7 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.

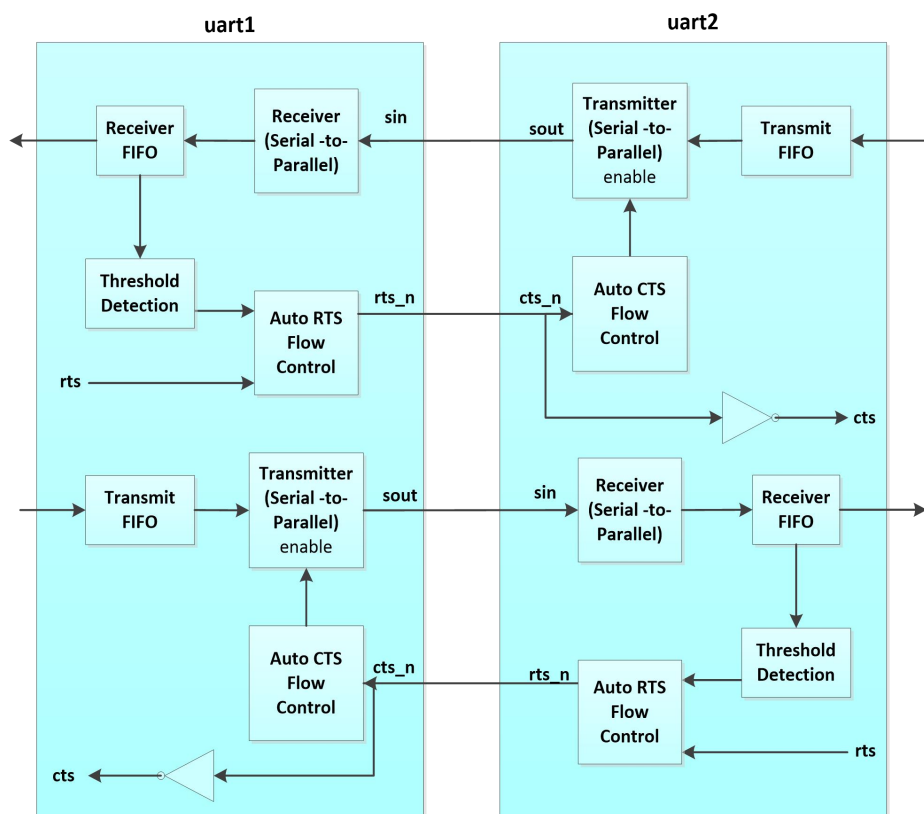


Figure 6.20 Auto Flow Control Block Diagram

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, $\frac{1}{4}$, $\frac{1}{2}$, 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 figure shows a timing diagram of Auto RTS operation.

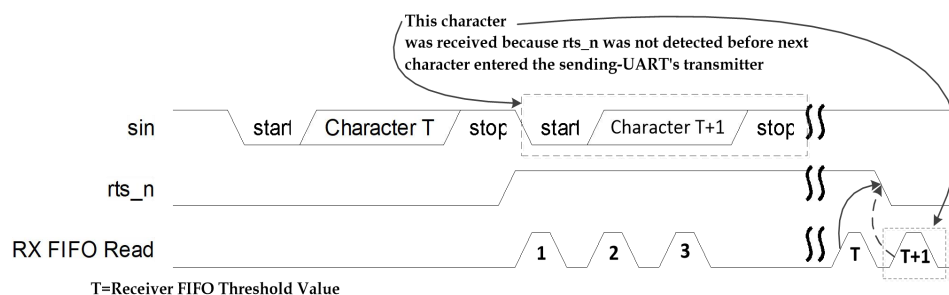


Figure 6.21 Auto RTS Timing

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, or
- 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.

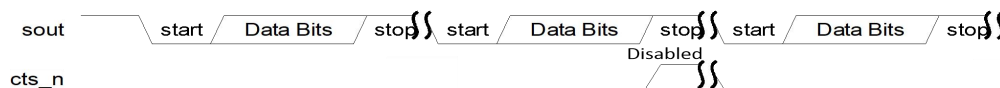


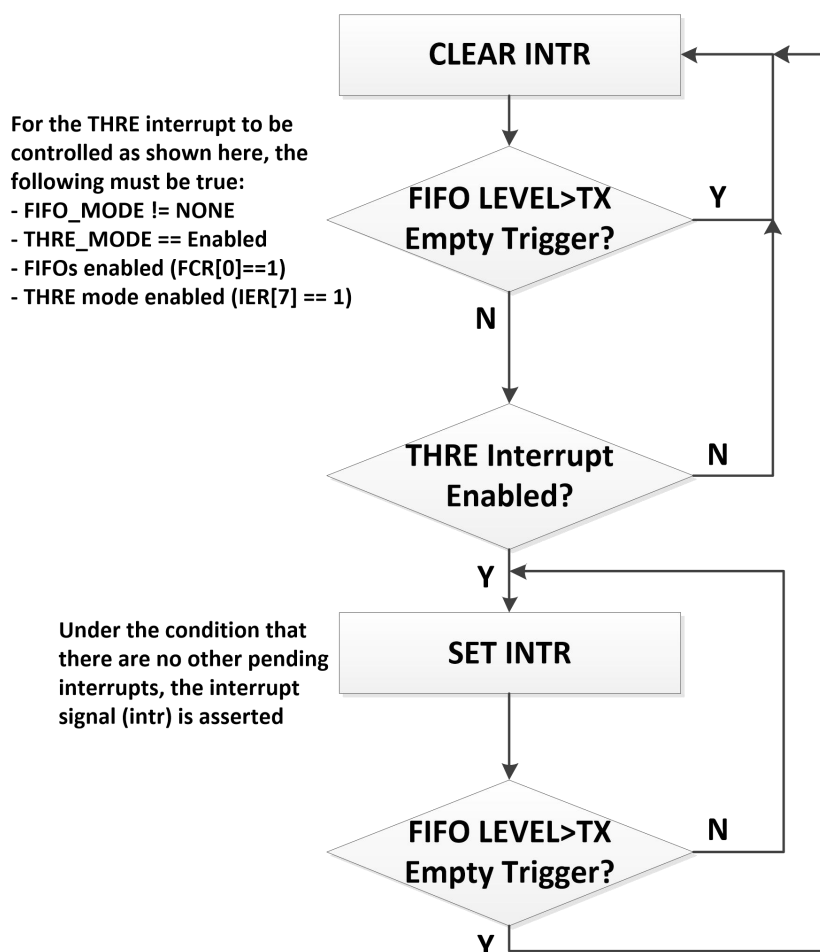
Figure 6.22 Auto CTS Timing

6.5.3.8 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.



**Figure 6.23 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, $\frac{1}{4}$, $\frac{1}{2}$. 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. For threshold setting details, refer to 10.7.4.6 FIFO_CTRL.

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.

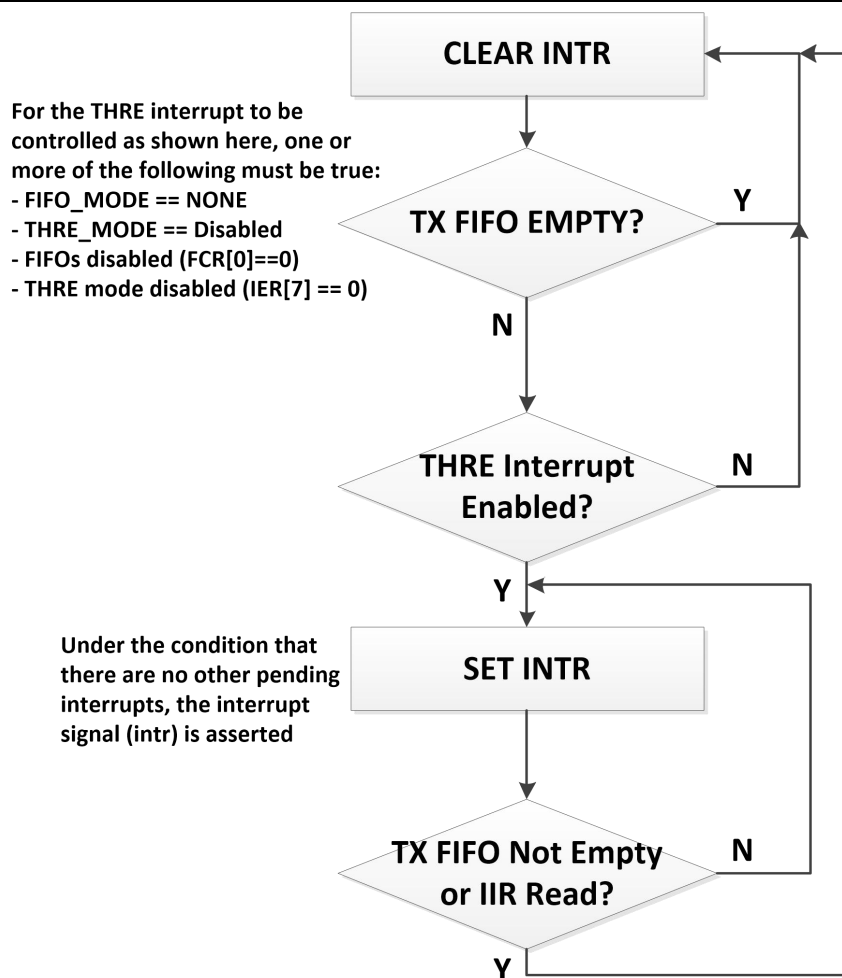


Figure 6.24 Flowchart of Interrupt generation
when not in Programmable THRE Interrupt Mode

6.5.3.9 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 .

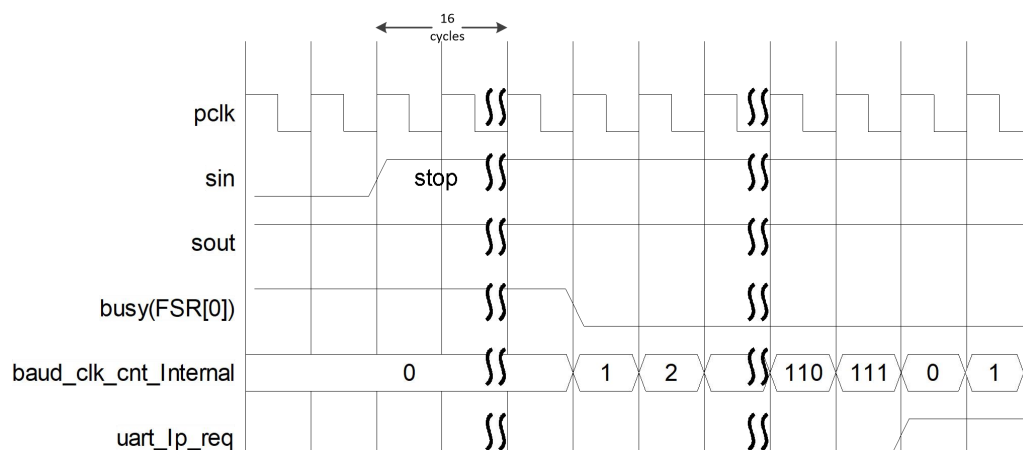


Figure 6.25 Clock Gate Enable Timing

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.

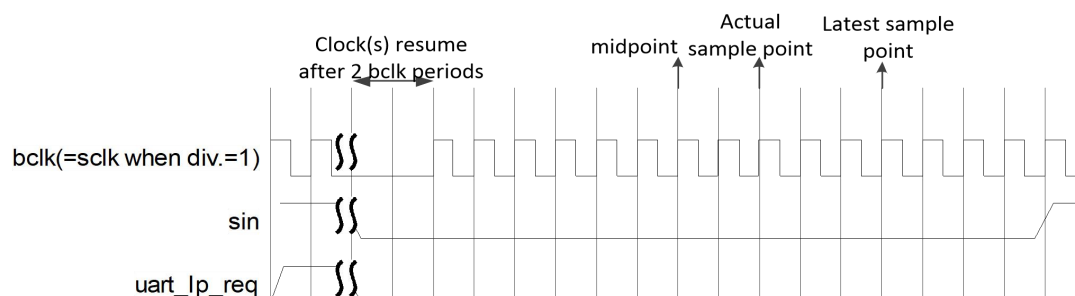


Figure 6.26 Resuming Clock(s) After Low Power Mode Timing

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).

6.5.4 UARTx Register Map

Offset	RW	Reset	Name	Description
0x0000	R	0x0	RBR	Receive Buffer Register Dependencies: LCR[7]bit = 0
	W	0x0	THR	Transmit Holding Register Dependencies: LCR[7]bit = 0
	R/W	0x0	DLL	Divisor Latch (Low) Dependencies: LCR[7] bit = 1
0x0004	R/W	0x0	DLH	Divisor Latch (High) Dependencies: LCR[7] bit = 1
	R/W	0x0	IER	Interrupt Enable Register Dependencies: LCR[7] bit = 0
0x0008	R	0x01	IIR	Interrupt Identification Register
	W	0x0	FCR	FIFO Control Register
0x000C	R/W	0x0	LCR	Line Control Register
0x0010	R/W	0x0	MCR	Modem Control Register
0x0014	R	0x60	LSR	Line Status Register
0x0018	R	0x0	MSR	Modem Status Register

OM6621Dx Bluetooth Low Energy Application

0X001C	R/W	0x0	SCR	Scratchpad Register
0x0020	R/W	0x0	LPDLL	Low Power Divisor Latch (Low) Register
0x0024	R/W	0x0	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	R	0x0	SRBR	Shadow Receive Buffer Register Dependencies: LCR[7] bit = 0
	W	0x0	STHR	Shadow Transmit Holding Register Dependencies: LCR[7] bit = 0
0x0070	R/W	0x0	FAR	FIFO Access Register
0x0074	R	0x0	TFR	Transmit FIFO Read
0x0078	W	0x0	RFW	Receive FIFO Write
0x007C	R	0x6	USR	UART Status Register
0x0080	R	0x0	TFL	Transmit FIFO Level Width: FIFO_ADDR_WIDTH + 1
0x0084	R	0x0	RFL	Receive FIFO Level Width: FIFO_ADDR_WIDTH + 1
0x0088	W	0x0	SRR	Software Reset Register
0x008C	R/W	0x0	SRTS	Shadow Request to Send
0x0090	R/W	0x0	SBCR	Shadow Break Control Register
0x0094	R/W	0x0	SDMAM	Shadow DMA Mode
0x0098	R/W	0x0	SFE	Shadow FIFO Enable
0x009C	R/W	0x0	SRT	Shadow RCVR Trigger
0x00A0	R/W	0x0	STET	Shadow TX Empty Trigger
0x00A4	R/W	0x0	HTX	Halt TX
0x00A8	W	0x0	DMASA	DMA Software Acknowledge
0x00AC-0x00F0				
0x00F4	R	Configuration-dependent	CPR	Component Parameter Register
0x00F8	R	See the Releases table in the AMBA 2 release notes.	UCV	UART Component Version
0x00FC	R	0x44570110	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	<p>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.</p> <p>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.</p> <p>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.</p>

THR address offset: 0x0000

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	W	0x0	THR	<p>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.</p> <p>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.</p> <p>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.</p>

DLH address offset: 0x0004

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	RW	0x0	DLH	<p>Upper 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.</p> <p>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.</p>

DLL address offset: 0x0000

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	RW	0x0	DLL	<p>Lower 8 bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.</p> <p>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.</p>

IER address offset: 0x0004

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

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 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] set to one) the sir_out_n line is continuously pulsed. When in Loop back 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 write able 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 write able 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	<p>Data Length Select</p> <p>This is used to select the number of data bits per character that the peripheral transmits and receives.</p> <p>The number of bit that may be selected areas follows:</p> <p>00 = 5 bits</p> <p>01 = 6 bits</p> <p>10 = 7 bits</p> <p>11 = 8 bits</p>

MCR address offset: 0x0010

Bit	R/W	Reset	Name	Description
31:7	N/A	0x0	N/A	reserved
6	RW	0x0	SIRE	<p>SIR Mode Enable.</p> <p>0 = disable</p> <p>1 = enable</p>
5	RW	0x0	AFCE	<p>Auto Flow Control Enable.</p> <p>0 = Auto Flow Control Mode disabled</p> <p>1 = Auto Flow Control Mode enabled</p>
4	RW	0x0	LB	<p>Loop Back Bit. This is used to put the UART into a diagnostic mode for test purposes.</p> <p>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 loop back 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.</p> <p>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.</p>
3	RW	0x0	OUT2	<p>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:</p> <p>0 = out2_n de-asserted (logic 1)</p> <p>1 = out2_n asserted (logic 0)</p>
2	RW	0x0	OUT1	<p>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</p>

				<p>out1_n, that is:</p> <p>0 = out1_n de-asserted (logic 1)</p> <p>1 = out1_n asserted (logic 0)</p>
1	RW	0x0	RTS	<p>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.</p> <p>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.</p> <p>Note that in Loop back 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.</p>
0	RW	0x0	DTR	<p>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:</p> <p>0 = dtr_n de-asserted (logic 1)</p> <p>1 = dtr_n asserted (logic 0)</p>

LSR address offset: 0x0014

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7	R	0x0	RFE	<p>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.</p> <p>0 = no error in RX FIFO</p> <p>1 = error in RX FIFO</p>
6	R	0x1	TEMT	<p>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</p>

				Transmitter Shift Register are both empty.
5	R	0x1	THRE	<p>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.</p> <p>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.</p>
4	R	0x0	BI	<p>Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data.</p> <p>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.</p> <p>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.</p>
3	R	0x0	FE	<p>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.</p> <p>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</p>

				<p>of the FIFO.</p> <p>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]).</p> <p>0 = no framing error 1 = framing error</p>
2	R	0x0	PE	<p>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.</p> <p>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.</p> <p>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]).</p> <p>0 = no parity error 1 = parity error</p>
1	R	0x0	OE	<p>Overrun error bit. This is used to indicate the occurrence of an overrun error.</p> <p>This occurs if a new data character was received before the previous data was read.</p> <p>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.</p> <p>0 = no overrun error 1 = overrun error</p>
0	R	0x0	DR	<p>Data Ready bit. This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO.</p> <p>0 = no data ready 1 = data ready</p>

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 Loop back 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 DDCCD bit is set when the reset is removed if the dcd_n signal remains asserted.
2	R	0x0	TERI	<p>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.</p> <p>0 = no change on ri_n since last read of MSR 1 = change on ri_n since last read of MSR</p>
1	R	0x0	DDSR	<p>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.</p> <p>0 = no change on dsr_n since last read of MSR 1 = change on dsr_n since last read of MSR</p> <p>Reading the MSR clears the DDSR bit. In Loop back Mode (MCR[4] = 1), DDSR reflects changes on MCR[0] (DTR).</p> <p>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.</p>
0	R	0x0	DCTS	<p>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.</p> <p>0 = no change on cts_n since last read of MSR 1 = change on cts_n since last read of MSR</p> <p>Reading the MSR clears the DCTS bit. In Loop back Mode (MCR[4] = 1), DCTS reflects changes on MCR[1] (RTS).</p> <p>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.</p>

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	<p>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:</p> $\text{Low power baud rate} = (\text{serial clock frequency}) / (16 * \text{divisor})$ <p>Therefore, a divisor must be selected to give a baud rate of 115.2K.</p> <p>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.</p>
-----	----	-----	-------	--

LPDLH address offset: 0x0024

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	RW	0x0	LPDLL	<p>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:</p> $\text{Low power baud rate} = (\text{serial clock frequency}) / (16 * \text{divisor})$

				<div>divisor)</div> <div>Therefore, a divisor must be selected to give a baud rate of 115.2K.</div> <div>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.</div>
--	--	--	--	--

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	<div>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</div> <div>and the THR to be read by the master.</div> <div>0 = FIFO access mode disabled</div> <div>1 = FIFO access mode enabled</div> <div>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.</div>

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	<p>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.</p> <p>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.</p> <p>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.</p>

STHR address offset: 0x0030--0x006C

Bit	R/W	Reset	Name	Description
31:8	N/A	0x0	N/A	reserved
7:0	W	0x0	STHR	<p>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.</p> <p>If in non-FIFO mode or FIFOs are disabled (FCR[0] set to zero) and THRE is set, writing a single</p>

				<p>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.</p> <p>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.</p>
--	--	--	--	---

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	<p>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.</p> <p>0 = FIFO access mode disabled 1 = FIFO access mode enabled</p> <p>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.</p>

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	<p>Transmit FIFO Read. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one).</p> <p>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.</p> <p>When FIFOs are not implemented or not enabled,</p>

				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	RFEE	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 is 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 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 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 Loopback 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	<p>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.</p> <p>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.</p> <p>If SIR_MODE == Enabled and active (MCR[6] = 1) the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver.</p>

SDMAM address offset: 0x0094

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	SDMAM	<p>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).</p> <p>0 = mode 0 1 = mode 1</p>

SFE address offset: 0x0098

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	SFE	<p>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</p>

				(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	<p>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.</p> <p>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:</p> <p>00 = 1 character in the FIFO 01 = FIFO ¼ full 10 = FIFO ½ full 11 = FIFO 2 less than full</p>

STET address offset: 0x00A0

Bit	R/W	Reset	Name	Description
31:2	N/A	0x0	N/A	reserved
1:0	RW	0x0	STET	<p>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.</p> <p>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:</p> <p>00 = FIFO empty 01 = 2 characters in the FIFO 10 = FIFO ¼ full 11 = FIFO ½ full</p> <p>Dependencies: Writes have no effect when THRE_MODE_USER = Disabled.</p>

HTX address offset: 0x00A4

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	RW	0x0	HTX	<p>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.</p> <p>0 = Halt TX disabled 1 = Halt TX enabled</p> <p>Note, if FIFOs are implemented and not enabled, the setting of the halt TX register has no effect on operation.</p> <p>Dependencies: Writes have no effect when FIFO_MODE == None.</p>

DMASA address offset: 0x00A8

Bit	R/W	Reset	Name	Description
31:1	N/A	0x0	N/A	reserved
0	W	0x0	DMASA	<p>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.</p>

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	<p>0x00 = 0 0x01 = 16 0x02 = 32 to 0x80 = 2048 0x81- 0xff = reserved</p>
15:14	R	0x0	N/A	Reserved and read as zero
13	R	0x0	DMA_EXTRA	<p>0 = FALSE 1 = TRUE</p>
12	R	0x0	UART_ADD_ENCODED_PARAMS	<p>0 = FALSE 1 = TRUE</p>
11	R	0x0	SHADOW	<p>0 = FALSE 1 = TRUE</p>
10	R	0x0	FIFO_STAT	<p>0 = FALSE</p>

				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 2 release notes.	UART Component Version	ASCII value for each number in the version, followed by *. For example 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.

6.6 SPIx

6.6.1 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 multi-master 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.

6.6.2 Main Features

- Master or slave operation
- Multi-master 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

6.6.3 Function Description

6.6.3.1 Block Diagram

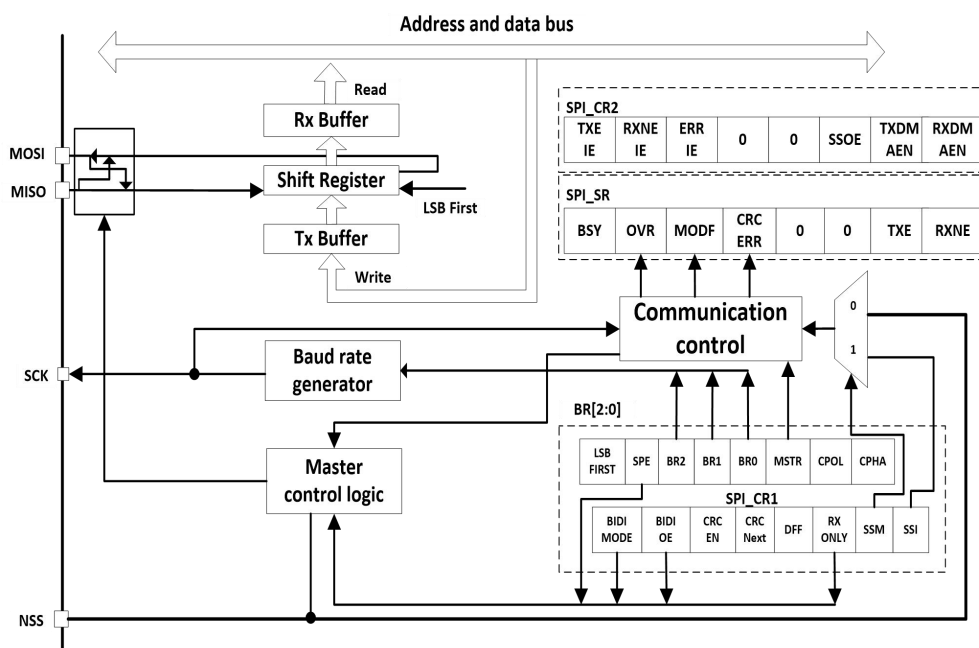


Figure 6.27 SPI block diagram

6.6.3.2 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.

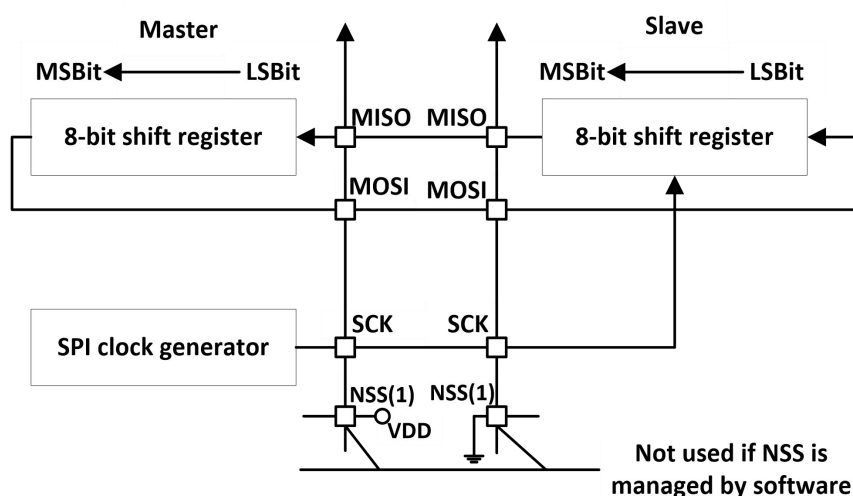


Figure 6.28 Single master/ single slave application

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 multi-master 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.

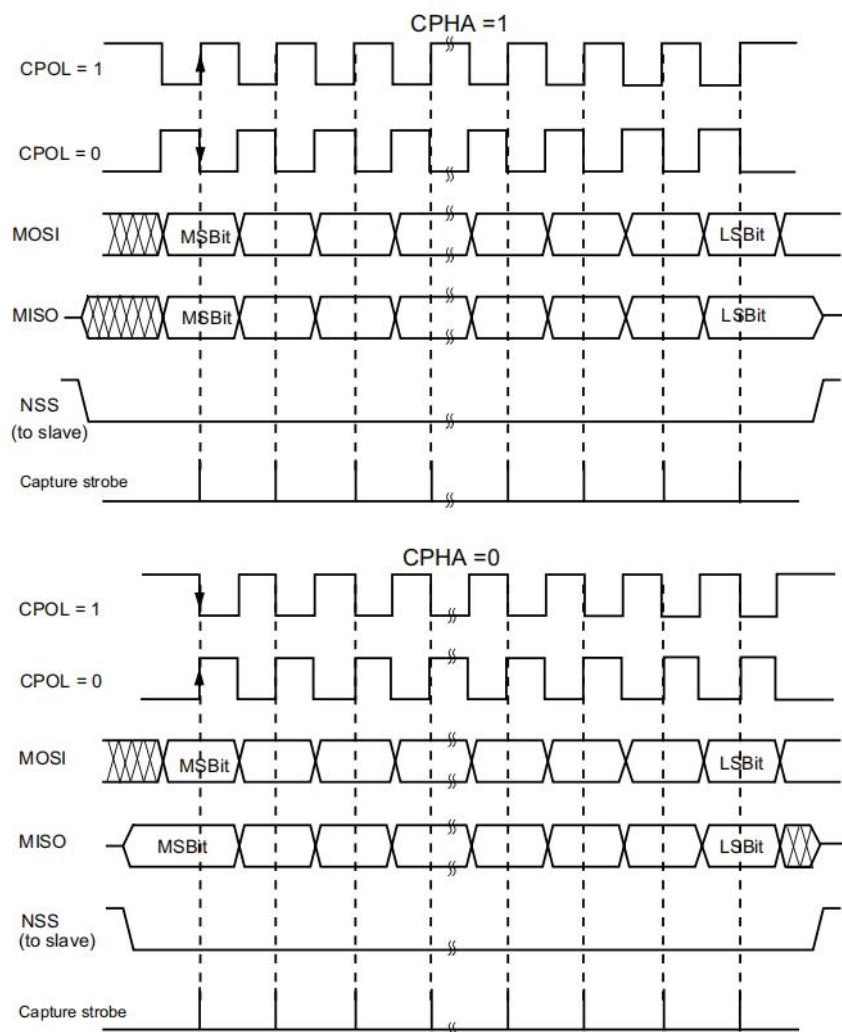


Figure 6.29 Data clock timing diagram

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.

6.6.3.3 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.

6.6.3.4 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 deselected between transmissions, the NSS pin must be configured as GPIO or another GPIO must be used and toggled by software.

6.6.3.5 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.

6.6.3.6 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.

6.6.3.7 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, LSB First, 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 bit rate 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 multi-slave 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 re-synchronize 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).

6.6.3.8 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 multi-master 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.

6.6.3.9 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, BDOE=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).

6.6.3.10 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.

6.6.4 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 IDLE) 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_DMCCR 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

				<p>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.</p> <p>0: Transmit DMA disable 1: Transmit DMA enable</p>
0	R/W	0x0	RDMAE	<p>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.</p> <p>0: Receive DMA disable 1: Receive DMA enable</p>

SPI_DMATDLR address offset: 0x0014

Bit	R/W	Reset	Name	Description
SPI_TX_ABW-1:0	RW	0x0	DMATDLR	<p>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 <= dmatlrl, dma_tx_req will be set 1.</p>

SPI_DMARDLR address offset: 0x0018

Bit	R/W	Reset	Name	Description
SPI_RX_ABW-1:0	RW	0x0	DMARDLR	<p>This register is only valid when the SPI is configured with a set of DMA</p>

				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 <code>rxflr >= dmardlr + 1,dma_rx_req</code> 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

6.7 TIMER

6.7.1 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.

6.7.2 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.
- Up to 2 independent channels for:
 - Input Capture(timer 1 only)
 - Output Compare(all timers)
 - PWM generation (Edge and Center-aligned Mode)
 - One-pulse mode output
- Complementary outputs with programmable dead-time(only one channel of each timer).
- 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.

6.7.3 Function Description

6.7.3.1 Block Diagram

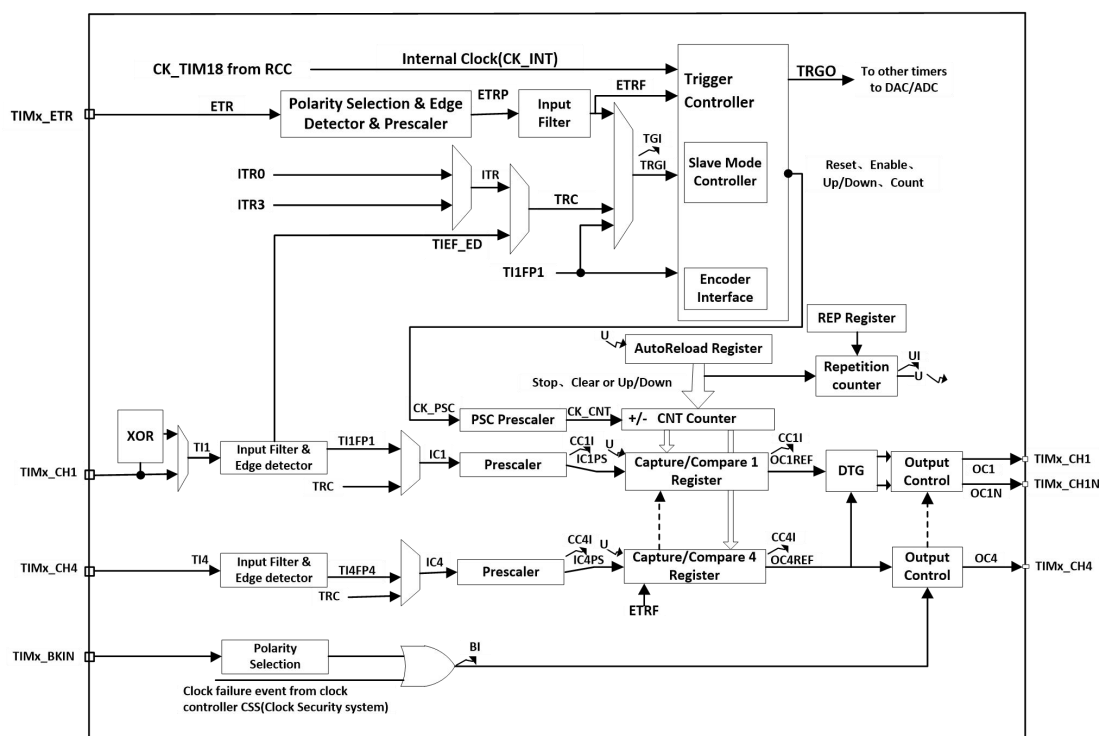




Figure 6.30 General timer block

Note:  Interrupt and DMA output

 Event

 According to the setting of the control bit, the contents of the preload register are transferred to the working register during the U event

6.7.3.2 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.

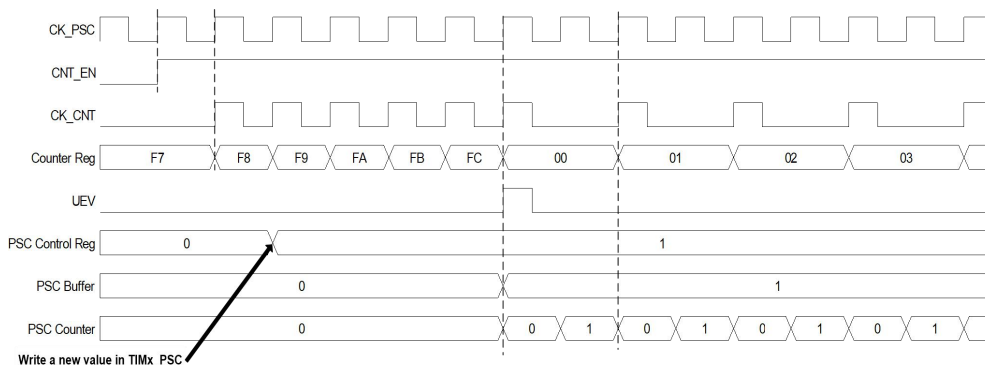
The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set.

Note: That the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

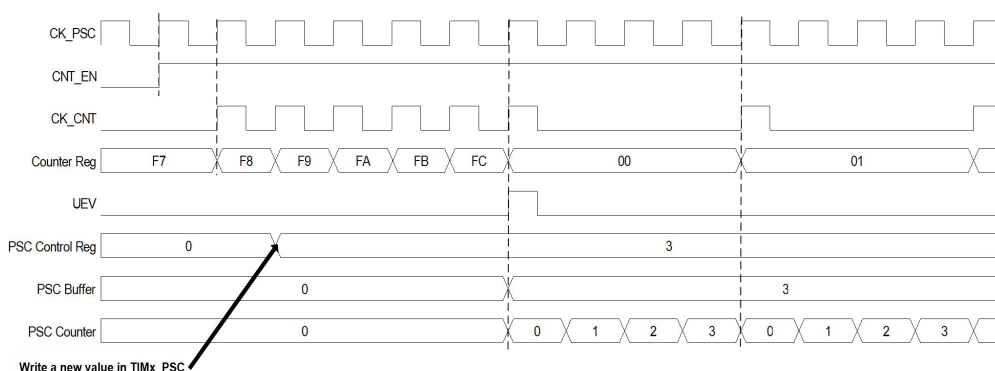
Prescaler description:

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 6.31 Counter timing diagram
with prescaler division change from 1 to 2**



**Figure 6.32 Counter timing diagram
with prescaler division change from 1 to 4**

6.7.3.3 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.

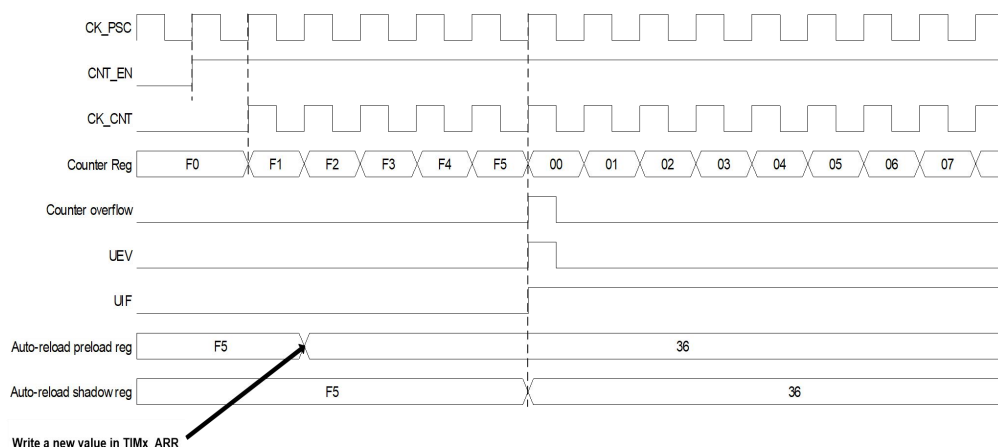


Figure 6.33 Counter timing diagram
update event when ARPE=1 (TIMx_ARR preloaded)

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 prescaler 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.

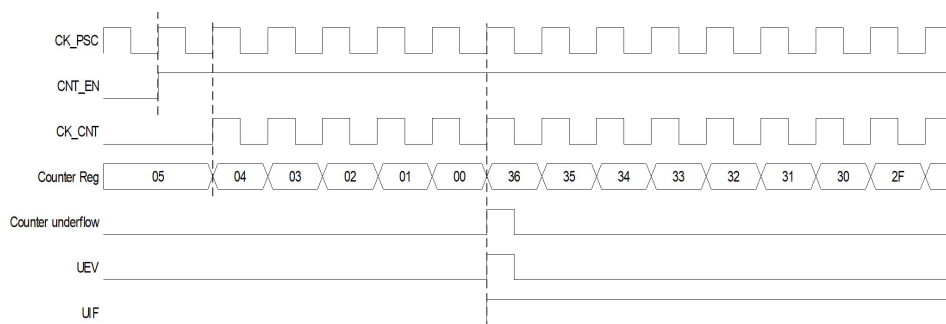


Figure 6.34 Counter timing diagram, internal clock divided by 1

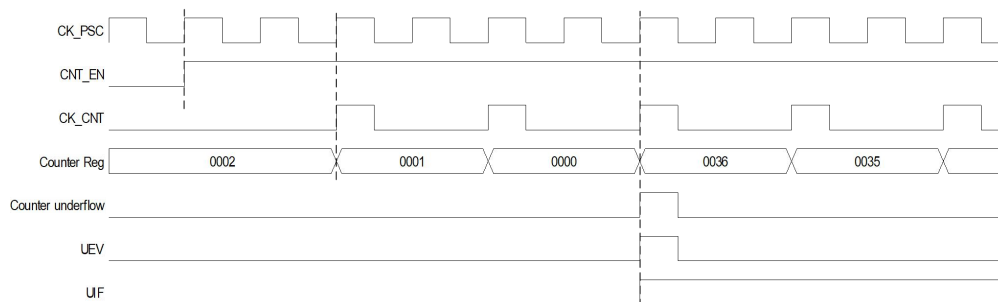


Figure 6.35 Counter timing diagram, internal clock divided by 2

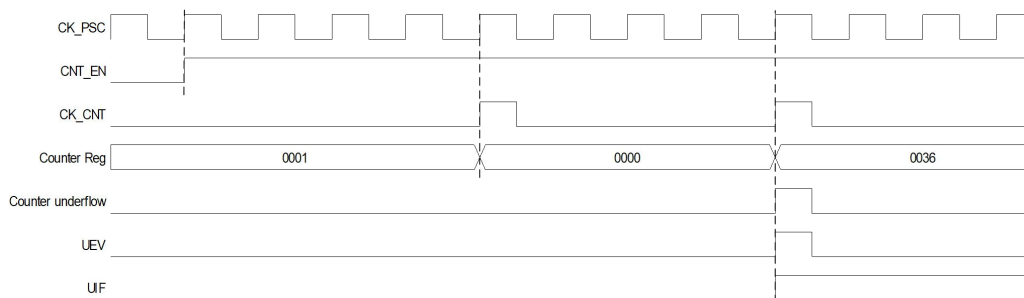


Figure 6.36 Counter timing diagram, internal clock divided by 4

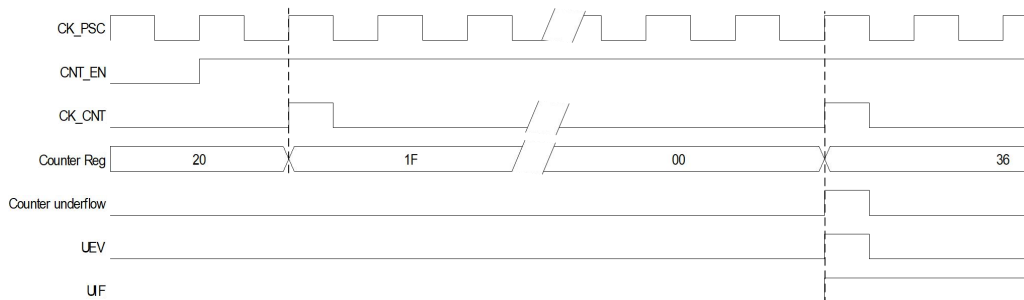
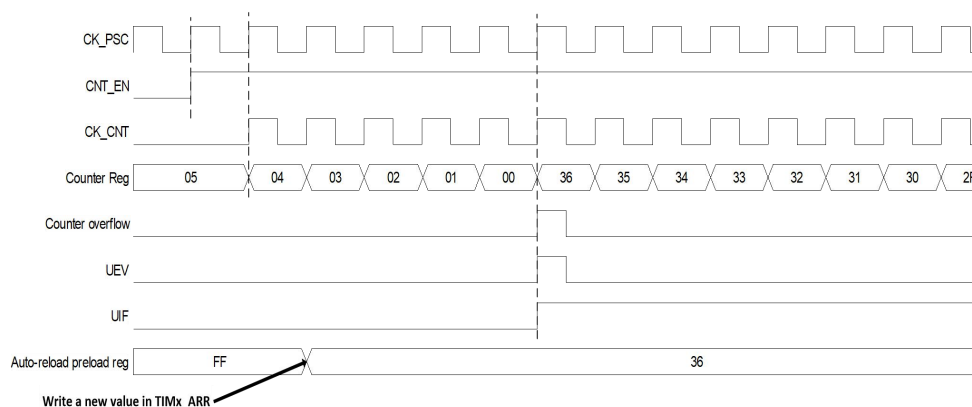


Figure 6.37 Counter timing diagram, internal clock divided by N



**Figure 6.38 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.

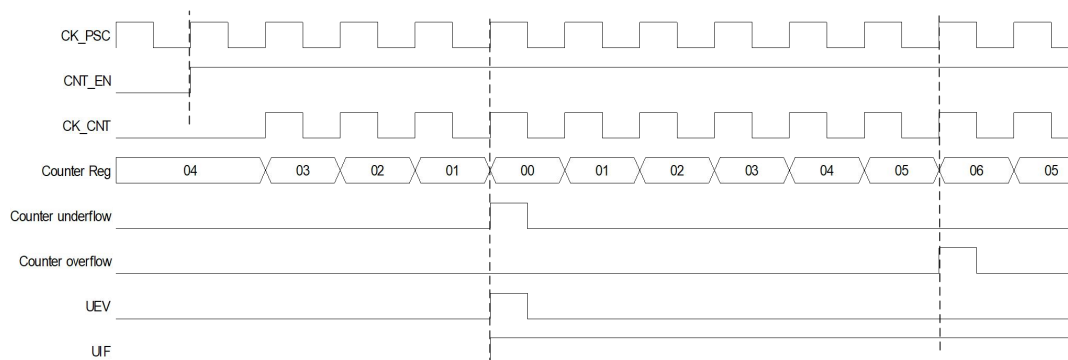
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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):

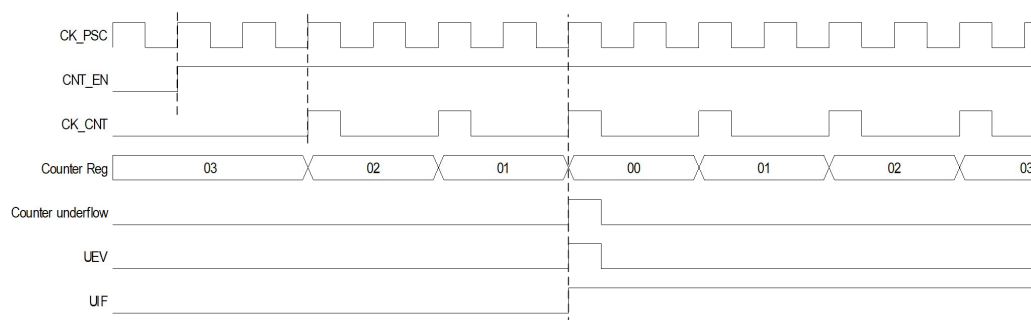
- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the

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).

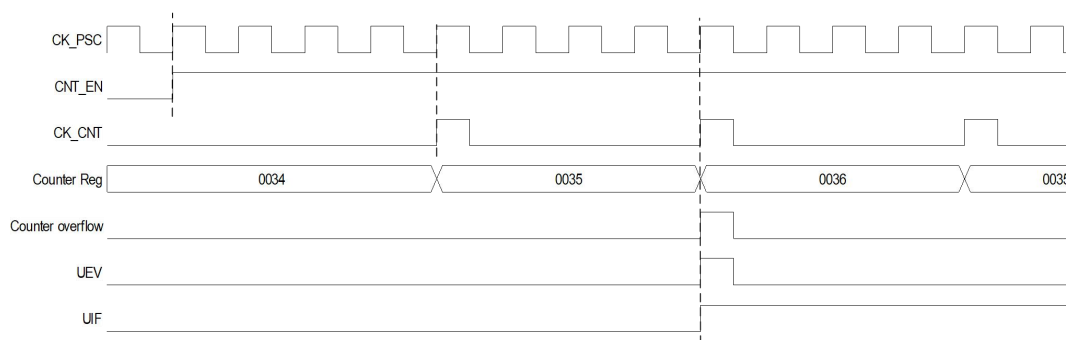
The following figures show some examples of the counter behavior for different clock frequencies.



**Figure 6.39 Counter timing diagram,
internal clock frequency division factor is 1, TIMx_ARR=0x6**



**Figure 6.40 Counter sequence diagram,
internal clock frequency division factor is 2**



**Figure 6.41 Counter timing diagram,
internal clock frequency division factor of 4, TIMx_ARR=0 x36**

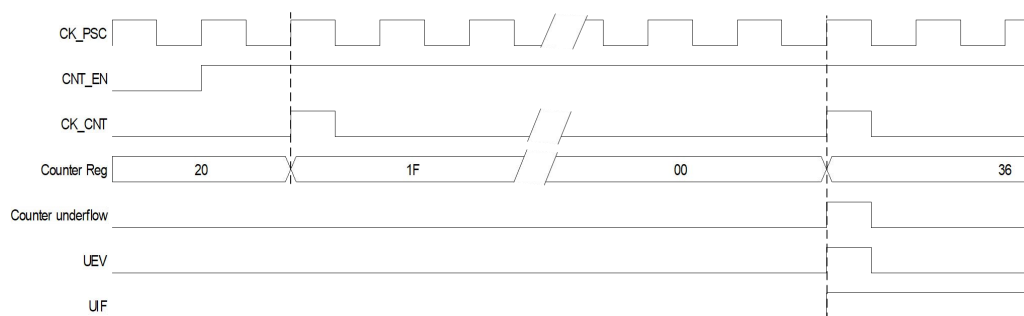
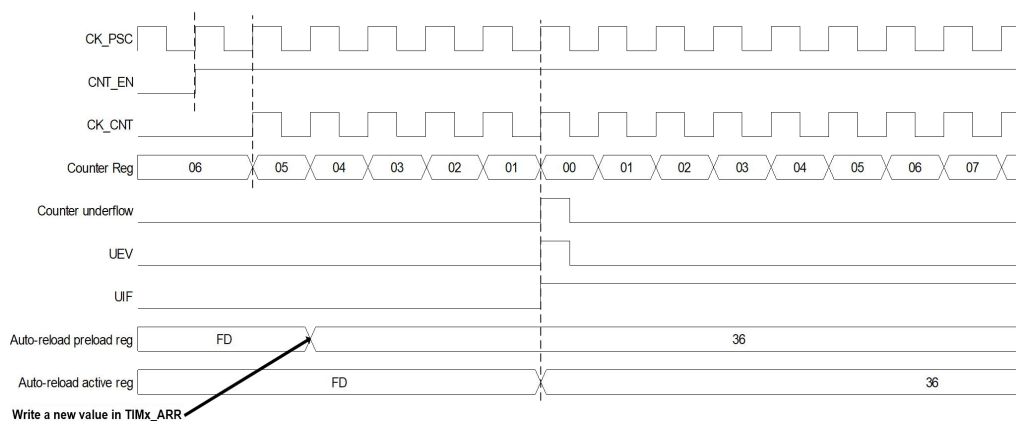
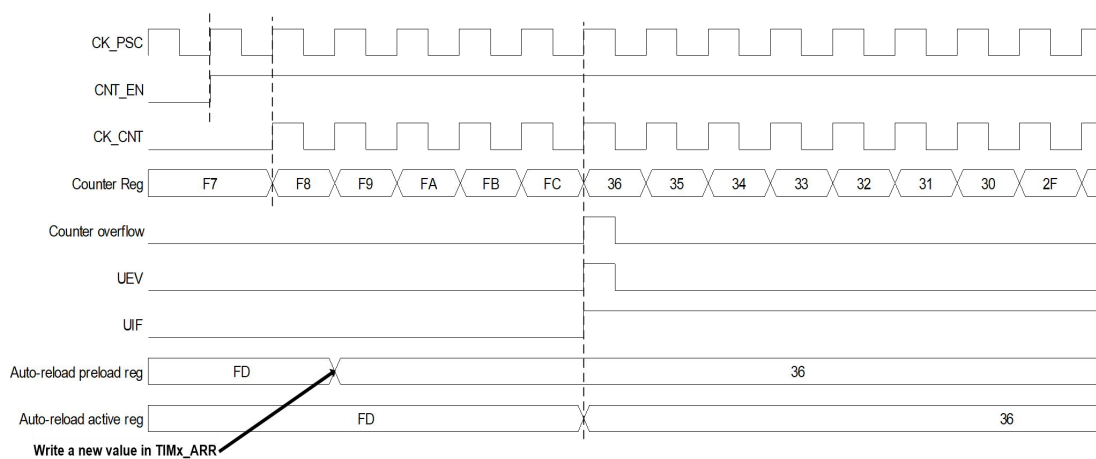


Figure 6.42 Counter timing diagram, internal clock divided by N



**Figure 6.43 Counter timing diagram,
update event at ARPE=1(counter underflow)**



**Figure 6.44 Counter timing diagram,
update event at ARPE=1(counter overflow)**

6.7.3.4 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 $2xT_{ck}$, 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.

6.7.3.5 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. Refer to Using one timer as prescaler for another for more details.

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.

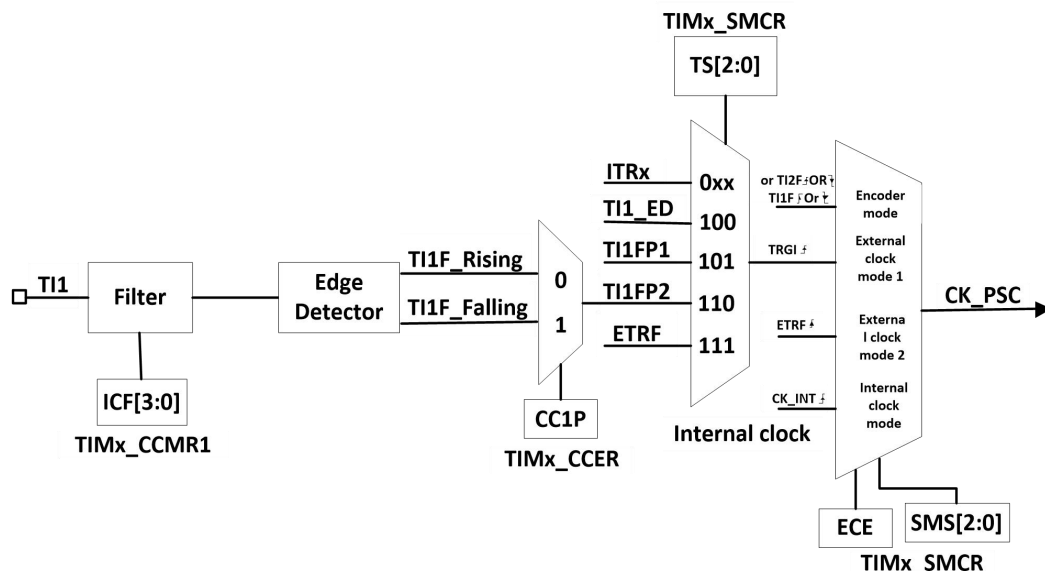


Figure 6.45 TI1 external clock connection example

For example, to configure the upcounter to count in response to a rising edge on the TI1 input, use the following procedure:

- Configure channel 1 to detect rising edges on the TI1 input by writing CC1S = '01' in the TIMx_CCMR1 register.
- Configure the input filter duration by writing the IC1F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC1F=0000).
- Select rising edge polarity by writing CC1P=0 and CC1NP=0 in the TIMx_CCER register.
- Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
- Select TI1 as the trigger input source by writing TS=101 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 TI1, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI1 and the actual clock of the counter is due to the resynchronization circuit on TI1 input.

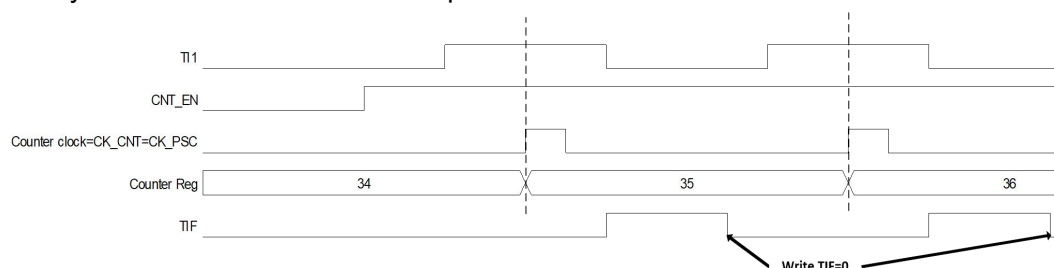


Figure 6.46 Control circuit in external clock mode 1

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.

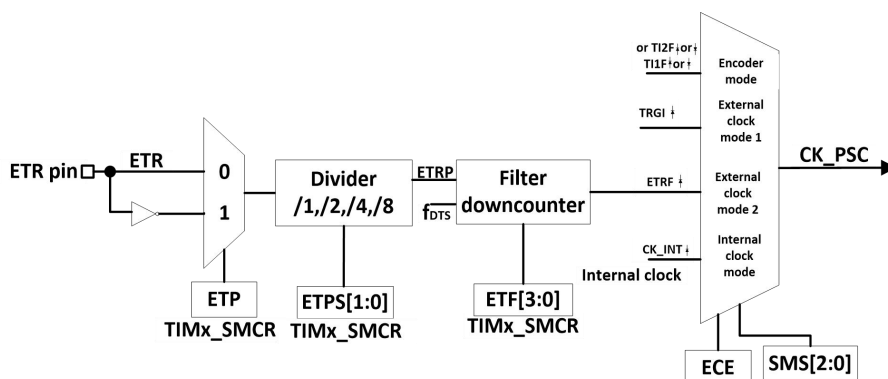


Figure 6.47 External trigger input block

For example, to configure the up-counter 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 ETSP[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 re-synchronization circuit on the ETRP signal.

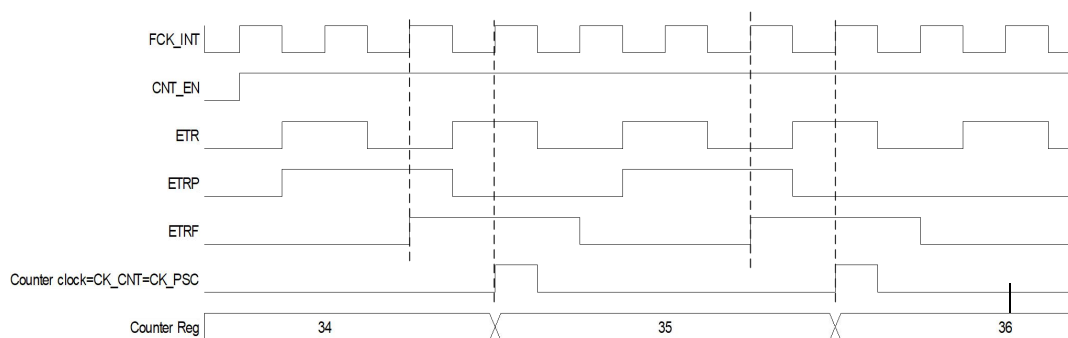


Figure 6.48 Control circuit in external clock mode 2

6.7.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), the input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives 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).

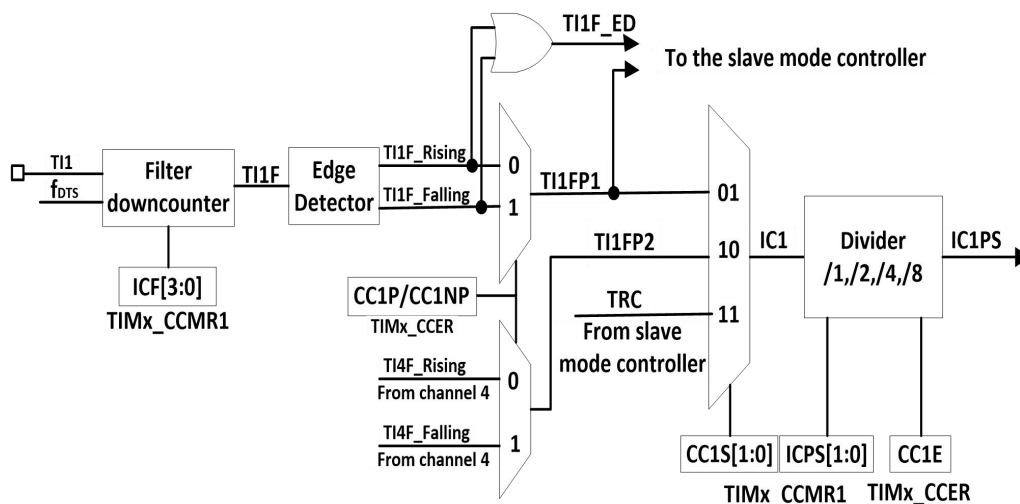


Figure 6.49 Capture/compare channel
(example: channel 1 input stage)

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.

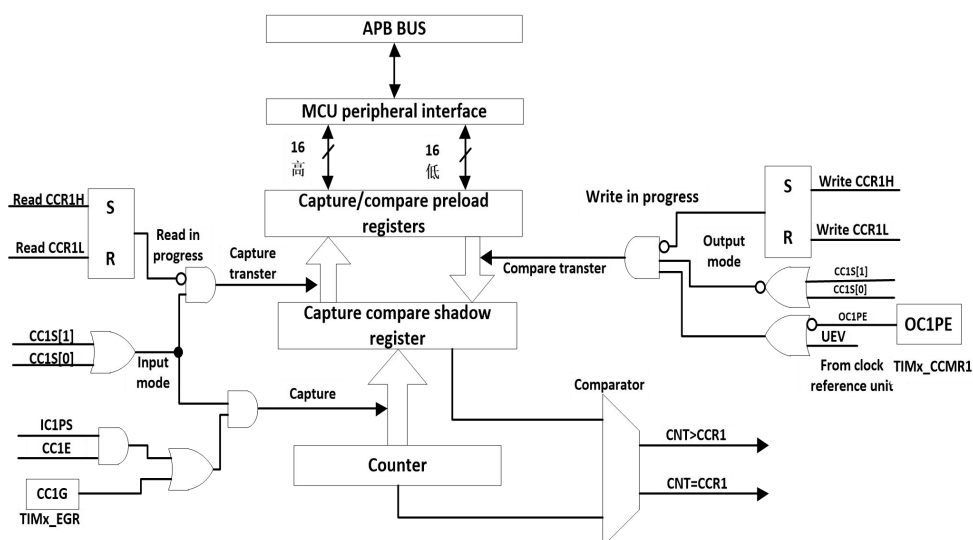


Figure 6.50 Main circuit of capture/compare channel 1

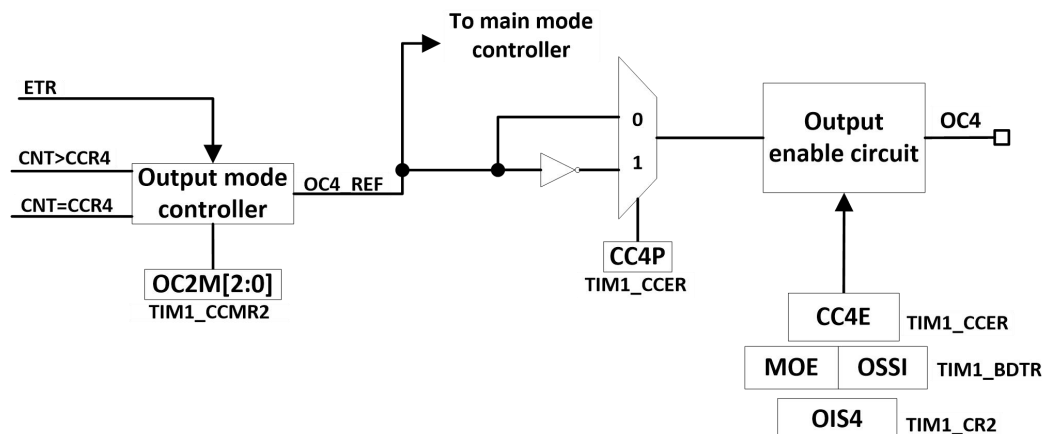


Figure 6.51 Capture/compare the output portion of the channel (channel 4)

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.

6.7.3.7 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.

6.7.3.8 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_CCR4 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 for both 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_CCR4: write the CC4S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP4 (used for capture in TIMx_CCR4): write the CC4P and CC4NP 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 CC4E bits to '1' in the TIMx_CCER register.

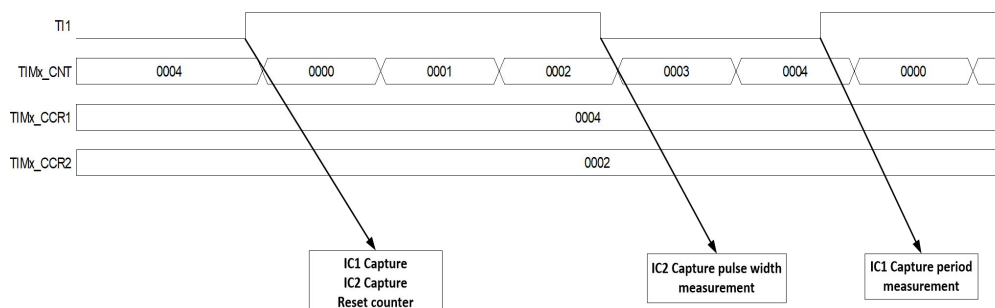


Figure 6.52 PWM Input Mode Timing

6.7.3.9 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.

6.7.3.10 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, called 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). An example is given in the following figure.

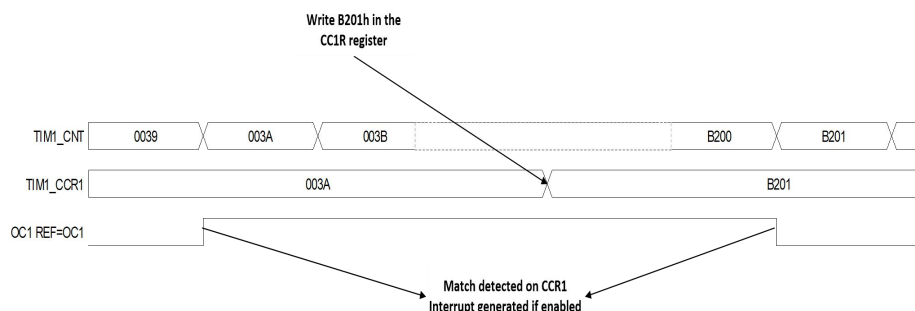


Figure 6.53 Output compare mode, toggle on OC1

6.7.3.11 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 $TIMx_CCRx \leq TIMx_CNT$ or $TIMx_CNT \leq TIMx_CCRx$ (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.

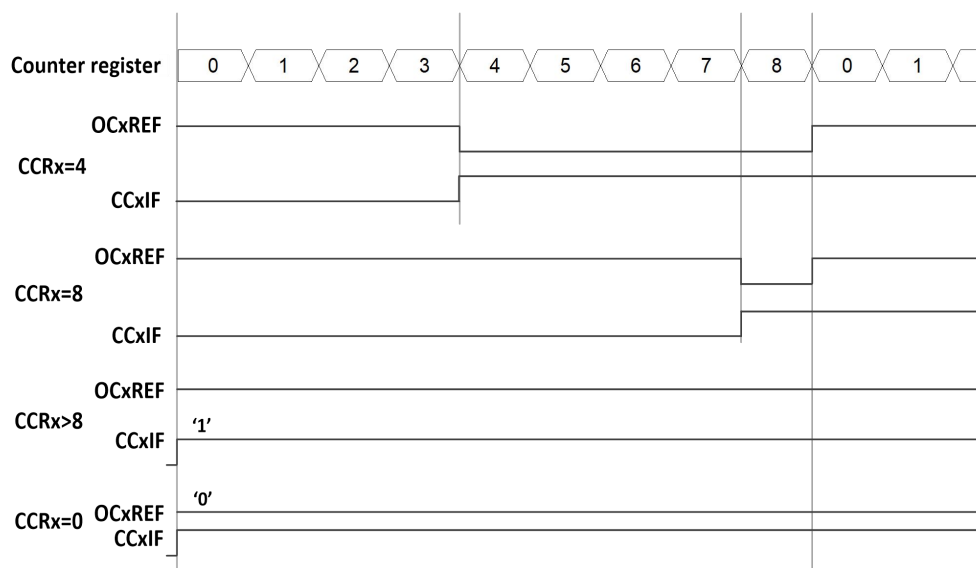


Figure 6.54 Edge-aligned PWM waveforms (ARR=8)

- 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.

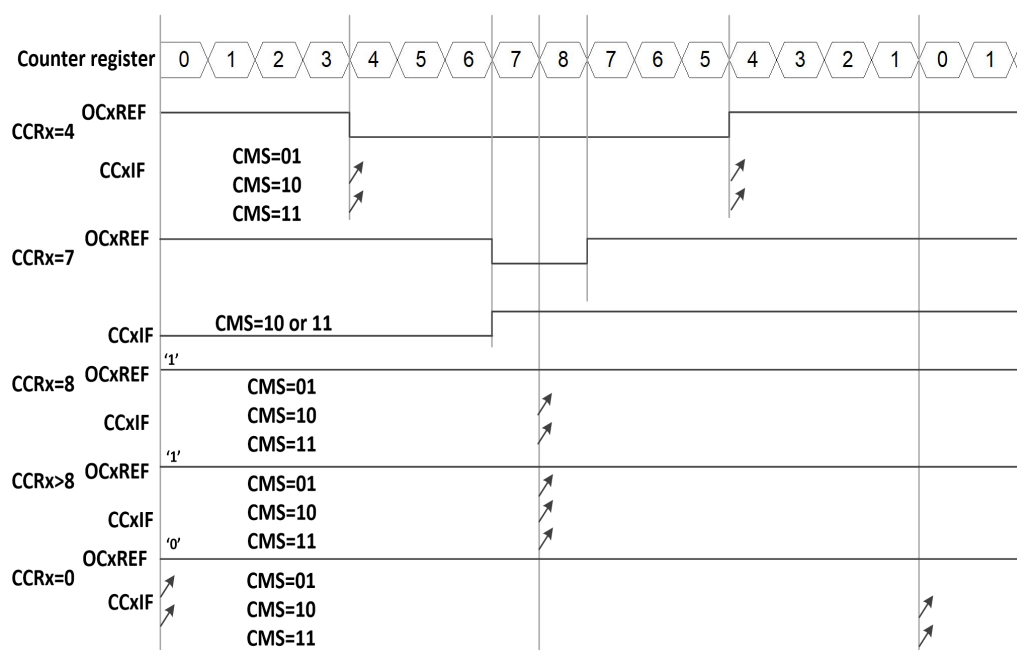


Figure 6.55 Center-aligned PWM waveforms (ARR=8)

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 running 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.

6.7.3.12 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, OSS1 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).

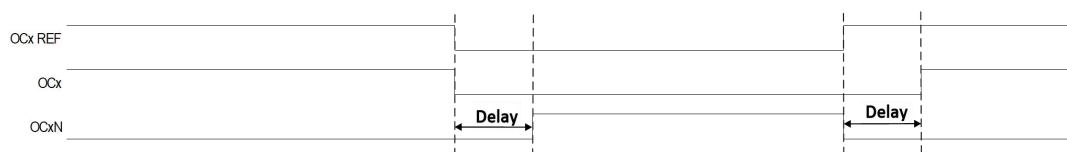


Figure 6.56 Complementary output with dead-time insertion

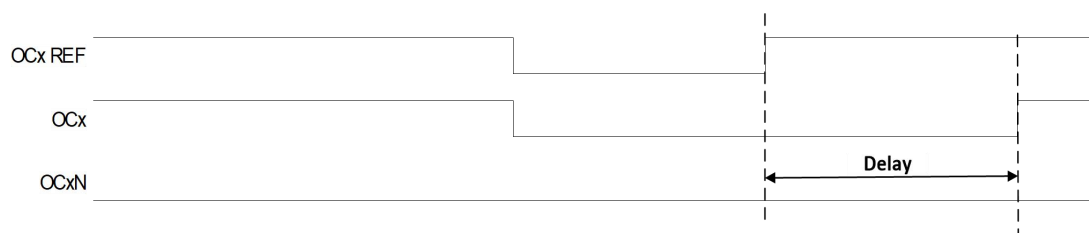


Figure 6.57 Dead-time waveforms with delay greater than the negative pulse

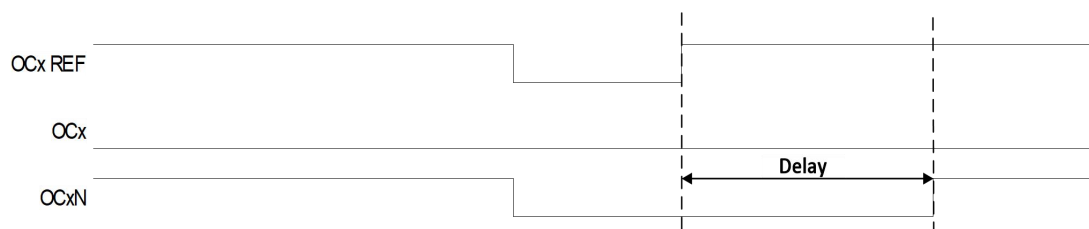


Figure 6.58 Dead-time waveforms with delay greater than the positive pulse

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.

6.7.3.13 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 OSS1 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 OSS1=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 OSS1=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.



- 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.

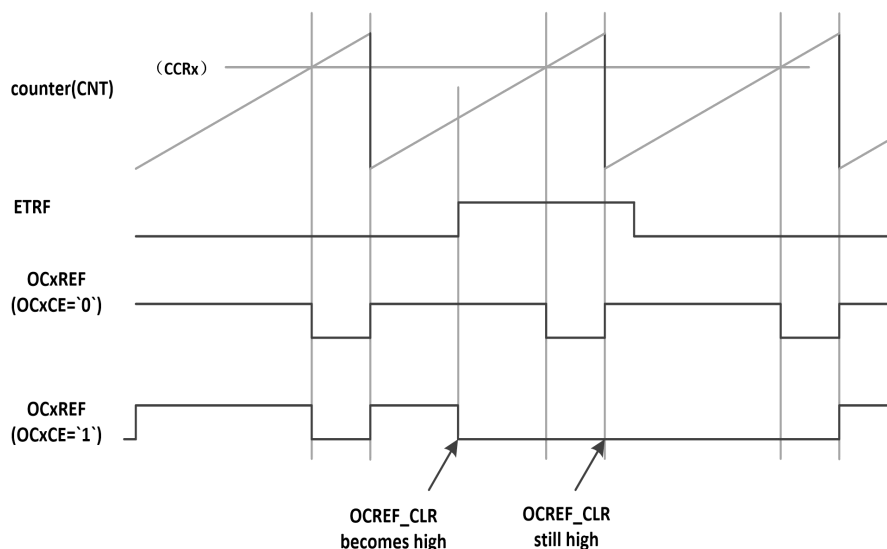


Figure 6.60 Clearing TIMx OCxREF

Note: In case of a PWM with a 100% duty cycle (if $CCR_x > ARR$), then OCxREF is enabled again at the next counter overflow.

6.7.3.15 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.

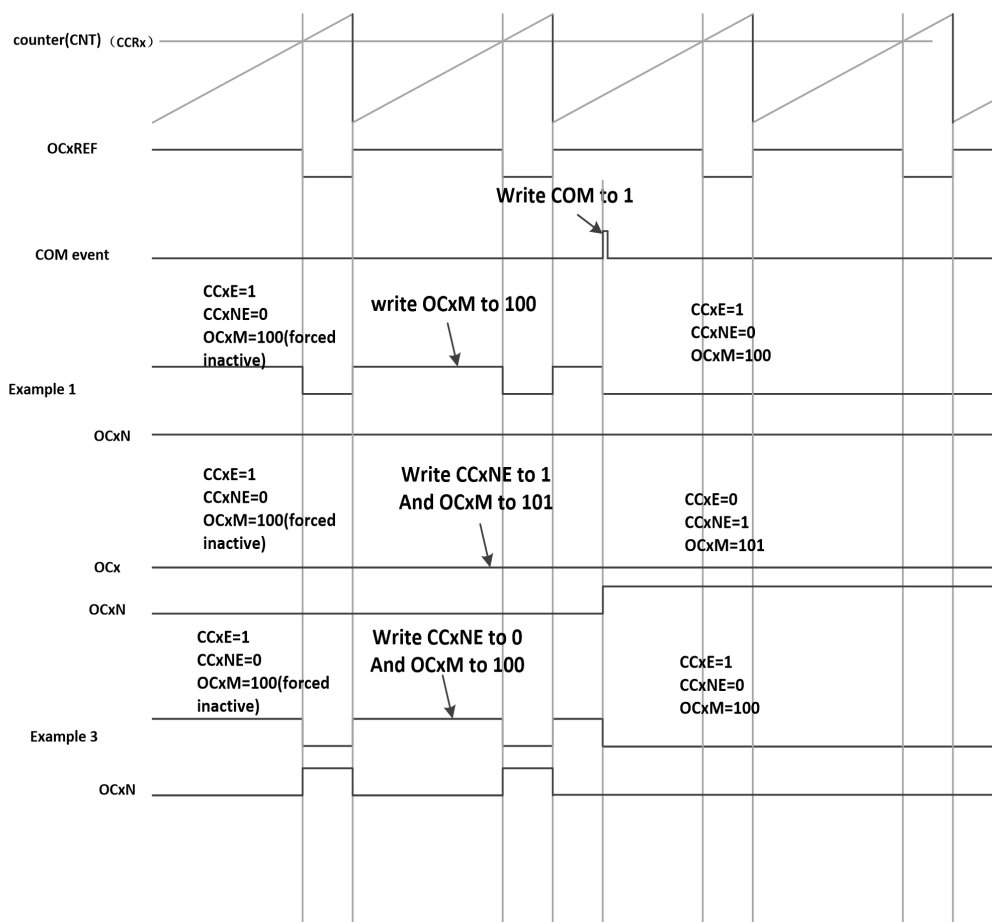


Figure 6.61 step generation, COM example (OSSR=1)

6.7.3.16 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: $CNT < CCRx \leq ARR$ (in particular, $0 < CCRx$)
- In downcounting: $CNT > CCRx$

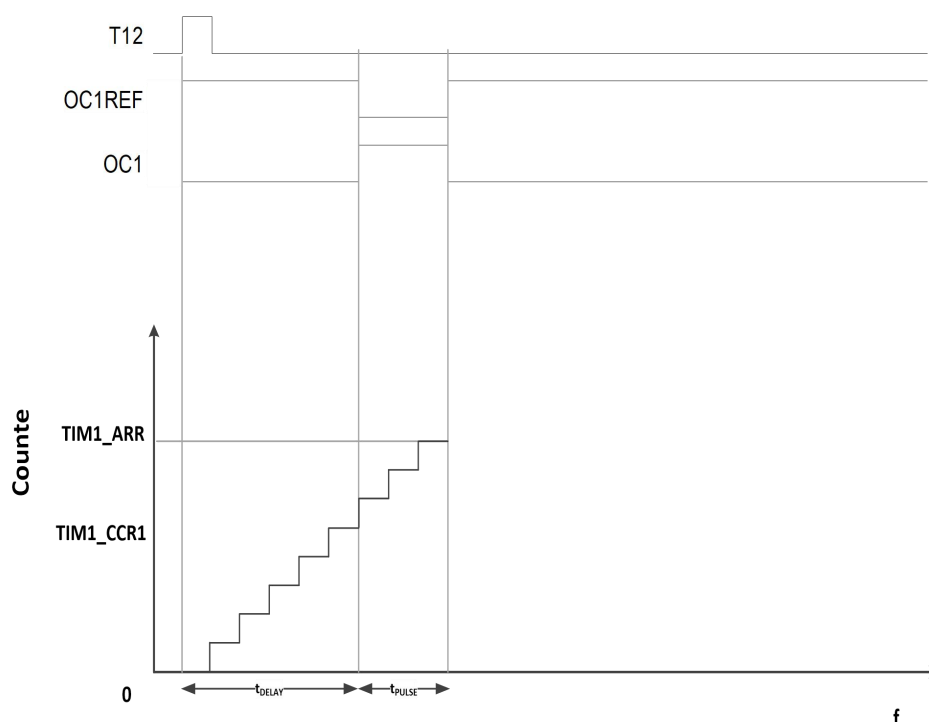


Figure 6.62 Example of one pulse mode

For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI4 input pin. Let's use TI4FP4 as trigger 1:

- Map TI4FP4 to TI4 by writing CC4S='01' in the TIMx_CCMR2 register.
- TI4FP4 must detect a rising edge, write CC4P='0' and CC4NP='0' in the TIMx_CCER register.
- Configure TI4FP4 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
- TI4FP4 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 t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} 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 TI4. 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 $t_{\text{DELAY min}}$ 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.

6.7.3.17 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.

The XOR output can be used with all the timer input functions such as trigger or input capture.

6.7.3.18 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.

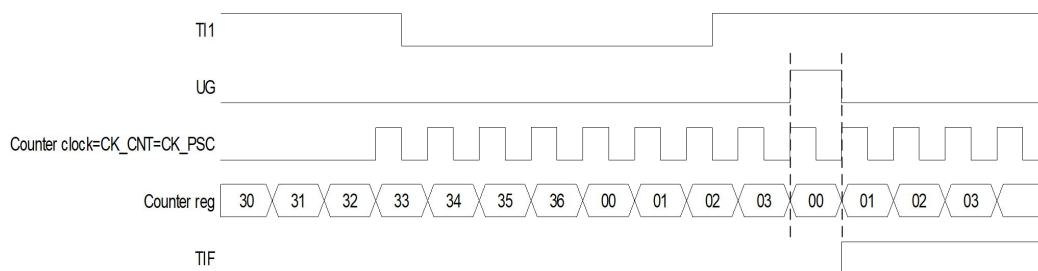


Figure 6.63 Control circuit in reset mode

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.

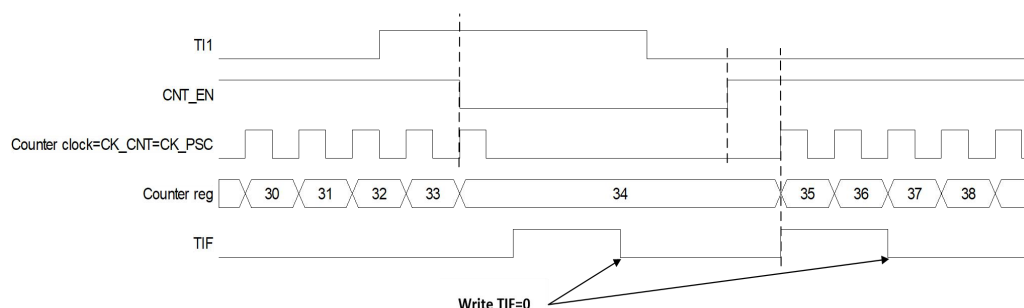


Figure 6.64 Control circuit in gated mode

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 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 are configured to 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 trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.

When a rising edge occurs on TI1, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI1 and the actual start of the counter is due to the resynchronization circuit on TI1 input.

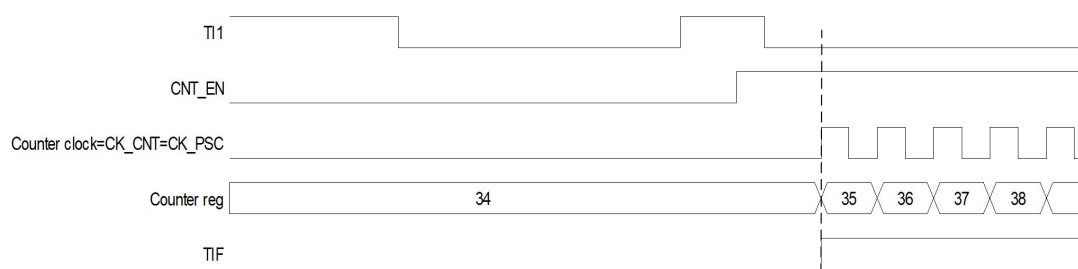


Figure 6.65 Control circuit in trigger mode

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 TI1:

- IC1F=0000: no filter.
- The capture prescaler is not used for triggering and does not need to be configured.
- CC1S=01 in 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.

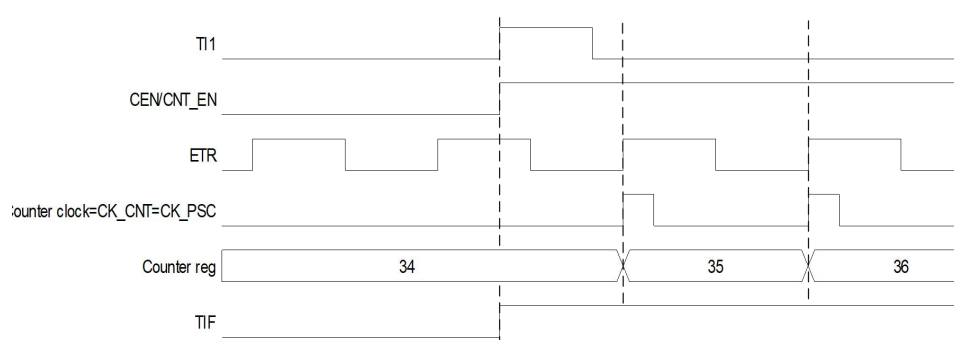


Figure 6.66 Control circuit in external clock mode2 + trigger mode

6.7.4 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

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
13:10	N/A	0x0	N/A	reserved
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	N/A	0x0	N/A	reserved
10	N/A	0x0	N/A	reserved
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	N/A	0x0	N/A	reserved
2	N/A	0x0	N/A	reserved
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 over capture flag
11	N/A	0x0	N/A	reserved
10	N/A	0x0	N/A	reserved
9	RW	0x0	CC1OF	Capture/Compare 1 over capture 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	N/A	0x0	N/A	reserved
2	N/A	0x0	N/A	reserved
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	N/A	0x0	N/A	reserved
2	N/A	0x0	N/A	reserved
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
31:8	N/A	0x0	N/A	reserved
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
31:8	N/A	0x0	N/A	reserved
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
7:0	N/A	0x0	N/A	reserved

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
1:0	N/A	0x0	N/A	reserved

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
11:4	N/A	0x0	N/A	reserved
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_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

6.8 OTP

6.8.1 Introduction

The OTP controller realizes all functions of the OTP macro cell in an automated and transparent way.

The controller facilitates all data transfers (reading and programming) and test items with the AHB bus master control.

6.8.2 Main Features

- Compliant with AMBA™ 2 AHB protocol specification
- Automatic single Error Code Correction (ECC) - 6 bits (implemented in the OTP cell)
- 32-bit read in a single read access from the OTP cell
- Single word buffer for programming. No burst programming supported
- Empty words are 0xFFFFFFFF. Zeros are programmed per 32-bit word
- Transparent random address access to the OTP memory cells via the AHB slave memory
- otp address:otp mem from 0x60000000~0x60003ffc;otp register from 0x60004000~0x60004030
- access speed: fast read speed is 16mhz*32bit;word program speed from 200us(all 1 data) to 1100us(all 0 data);

6.8.3 Function Description

6.8.3.1 Block Diagram

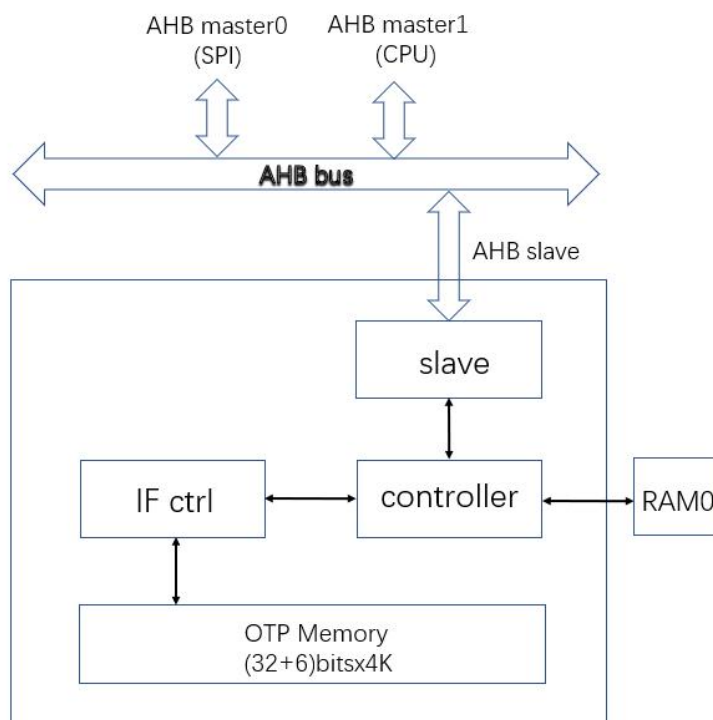


Figure 6.67 OTP controller block diagram

6.8.3.2 Operation flow

Block write flow:

Block write operation will read a block data from RAM1 and program these data to OTP mem, before Block write operation is execute, make sure write_en bit is set and test_row_en=0;

- **Soft:**
 - read status register OTP_READY bit, make sure OTP control is in READY state;
 - set ram_r_baddr and ram_r_length bit in **RAM Config register**;
 - set **OTP Write Address register**;
 - set **Write Enable register**=32'h1133_5577 to enable OTP write authority;
 - write 16'h5a5a to **Command register**;
 - set **Execute register**=1 to execute block write command;
- **Hardware:**
 - After receive block write command, hardware will read data from RAM1 and write it into OTP memory, after all data be written, OTP controller will back to ready state again;
- **Soft:**
 - Read OTP STATUS register until otp_ready bit is 1 again;
 - set **Write Enable register** =32'haabb_ccdd to disable OTP write authority;

case illustration for software

In this case, 2 words (8 bytes) in RAM1 byte address 0 to RAM1 byte address 7 (byte address) will be programmed into OTP mem word address 0 and OTP address word address 1 location:

- read OTP_READY bit until OTP_READY bit=1;
- set ram_r_baddr=0 and ram_r_length=8;
- set OTP Write Address register =d'4091;
- set Write Enable register=32'h1133_5577;
- write 16'h5a5a to Command register;
- set Execute register=1;
- read OTP_READY bit until OTP_READY bit=1;
- set Write Enable register =32'haabb_ccdd;

Word write flow:

Word write operation will program one word (4 bytes) into OTP mem at a time; In this case, one words will be programmed into OTP mem word address 1 location (byte address is 4~7)

- **Soft:**
 - read status register OTP_READY bit, make sure OTP control is in READY state;
 - set **OTP Write Address register=0x4;**
 - set **OTP Write Data register;**
 - set **Write Enable register=32'h1133_5577** to enable OTP write authority;
 - write 16'h0123 to **Command register;**
 - set **Execute register=1** to execute word write command;
- **Hardware:**
 - After receive word write command, hardware will program the data into OTP memory, and then OTP controller will back to ready state again;
- **Soft:**
 - Read OTP STATUS register until otp_ready bit is 1 again;
 - set **Write Enable register =32'haabb_ccdd** to disable OTP write authority;

case illustration for software

In this case, 4 bytes data 0x0000aa55 will be programmed into OTP mem word address 1 location (byte address is 0x4~0x7):

- read OTP_READY bit until OTP_READY bit=1;
- set OTP Write Wdata register =0x0000aa55;
- set OTP Write Address register =0x4;
- set Write Enable register=32'h1133_5577;
- write 16'h01123 to Command register;
- set Execute register=1;
- read OTP_READY bit until OTP_READY bit=1;
- set Write Enable register =32'haabb_ccdd;

Transparent read flow

Transparent read function use a ahb slave interface, which use for read the contents of the OTP memory and is read-only;

This ahb slave interface supports the following all burst types in ahb protocol witch include: SINGLE INCR INCR4 INCR8 INCR16 WRAP4 WRAP8 WRAP16;

SINGLE read supports the hsize of byte halfword and word;

BURST read only support the hsize of word;

initial margin read FF test(for main array)

initial margin read FF test is for check if all 4k words main array is all 1 in CP/FT stage;

- **Soft:**
 - read status register OTP_READY bit,make sure OTP control is in READY state;
 - set **OTP_CFG register RMODE** bits;
 - write 16'h4567 to **Command register**;
 - set **Execute register=1** to execute word write command;
- **Hardware:**
 - After receive M_CHECK command,OTP controller will read all 4k words in main array and check if all data is 1 automatic;after check all data is 1 or read a error data this process will finish,OTP controller back to read ready status;
- **Soft:**
 - Read OTP STATUS register until otp_ready bit is 1 again;
 - read bit TEST_RESULT;if TEST_RESULT =0 indicate test is pass,or test is fail;

OTP mem state:

Consider reduce OTP mem power consumption,software can control OTP mem transfer in four power states with **Command register**; The following figure illustrates the commands that need to be set for state transition,invalid command will be ignored by OTP controller;

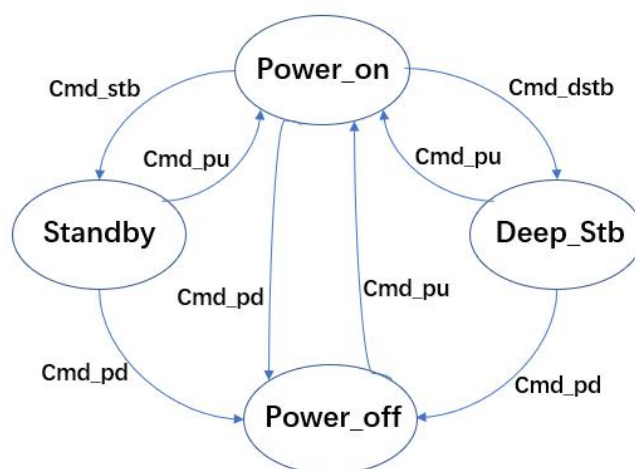


Figure 6.68 OTP mem state transfer block

6.8.4 OTP Register Map

Offset	Name	Description
0x0000	otp_config	OTP Config Register
0x0004	otp_command	OTP Command Control Register
0x0008	otp_w_protect	OTP Write Enable Register
0x000c	otp_trow_en	OTP Test_row_en Register
0x0010	otp_execute	OTP Command Execute register
0x0014	otp_addr	OTP Write Address register
0x0018	otp_wdata	OTP Write Data register
0x001c	otp_raw_baddr_l	OTP RAM Config register
0x0020	otp_status	OTP Status Control Register
0x0024	otp_int	OTP Int register
0x0028	otp_protect_code	OTP Read Protect Code register
0x002c	otp_pclk timing cfg	OTP PCLK TIMING CFG Register
0x0030	otp_pclk timing cfg en	OTP PCLK TIMING CFG Enable Register

Config register Offset: 0x4000+0x00

Bit	R/W	Reset	Name	Description
Bit[31:6]	R	26'h0	Res	
Bit[5:4]	W/R	2'b01	FDIV	Frequency division from 64mhz 2'b00: don't divide, otp_clk=64hmz; 2'b01: divide by 2, otp_clk=32hmz; 2'b10: divide by 4, otp_clk=16hmz;
Bit[3:0]	W/R	4'b0000	RMODE	4'b0000: user read mode 4'b0001: initial margin read mode(test mode) 4'b0100: PGM margin read mode(test mode) 4'b1001: high temp initial margin read mode(test mode) 4'b1100: high temp PGM margin read mode(test mode)

Note: RMODE will only change after command "switch read mode" execute;

Command register Offset: 0x4000+0x04

Bit	R/W	Reset	Name	Description
Bit[31:16]	R		Res	

Bit[15: 0]	W/R	6'h0	COMMAND	16'h0:res 16'h5a5a:block write command, write otp which data read from RAM;(WEN bit must be set, TROW_EN must be 0) 16'h0123:word write command (WEN bit must be set) 16'h4567:initial margin read_FF command, Check if 4K Main array initial value is all 1; 16'h0110:power_down; 16'h0220:power_up; 16'h0330:goto standby state; 16'h0440:goto deep standby state; 16'h0550:switch read mode;
------------	-----	------	---------	---

Note:all command only effect after set **Execute** reg;

Write Enable register Offset: 0x4000+0x08

Bit	R/W	Reset	Name	Description
Bit[31:0]	WR	32'haabb_ccdd	Write enable	32'h1133_5577:enable, OTP mem can be programed; 32'haabb_ccdd: disable, OTP mem can't be programed;

Test Row En register Offset: 0x4000+0x0c

Bit	R/W	Reset	Name	Description
Bit[31:0]	WR	32' hfff_aabb	Test_row_en	32'h00005566:enable test row access; 32'hffffaabb:disable test row access;

Note1:test row is individual memory block for testing,it does not include in main array size;

Note2:the size of test row is 16 word;before test_row_en is set,a right RMODE must be set;

Command Execute register Offset: 0x4000+0x10

Bit	R/W	Reset	Name	Description
Bit[31:1]	R		Res	
Bit[0]	WO	1'b0	execute	Write 1 to this bit will execute current command and read always return 0;

OTP Write Address register Offset: 0x4000+0x14

Bit	R/W	Reset	Name	Description
Bit[31:14]	R	0	-	Res

OM6621Dx Bluetooth Low Energy Application

Bit[13:0]	W/R	14'h0	OTP_waddr	Write OTP address in single word operation or OTP base address in block write operation which data from RAM1; Range: 14'd0~14'd16383; (OTP mem size is d'16384)
-----------	-----	-------	-----------	--

Note: When COMMAND=16'h0123, OTP_ADDR is otp address for single word write operation; When COMMAND=16'h5a5a, OTP_ADDR is write otp base address for block write operation;

Write Data register Offset: 0x4000+0x18

Bit	R/W	Reset	Name	Description
Bit[31:0]	W/R	32'h0	WDATA	write data to OTP memory;

Note: Only valid When execute word write command;

RAM Config register Offset: 0x4000+0x1c

Bit	R/W	Reset	Name	Description
Bit[31:30]			-	Res
Bit[29:16]	W/R	14'h0	RAM_r_length	read data byte num from RAM1, range is 4~8192; 'd4:num=4; 'd8:num=8; 'd8192:num=8192; (RAM1 size is 2048x32)
Bit[15:13]				Res
Bit[12:0]	W/R	13'h0	RAM_r_baddr	read data byte base address of RAM1, range is 0~8188; 0,4,8...8188;

Note1: only valid when execute block write command;

Note2: (RAM_r_length/4<=d'2048)&& (OTP_waddr/4+ RAM_r_length/4<=d'4096);

Status register Offset: 0x4000+0x20

Bit	R/W	Reset	Name	Description
Bit[31:14]	R	0	-	Res
Bit[13]	R	1'b1	W_protect_en	1'b1:enable;program is forbidden; 1'b0:disable;program is permit;
Bit[12]	R	1'b0	Test_row_en	1'b1:enable; 1'b0:disable;
Bit[11:9]	R	3'h000	CTRL_ST	Otp controller state: 3'b000:power on(ready) 3'b001:standby 3'b010:deep standby 3'b011:power off

				3'b100:programming 3'b101:busy
Bit[8]	R	1'b1	PENVDD2_VDD2	VDD2 EN
Bit[7]	R	N/A	VDD2_RDY	VDD2 ready
Bit[6]	R	1'b0	R_protect en	1'b1:enable; 1'b0:disable;
Bit[5:2]	R	4'h0	PTM	Otp memory operation mode
Bit[1]	R	1'b0	TESTSTU	Check if 4K Main array initial value is all 1; 1'b1:fail, Main array initial value is not all 1; 1'b0:right, Main array initial value is all 1;
Bit[0]	R	N/A	OTP_READY	1'b1:OTP is in idle state and can accept new command 1'b0:not ready;

OTP Int register Offset: 0x4000+0x24

Bit	R/W	Reset	Name	Description
Bit[31:4]	R	0	-	Res
Bit[3]	R	1'b0	Otp_int	Otp int= Otp_int_raw& Otp_int_en; 1'b1:otp interrupt happen; 1'b0:no otp interrupt
Bit[2]	R	N/A	Otp_int_raw	Otp int status before otp_int_en Before otp_ready Otp_int_raw = 0, after otp_ready Otp_int_raw = 1;
Bit[1]	W	1'b0	Otp_int_clr	Set this bit will clear bit2 and bit3
Bit[0]	WR	1'b0	Otp_int_en	Interrupt enable 1'b1:enable 1'b0:disable

Read Protect Code register Offset: 0x4000+0x28

Bit	R/W	Reset	Name	Description
Bit[31: 0]	R	32'hffff_ffff;	Read_protect_code	Read protect code,read from otp mem last word address; user only can write data 32'h1234_abcd to this address to enable otp read protect;

PCLK TIMING CFG register Offset: 0x4000+0x2c

Bit	R/W	Reset	Name	Description
Bit[31:22]	R	0	-	Res
Bit[21:20]	WR	2'h1	Pclk16m_soft_h	Pclk16 cycle number;

				Pclk16_soft_h >=1;
Bit[19:16]	WR	4'h2	Pclk16m_soft_cycle	Pclk16 cycle number; Pclk16_soft_cycle >=2;
Bit[15:14]	R	2'h0	-	Res
Bit[13:12]	WR	2'h1	Pclk32m_soft_h	Pclk32 cycle number; Pclk32_soft_h >=1;
Bit[11:8]	WR	4'h3	Pclk32m_soft_cycle	Pclk32 cycle number; Pclk32_soft_cycle >=2;
Bit[7:6]	R	2'h0	-	Res
Bit[5:4]	WR	2'h2	Pclk64m_soft_h	Pclk64 cycle number; Pclk64_soft_h >=2;
Bit[3:0]	WR	4'h5	Pclk64m_soft_cycle	Pclk64 cycle number; Pclk64_soft_cycle >=4;

PCLK TIMING CFG EN register Offset: 0x4000+0x30

Bit	R/W	Reset	Name	Description
Bit[31:15]	R	0	-	Res
Bit[14]	R	0	Clk_cali_done	
Bit[13:12]	R	Depend on freq_div	Pclk_h	Pclk effect cycle number;
Bit[11:8]	R	Depend on freq_div	Pclk_cycle	Pclk effect cycle number;
Bit[7:1]	R	2'h0	-	Res
Bit[0]	WR	1'b0	Pclk_soft_set	Pclk_soft_cycle and Pclk_soft_h set enable: 1'b1:enable, 1'b0:disable If Pclk_soft_set =1, pclk param set by Pclk_soft_cycle and Pclk_soft_h; If Pclk_soft_set =0, pclk param set by hardware;

Before system clock calibrate done, hardware will set pclk a slow speed param considering otp module clock skew;

Soft can change pclk timing by write PCLK TIMING CFG EN register and set Pclk_soft_set=1;

After system clock calibrate done, OTP read timing set will use a stable and fast param;

Otp frequency	Read speed(pclk freq)
16mhz	8mhz*32bit
32mhz	10.6mhz*32bit
64mhz	12.8mhz*32bit

Table 6.4 OTP read timing set

Fastest speed set

OTP read speed can reach the maximum speed 16mhz by set pclk timing cfg register and pclk timing cfg en register ;Pclk_soft_set = 1;

Otp frequency	Read speed (pclk freq)	pclk timing cfg register set	
16mhz	8mhz*32bit	Pclk16m_soft_h=1	Pclk16m_soft_cycle=2
32mhz	16mhz*32bit	Pclk32m_soft_h=1	Pclk32m_soft_cycle=2
64mhz	16mhz*32bit	Pclk64m_soft_h=2	Pclk64m_soft_cycle=4

Table 6.5 OTP read speed

6.9 GPADC

6.9.1 Introduction

The OM6621Dx is equipped with a high-speed low power 10-bit general purpose Analog-to-Digital Converter (GPADC). It can operate in unipolar (single ended) mode as well as in bipolar (differential) mode.

The ADC has its own voltage regulator (LDO) of 1.0V, which represents the full scale reference voltage.

6.9.2 Main Features

Features:

- 10-bit dynamic ADC with 8 us conversion time
- Maximum sampling rate 125k sample/s
- Single-ended input
- Four single-ended external input channels
- Battery monitoring function
- Offset and zero scale adjust

6.9.3 GPADC Register Map

Offset	Reset	Name	Description
0x002	0x145050	dly_cfg	Configure pd Time
0x004	0x1	adc_cfg0	Start adc
0x005	0x1F400045	adc_cfg1	Configure channel and sequence Time
0x006	0x10800	adc_cfg2	Configure channel and sequence
0x008	0x0	adc_sw_trigger	trigger
0x00C	0x0	ch_0_cfg	Channel 0 gpadc_mode_verf and gpadc_bp
0x00D	0x0	ch_1_cfg	Channel 1 gpadc_mode_verf and gpadc_bp

0x00E	0x0	ch_2_cfg	Channel 2 gpadc_mode_verf and gpadc_bp
0x00F	0x0	ch_3_cfg	Channel 3 gpadc_mode_verf and gpadc_bp
0x010	0x0	ch_4_cfg	Channel 4 gpadc_mode_verf and gpadc_bp
0x011	0x0	ch_5_cfg	Channel 5 gpadc_mode_verf and gpadc_bp
0x012	0x0	ch_6_cfg	Channel 6 gpadc_mode_verf and gpadc_bp
0x013	0x0	ch_7_cfg	Channel 7 gpadc_mode_verf and gpadc_bp
0x014	0x0	gain_err_reg	gain_error data
0x015	0x0	vos_reg	vos data
0x016	0x0	vos_temp_reg	vos_temp data
0x01C	0x0	ch_0_data	Channel 0 Adc data
0x01D	0x0	ch_1_data	Channel 1 Adc data
0x01E	0x0	ch_2_data	Channel 2 Adc data
0x01F	0x0	ch_3_data	Channel 3 Adc data
0x020	0x0	ch_4_data	Channel 4 Adc data
0x021	0x0	ch_5_data	Channel 5 Adc data
0x022	0x0	ch_6_data	Channel 6 Adc data
0x023	0x0	ch_7_data	Channel 7 Adc data

7 Communication Subsystem

7.1 Supported Features

OM6621Dx on-chip Bluetooth system compliant with version 5.0, support all of Bluetooth standard 5.0 feature.

7.2 Radio Transceiver

The Radio Transceiver implements the RF part of the Bluetooth Low Energy protocol. Together with the Bluetooth 5.0 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.

7.2.1 Bluetooth Radio Receiver

The OM6621Dx 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.

7.2.2 Bluetooth Radio Transmitter

The OM6621Dx 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.

7.2.3 Frequency Synthesizer

The OM6621Dx 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.

7.3 Bluetooth Base band Unit

The BLE (Bluetooth Low Energy) core is a qualified Bluetooth 5.0 base-band controller compatible with Bluetooth Smart specification and it is in charge of packet encoding/decoding and frame scheduling.

7.3.1 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

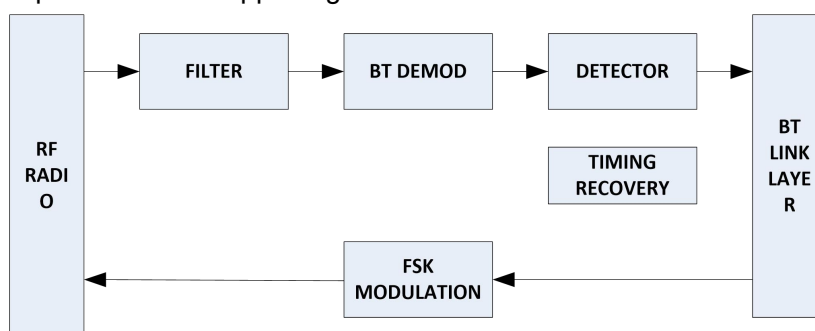


Figure 7.1 OM6621Dx BT Base band

7.4 Performance

7.4.1 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
Maximum received signal , uncoded data at 1 Ms/s		-		0	dBm
Maximum received signal , uncoded data at 2 Ms/s				0	dBm
I/C co-channel Sensitivity, uncoded data at 1 Ms/s		-		-9	dB
I/C co-channel Sensitivity, uncoded data at 2 Ms/s				-9	dB
Adjacent channel selectivity	F = F0+1MHz , uncoded data at 1 Ms/s	-	-	1	dB
	F = F0+1MHz , uncoded data at2 Ms/s			1	dB
	F = F0 -1MHz , uncoded data at 1 Ms/s	-	-	1	dB

I/C Note: F0=2440 MHz	F = F0 -1MHz , uncoded data at 2 Ms/s			1	dB
	F = F0+2MHz , uncoded data at 1 Ms/s	-	-	21	dB
	F = F0+2MHz , uncoded data at 2 Ms/s			21	dB
	F = F0-2MHz , uncoded data at 1 Ms/s	-	-	23	dB
	F = F0-2MHz , uncoded data at 2 Ms/s			23	dB
	F = F0+3MHz , uncoded data at 1 Ms/s	-	-	33	dB
	F = F0+3MHz , uncoded data at 2Ms/s			33	dB
	F = F0-3MHz , uncoded data at 1 Ms/s	-	-	34	dB
	F = F0-3MHz , uncoded data at 2 Ms/s			34	dB

Table 7.1 OM6621Dx BLE Receiver architecture

7.4.2 BLE Transmitter Performance

[Supply Voltage = 3.3V @ 25℃]

Parameter		Min	Typ	Max	Unit
Maximum RF transmit power		-	10	-	dBm
RF power control range		-20	-	10	dBm
RF power range control resolution		2.7	3	3.7	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 (uncoded data at 2 Ms/s)		450	500	550	KHz
Δf2max maximum modulation (uncoded data at 1 Ms/s)		100%			
Δf2max maximum modulation (uncoded data at 2 Ms/s)		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 Offset (uncoded data at 1 Ms/s)			4.02		KHz
Frequency Offset (uncoded data at 2 Ms/s)			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 rate (uncoded data at 1 Ms/s)			-3.13		KHz/50us
Frequency Drift rate (uncoded data at 2 Ms/s)			-3.13		KHz/50us

OM6621Dx Bluetooth Low Energy Application

Parameter	Min	Typ	Max	Unit
Initial Frequency Drift (uncoded data at 1 Ms/s)		-2.25		KHz
Initial Frequency Drift (uncoded data at 2 Ms/s)		-2.25		KHz
2nd harmonic content			-50	dBm
3rd harmonic content	-		-50	dBm

Table 7.2 OM6621Dx BLE Transceiver architecture

8 Package Information

The OM6621Dx has QFN20 and QFN32 package, the information is as below:

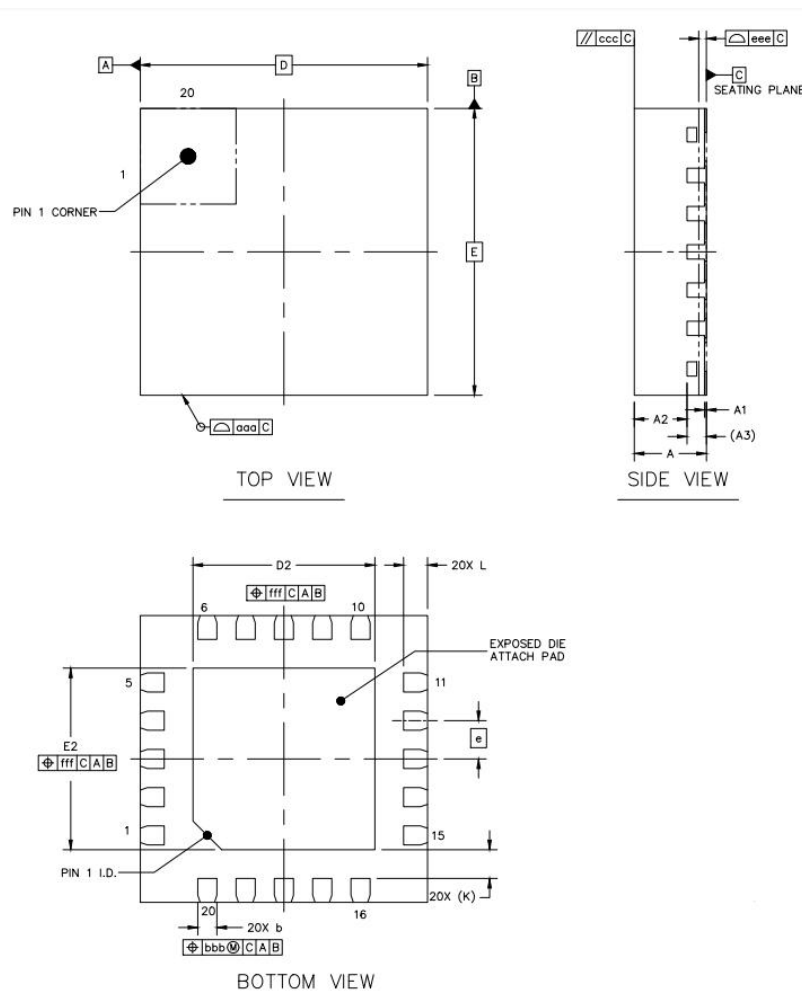


Figure 8.1 OM6621DB QFN20 package

SYMBOL	MILLMETER		
	MIN	NOM	MAX
A	0.7	0.75	0.8
A1	0	0.02	0.05
A2	---	0.55	---
A3	0.203 REF		
b	0.15	0.2	0.25
D	3 BSC		
E	3 BSC		
e	0.4 BSC		
D2	1.8	1.9	2
E2	1.8	1.9	2
L	0.15	0.25	0.35

K	0.3 REF
aaa	0.1
ccc	0.1
eee	0.08
bbb	0.07
fff	0.1

Table 8.1 OM6621DB QFN20 package

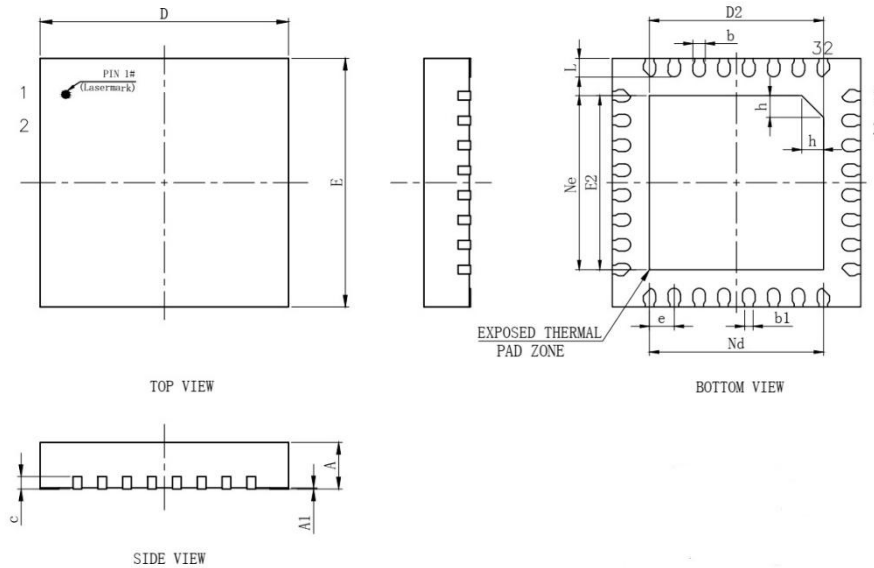


Figure 8.2 OM6621DQ QFN32 package

SYMBOL	MILLMETER		
	MIN	NOM	MAX
A	0.70	0.75	0.80
	0.80	0.85	0.90
	0.85	0.90	0.95
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	122*122		

Table 8.2 OM6621DQ QFN32 package

9 Ordering Information

OM6621Dx offers devices below for different application requirement.

Part No.	Type	Package Size	Packing	MOQ(PCS)	Status
OM6621DB	QFN-20L	3*3mm 0.4mm Pitch	Tape/Reel	3000	MP
OM6621DQ	QFN-32L	4*4mm 0.4mm Pitch	Tape/Reel	3000	MP

Table 9.1 OM6621DB and OM6621DQ ordering information

10 Tape and reel information

10.1 Tape orientation

General orientation of OM6621DQ package in the carrier tape.

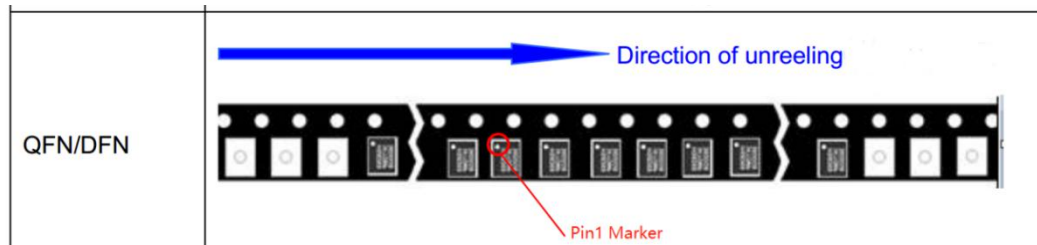


Figure 10.1 OM6621DQ Tape Orientation

10.2 Tape and reel dimensions

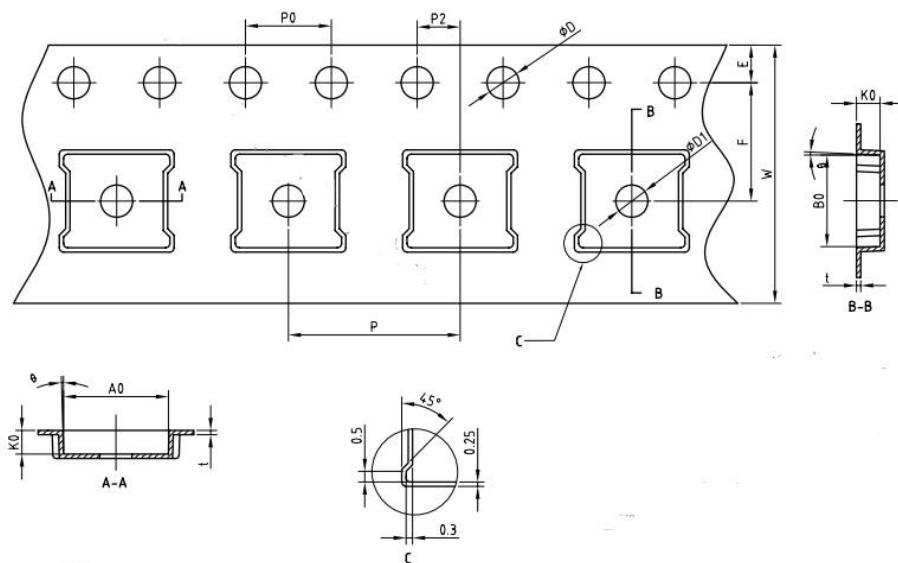


Figure 10.2 OM6621DQ Tape and Reel Dimensions

Common Size:

Appearance	Size / mm
E	1.75 ± 0.10
F	5.50 ± 0.05
P2	2.00 ± 0.10
D	1.55 ± 0.05
D1	$1.50_0^{+0.25}$
P0	4.00 ± 0.10
10P0	40.00 ± 0.20

Table 10.1 OM6621DQ Common Size

Bag Size:

Appearance	Size / mm
W	12.00 ± 0.30
P	8.00 ± 0.10
A0	4.30 ± 0.10
B0	4.30 ± 0.10
K0	1.10 ± 0.10
t	0.30 ± 0.05

Table 10.2 OM6621DQ Bag Size

11 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 11.1 Glossary and Abbreviations

12 Reference Documents

Below documents are referred in this datasheet:

Documents	Description
OM6621Dx Hardware guideline	Introduce OM6621Dx hardware design

Table 12.1 Reference Documents

Sales and Service



Beijing OnMicro Electornics Co., Ltd.

5F, HuiZhong Tower 1, No.1 Shangdi 7th Street, Hai Dian District, Beijing, China,100084

Tel: +86-10-82858804

Fax: +86-10-82858761

Shanghai Subsidiary Company

Room 501, E-Building No. 666, Shengxia Road, Pudong New Area, Shanghai, China, 201210

Tel: +86-21-60137655

Fax: +86-21-60137656

Guangzhou Subsidiary Company

Room 1002, Building A(Chip Building), No.18 , Science Avenue, Huang Pu District, Guangzhou, China, 510670

Tel: +86-20-82038985

Shenzhen Subsidiary Company

Room 407-408, West Block, Skyworth Semiconductor Building, Gaoxin Nansi Road, Yuehai Street, Nanshan District, Shenzhen China, 518063

Tel: +86-755-26542255

Fax: +86-755-86360696

Hong Kong Subsidiary Company

Office D, 20/F, Kings Wing Plaza 2, No.1 On Kwan Street, Sha Tin, N.T., Hong Kong, 999077

Tel: +852-36190820

OnMicro International Sales Office

Sales: William Zhong Tel: +1 (609) 454-0017(US)

Email: William.Zhong@onmicro.com.cn

Technical Support: Sheldon Xu Tel: +1 (908) 884-3771(US)

Email: Sheldon.Xu@onmicro.com.cn

Important Notice

The information provided herein by OnMicro is believed to be reliable; however, OnMicro makes no warranties regarding the information and assumes no responsibility or liability whatsoever for the use of the information. Customers should be aware that all information contained herein is subject to change without notice. Unless explicitly specified, OnMicro products are not warranted or authorized for use as critical components in medical, life-saving, or life-sustaining applications, or other applications where a failure would cause severe personal injury or death.