

WT32L064/032

Flash Memory Type

32-bit Microcontroller

Data Sheet

Rev. 1.02

February 2022

Copyright Notice

This data sheet is copyrighted by Weltrend Semiconductor, Inc. Do not reproduce, transform to any other format, or send/transmit any part of this documentation without the express written permission of Weltrend Semiconductor, Inc.

Disclaimers

Right to make change –

This document provides technical information for user. Weltrend Semiconductor, Inc. reserves the right to make change without further notice to any products herein.

Table of Contents

Table of Contents	i
List of Figures	vii
List of Tables	ix
1 General Description	1
1.1 Main Features	2
1.2 System Block Diagram	4
1.3 System Power Architecture	5
1.4 System Clock Tree	6
1.5 ARM® Cortex™-M0 Processor	7
1.5.1 Cortex M0 Processor Features	7
1.5.2 Nested Vectored Interrupt Controller (NVIC)	8
1.5.3 System Timer (SysTick)	10
2 Pin Assignment	11
2.1 Package	11
2.1.1 LQFP-64	11
2.1.2 LQFP-48	12
2.1.3 QFN-32	13
2.2 Pin Description	14
2.3 Alternate Function I/O Priority	18
3 Memory Mapping	19
3.1 AMBA Bus Address Mapping	19
3.2 APB Memory Space	19
4 Reset and Clock Control	22
4.1 Main Features	22
4.2 Block Diagram	22
4.3 Register Definition	24
4.4 Functional Description	28
4.4.1 Reset	28
4.4.1.1 System Reset	28
4.4.1.2 Power Reset	30
4.4.1.3 Standby Exit Reset	31
4.4.2 Clock	31
4.4.2.1 PLL Clock	31
5 Power Manage Unit	32
5.1 Power Management Unit Features	32
5.2 Power Domain Overview	32
5.3 Voltage Regulator	33
5.4 Dynamic Core Voltage	33
5.5 Power Saving Mode	34
5.5.1 Extended Interrupt (EXTI) for wake up	35
5.5.2 Behavior of Clocks in Power Saving Modes	35
5.6 Register Definition	37
5.7 State Machine	40
5.7.1 Run mode	40
5.7.2 Sleep Mode	41
5.7.3 Stop mode	44
5.7.4 Standby mode	47
5.8 Setting Examples	50
5.9 Operating mode & Wake up Source	50

6	eFlash Control	52
6.1	Main Features	52
6.2	Block Diagram	52
6.3	Register Definition	53
6.4	Functional Description	58
6.4.1	NVM Functional Description	58
6.4.2	Reading the NVM	58
6.4.3	Writing/Erasing the NVM	62
6.4.4	Unlocking/Locking Operations	63
6.4.5	Status Register	64
6.5	Memory Protection	65
6.5.1	RDP (Read Out Protection)	65
6.5.2	PcROP (Proprietary Code Read-Out Protection)	66
6.5.3	Protections Against unwanted Write/Erase Operations	67
6.5.4	Reading the NVM	68
6.5.5	Protection Errors	68
6.6	NVM Interrupts	68
6.6.1	Hard Fault	69
6.7	Memory Interface Management	69
6.7.1	Operation Priority and Evolution	69
6.7.2	Sequence of Operations	69
6.7.3	Change the Number of Wait-States while Reading	70
6.8	Option Bytes	70
6.8.1	API for Option Bytes	70
6.8.2	Mismatch when loading protection flags	71
6.8.3	Simulated EEPROM	72
6.8.3.1	Description	72
7	DMA	73
7.1	Features	73
7.2	Function description	74
7.2.1	Block Diagram	74
7.2.2	AHB Master Interface	74
7.2.3	AHB Slave Interface	74
7.2.4	FIFO Buffer	74
7.2.5	DMA Core	74
7.2.6	Prioritizing Arbiter	74
7.2.7	Control Logic & Register bank	75
7.3	DMA Register Table	75
7.3.1	Global setting for interrupt status / clear, Channel Busy	75
7.3.2	DMA Channel x source AddrESS register	78
7.3.3	DMA Channel x destination address register	78
7.3.4	DMA channel x number of data register	79
7.3.5	DMA channel x configuration register	80
8	GPIO	86
8.1	Main Features	86
8.2	Block Diagram	86
8.3	Register Definition	86
8.4	Functional Description	96
8.4.1	General-purpose I/O	97
8.4.2	General-purpose I/O	97
8.4.3	I/O port control registers	97
8.4.4	I/O port data registers	98
8.4.5	I/O data bitwise handling	98
8.4.6	I/O alternate function input/output	98

8.4.7	Input configuration	98
8.4.8	Output configuration	99
8.4.9	Alternate function configuration	100
8.4.10	Analog configuration	101
8.4.11	Using the HXTAL or LXTAL oscillator pins as GPIOs	102
9	USB	103
9.1	Main Features	103
9.2	Block Diagram	103
9.3	Register Definition	104
9.4	Function Description	115
9.4.1	Main blocks	115
9.4.2	Function endpoints	116
9.4.3	Transmit FIFOs	116
9.4.3.1	Transmit FIFOs Features	116
9.4.3.2	Transmit Data Set Management	117
9.4.4	Receive FIFOs	118
9.4.4.1	Receive FIFOs Features	118
9.4.4.2	Receive Data Set Management	119
9.4.5	Setup Token Receive FIFO Handling	120
9.4.6	Suspend and Resume	120
9.4.7	FIFO memory address mapping	122
10	Clock Recovery System	123
10.1	Main Features	123
10.2	Block Diagram	124
10.3	Register Definition	124
10.4	Functional Description	126
10.4.1	Synchronization input	126
10.4.2	Frequency error measurement	126
10.4.3	Frequency error evaluation and automatic trimming	127
10.4.4	CRS initialization and configuration	127
10.4.5	CRS low-power modes	128
10.4.6	CRS interrupts	128
11	I2C	129
11.1	Main Features	129
11.2	Block Diagram	130
11.3	Register Definition	130
11.4	Functional Description	136
11.4.1	Mode Selection	136
11.4.1.1	Communication Flow	136
11.4.2	I2C initialization	137
11.4.2.1	Noise filter	137
11.4.2.2	Timing	137
11.4.2.3	Software reset	137
11.4.3	Data transfer	137
11.4.3.1	Reception	137
11.4.3.2	Transmission	137
11.4.3.3	Hardware transfer management	137
11.4.4	I2C Slave Mode	139
11.4.5	I2C Master Mode	140
11.4.5.1	Communication Initialization	140
11.4.5.2	Master Transmitter	140
11.4.5.3	Master Receiver	141
11.4.6	DMA requests	141

11.4.6.1	Transmission using DMA	141
11.4.6.2	Reception using DMA.....	142
12	UART	143
12.1	Main Features	143
12.2	Block Diagram.....	144
12.3	Register Definition.....	144
12.4	Functional Description	147
12.4.1	Fractional baud rate generation.....	147
12.4.2	Receiver	148
12.4.3	Parity control	149
12.4.4	Single-wire half-duplex communication	149
13	SPI.....	150
13.1	Main Features	150
13.2	Block Diagram.....	151
13.3	Register Definition.....	151
13.4	Functional Description	154
13.5	SPI Timing Diagram	157
14	I2S.....	159
14.1	Main Features	159
14.2	Block Diagram.....	159
14.3	Register Definition.....	159
14.4	Functional Description	161
14.4.1	The Basics of I2S Bus	161
14.4.2	Left Justified Mode.....	162
14.4.3	I2S Mode.....	162
14.4.4	I2S Clock Generation.....	163
15	Real-Time Clock	165
15.1	Main Features	165
15.2	Block Diagram.....	165
15.3	Register Definition.....	166
15.4	Functional Description	168
15.4.1	Programmable Alarm	168
15.4.2	Periodic Interrupt	168
16	Independent Watchdog Timer.....	169
16.1	Main Features	169
16.2	Block Diagram.....	169
16.3	Register Definition.....	170
16.4	Functional Description	171
17	Window Watchdog Timer	172
17.1	Main Features	172
17.2	Block Diagram.....	172
17.3	Register Definition.....	172
17.4	Functional Description	173
18	PWM.....	175
18.1	Main Features	175
18.2	Register Definition.....	176
18.3	Functional Description	178
18.3.1	Independent PWM	178
19	Timer.....	179
19.1	Main Features	179
19.2	Block Diagram.....	180

19.3	Register Definition.....	180
19.4	Functional Description	186
19.4.1	Multiple Capture and Match Pins.....	186
19.4.2	Interrupts.....	186
19.4.3	DMA	186
19.4.4	Count Control.....	186
19.4.5	Capture Control	187
19.4.6	Match Control	187
20	ADC.....	188
20.1	Main Features	188
20.2	Block Diagram.....	188
20.3	Register Definition.....	188
20.4	Functional Description	191
20.4.1	Conversion function	191
20.4.1.1	ADC_SLOW=0, When ADC clock ≤ APB clock	191
20.4.1.2	ADC_SLOW=1, When ADC clock > APB clock	191
20.4.2	Conversion timing	192
20.4.3	Analog Watchdog Window.....	192
21	DAC.....	193
21.1	Main Features	193
21.2	Block Diagram.....	193
21.3	Register Definition.....	194
22	Ultra-Low Power Comparators	195
22.1	Main Features	195
23	CRC32.....	196
23.1	Main Features	196
23.2	Register Definition.....	196
23.3	Functional Description	196
23.4	CRC32 example code: using WT32L064/032	197
24	Boot ROM & IAP	200
24.1	Introduction	200
24.2	How to enter the IAP.....	200
24.3	Boot & IAP Memory Mapping.....	200
24.4	Flowchart.....	202
25	Electrical Characteristics	203
25.1	Absolute Maximum Ratings	203
25.2	DC Characteristics	204
25.2.1	Power consumption	204
25.2.2	Digital I/O Characteristics	205
25.2.3	LVR Characteristics	206
25.2.4	PLL Characteristics.....	206
25.2.5	ADC Characteristics	207
25.2.6	DAC Characteristics	208
25.2.7	LDO Characteristics.....	208
25.2.8	HSI 16MHz.....	209
25.2.9	MSI 4.2MHz	209
25.2.10	IRC48MHz Characteristics	210
25.2.11	IRC37kHz Characteristics.....	211
25.2.12	Crystal 32768 Characteristics.....	211
25.2.13	CMP Characteristics	212
25.2.14	BOR/PVD Characteristics.....	213
25.2.15	POR Characteristics	214

25.2.16	POWER Supply Supervisors	215
25.3	AC Characteristics	216
25.4	Thermal Characteristics	217
26	Application Circuit	218
26.1	VDD33 power supply	218
26.2	VDDBat power supply.....	218
26.3	USB VBus 5V power supply	218
26.4	VBAT & VDD33 dual power supply.....	219
27	Product Naming Rule.....	220
28	Ordering Information	221
28.1	WT32L064.....	221
28.1.1	Top Marking – LQFP64/48	221
28.1.2	Top Marking – QFN32	221
28.2	WT32L032.....	222
28.2.1	Top Marking – LQFP64/48	222
28.2.2	Top Marking – QFN32	222
29	Package Outline Drawing	223
29.1	LQFP-64 Outline Drawing.....	223
29.2	LQFP-48 Outline Drawing.....	224
29.3	QFN-32 Outline Drawing.....	225
30	Development Tool	226
31	Revision History	227

List of Figures

Figure 1	WT32L064/032 System Block Diagram	1
Figure 2	System Block Diagram	4
Figure 3	System Power Architecture	5
Figure 4	System Clock Tree	6
Figure 5	ARM® Cortex™-M0 Processor	7
Figure 6	Cortex M0 NVIC Vector	8
Figure 7	Cortex M0 SysTick	10
Figure 8	RCC Module Reset Diagram	22
Figure 9	RCC Module Clock Diagram	23
Figure 10	PLL Diagram.....	31
Figure 11	Power Domain Overview	32
Figure 12	Low Power Run mode state machine	40
Figure 13	Sleep state machine	41
Figure 14	Timing Diagram of Entering Sleep Mode	42
Figure 15	Timing Diagram of Exit Sleep mode	43
Figure 16	Stop Mode State Machine	44
Figure 17	Timing Diagram of Entering Stop Mode	45
Figure 18	Timing Diagram of Exiting Stop Mode	46
Figure 19	Standby Mode State Machine	47
Figure 20	Timing Diagram of Enter Standby Mode	48
Figure 21	Timing Diagram of Exit Standby Mode.....	49
Figure 22	Power Down Diagram	50
Figure 23	eFlash Control Block Diagram.....	52
Figure 24	Structure of one internal buffer	60
Figure 25	Memory Protection	65
Figure 26	DMA Block Diagram	74
Figure 27	GPIO Block Diagram	86
Figure 28	Basic structure of an I/O port bit.....	96
Figure 29	Input configuration	99
Figure 30	Output configuration	100
Figure 31	Alternate function configuration.....	101
Figure 32	Analog configuration.....	102
Figure 33	USB Block Diagram.....	103
Figure 34	USB Endpoints FIFO	104
Figure 35	Transmit FIFO Buffer Outline (EX: for 8 bytes FIFO)	116
Figure 36	Receive FIFO Outline (EX: for 8 bytes FIFO)	118
Figure 37	Suspend and Resume State Diagram.....	121
Figure 38	CRS block diagram	124
Figure 39	CRS Counter Behavior	126
Figure 40	I2C Block Diagram	130
Figure 41	I2C Bus Protocol.....	136
Figure 42	UART Block diagram	144
Figure 43	Example for Baud Rate Generator	148
Figure 44	SPI module block diagram	151
Figure 45	I2S block diagram.....	159
Figure 46	(a) I2S Master mode (b) I2S Slave mode.....	161
Figure 47	The basic timing diagram of the I2S interface.....	162
Figure 48	Left justified mode timing diagram of I2S interface	162

Figure 49	I2S mode timing diagram of I2S interface	163
Figure 50	RTC Block Diagram	165
Figure 51	IWDT Block Diagram	169
Figure 52	WWDT Block Diagram	172
Figure 53	Timing Diagram of WWDT	174
Figure 54	PWM Block Diagram	175
Figure 55	PWM0~3 Function	178
Figure 56	Timer Module Block Diagram	180
Figure 57	ADC Block Diagram	188
Figure 58	Conversion of a Sequence by ADC_SLOW=0	191
Figure 59	Conversion by ADC_SLOW=1	191
Figure 60	ADC Conversion Time	192
Figure 61	Timing Diagram by ADC_SLOW=0	192
Figure 62	Analog Watchdog Window	192
Figure 63	DAC Block Diagram	193
Figure 64	DAC Analog Block Diagram	193
Figure 65	WT32L064/032 Development Kit	226

List of Tables

Table 1	WT32L064/032 Interrupt and Exception Vectors	9
Table 2	Pin Count Table	14
Table 3	Pin Assignment Table	15
Table 4	RCC Control Register	24
Table 5	Performance versus V_{CORE} Ranges	33
Table 6	Summary of Saving Power Modes	34
Table 7	EXTI Lines Connections	35
Table 8	PMU Control Register	37
Table 9	Peripherals on the Operating Mode	51
Table 10	eFlash Control Register Table	53
Table 11	NVM organization	58
Table 12	Number of wait states	58
Table 13	Pre-fetch & Pre-buffer Management	60
Table 14	Configurations for buffers and speculative reading	62
Table 15	PcROP & wrprot Flash Memory Protected Range	66
Table 16	Memory access vs mode	67
Table 17	Flash interrupt request	68
Table 18	API Functions	71
Table 19	GPIO Control Register	86
Table 20	USB Control Register	104
Table 21	Writing to the Byte Count Register	117
Table 22	Truth Table for Transmit FIFO Management	118
Table 23	Status of the Receive FIFO Data Set	119
Table 24	Truth Table for Receive FIFO Management	119
Table 25	FIFO Memory Address Mapping	122
Table 26	CRS Control Register	124
Table 27	Effect of Low-Power Modes on CRS	128
Table 28	CRS Interrupt Control Bits	128
Table 29	I2C Control Register	130
Table 30	UART Control Register	144
Table 31	UART Baud Rate Table	147
Table 32	UART Noise Error	149
Table 33:	Parity control table	149
Table 34	SPI Control Register	151
Table 35	I2S Control Register	159
Table 36	I2S Sampling Rate	163
Table 37	I2S System Clock = 24MHz (assume $f_{BCLK}=64fs$)	163
Table 38	I2S System Clock = 12MHz (assume $f_{BCLK}=64fs$)	164
Table 39	RTC Control Register	166
Table 40	IWDT Control Register	170
Table 41	IWDT Min./Max. Timeout Period at 37 kHz Clock Input	171
Table 42	WWDT Control Register	172
Table 43	WWDT Min./Max Timeout Value at 24.00MHz	174
Table 44	PWM Control Register	176
Table 45	Timer Control Register	180
Table 46	ADC Control Register	188
Table 47	DAC Control Register	194
Table 48	CRC32 Control Register	196

1 General Description

The WT32L064/032 is an ultra-low power Arm Cortex M0-based 32-bit microcontroller with 64KB/32KB embedded flash memory and 8KB SRAM. It could operate up to 32MHz and is able to achieve 32 MIPS performance due to the advanced design by Weltrend, where zero wait-state can be obtained while accessing flash memory.

In addition, with low power design technique and extremely low leakage semiconductor process technology, the power consumption of WT32L064/032 can be less than 0.6 μ A in stop mode and less than 0.3 μ A in standby mode.

This highly integrated MCU contains rich peripherals such as 12-bit ADC, 12-bit DAC, CRC32, GPIO, PWM, USB2.0, I2S, I2C, UART, SPI, and etc., which is perfectly suitable for battery-powered and energy harvesting applications.

Some examples of the applications of the WT32L064/032 include True Wireless Stereo (TWS) charging case, dual-mode wireless gaming keyboard and mouse, flowmeter, various sensor applications such as gas, temperature, humidity and pressure sensor as well as sensor hub.

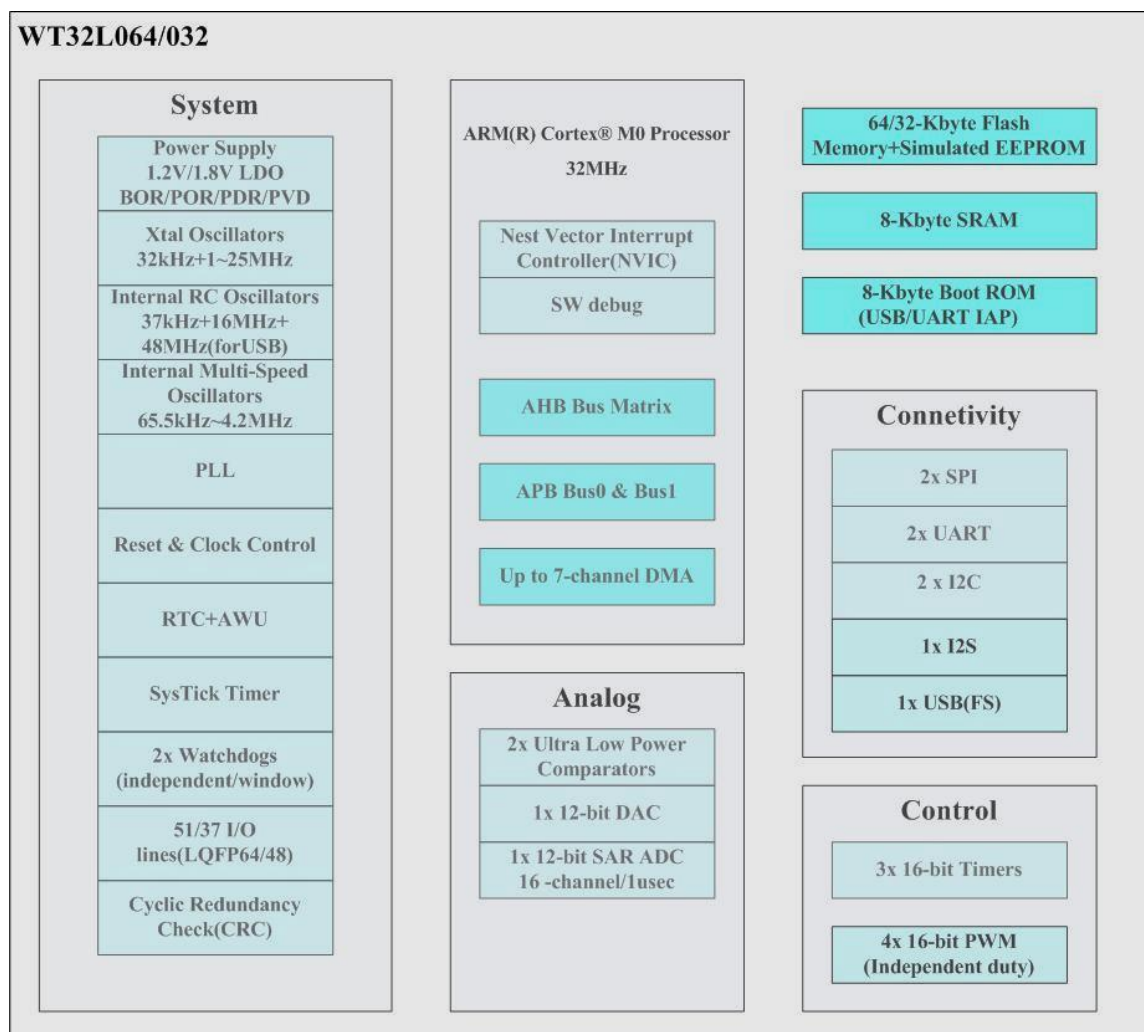


Figure 1 WT32L064/032 System Block Diagram

1.1 Main Features

- Ultra-low-power platform
 - 1.65V to 3.6V power supply
 - ✓ **1.65V~1.8V @Max.4.2 MHz (Flash read only)**
 - ✓ **1.8V~2.2V @Max.4.2 MHz (Flash read & write)**
 - ✓ **2.3V~3.6V @Max.32 MHz (Flash read & write)**
 - -40°C to 105°C temperature range
 - 0.3 µA Standby mode
 - **1.0 µA** Standby mode with RTC on
 - **0.6 µA** Stop mode
 - **1.3 µA** Stop mode with RTC on
- Core: ARM[®] 32-bit Cortex[®]-M0 processor
 - From 65.5 kHz up to 32 MHz Max.
 - **0.88 DMIPS/MHz**
- Interrupt Source

ARM Cortex-M0 built-in Nested Vectored Interrupt Controller (NVIC).

 - 32 external interrupt inputs, each with four levels of priority
 - Dedicated non-Maskable Interrupt (NMI) input
 - Support for both level-sensitive and pulse-sensitive interrupt lines
 - Wake-up Interrupt Controller (WIC), supports sleep mode
- System Tick Timer
 - 24-bit timer clock is fixed to the frequency of the system clock
- Memories
 - WT32L064: Up to 64 KB Flash memory
 - WT32L032: Up to 32 KB Flash memory
 - 8 KB SRAM
- Up to 51 fast I/Os (most 5V tolerant)
- Reset and supply management
 - Ultra-safe, low-power BOR (brownout reset) with 5 selectable thresholds
 - Ultra-low-power POR/PDR
 - Programmable voltage detector (PVD)
- Clock sources
 - 1 MHz to 25 MHz crystal oscillator

- 32 kHz oscillator for RTC and calendar function (Notice: RTC clock needs to be enabled first, please refer to RCC Table 4 for more details)
- High speed internal 16 MHz factory-trimmed RC ($\pm 1\%$)
- Internal low-power 37 kHz RC
- Internal multispeed low-power 65 kHz to 4.2 MHz RC
- Internal self- calibration of 48 MHz RC for USB
- PLL for CPU clock
- Development support
 - Serial wire debug supported
- Rich Analog peripherals
 - 12-bit SAR ADC **0.5** Ms/s up to 16 channels (down to 1.8 V)
 - 12-bit channel DACs with output buffers (down to 1.8 V)
 - Ultra-low-power comparators (window mode and wake up capability, down to 1.65 V)
- 7-channel DMA controller, supporting ADC, SPI, I²C, UART, Timers, and USB
- Peripheral communication interfaces
 - 1x USB 1.1 crystal-less
 - 2x UART
 - Up to 2x SPI 16 M bits/sec.
 - 2x I2C
 - 1x I2S up to 32-bit word size
- 6x timers: 3x 16-bit with up to 4 channels, 1x RTC and 2x Watchdogs (independent/window)
- 4-channel PWM
- CRC calculation unit
- Green package: LQFP-64 (7mmx7mm), LQFP-48 (7mmx7mm), and QFN32 (5mmx5mm)

1.2 System Block Diagram

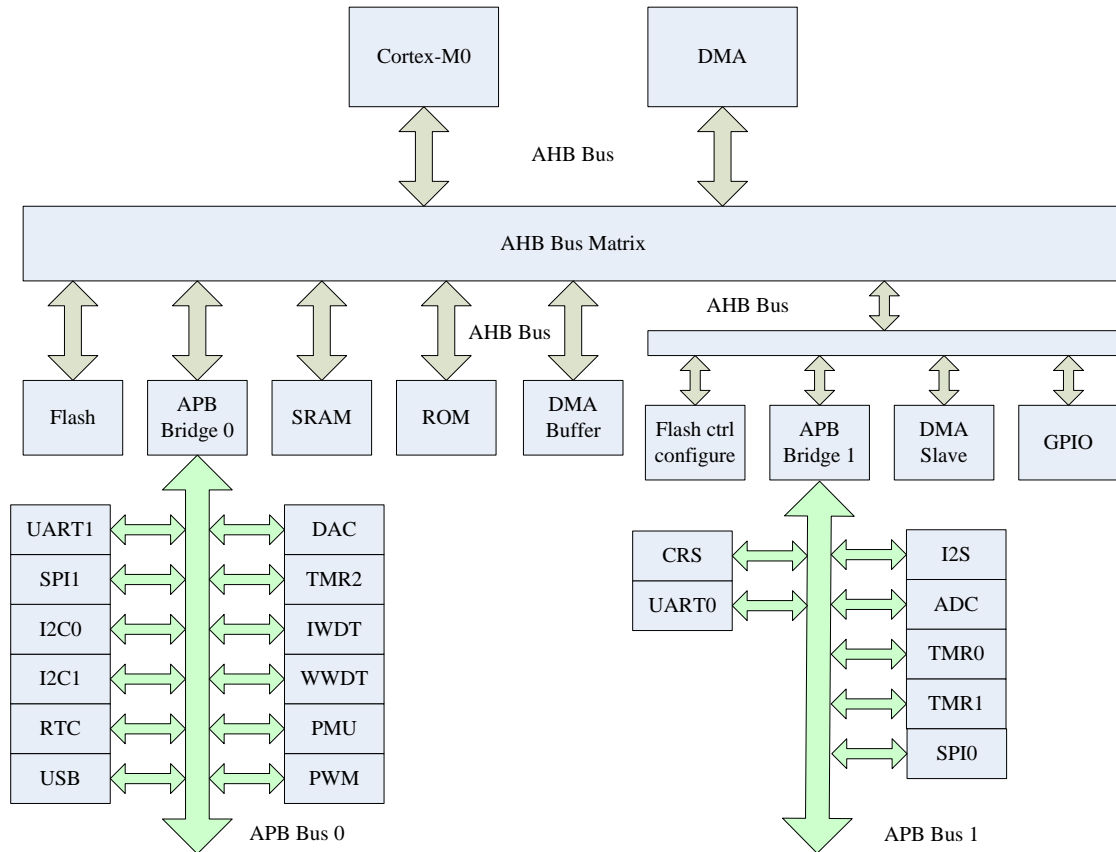
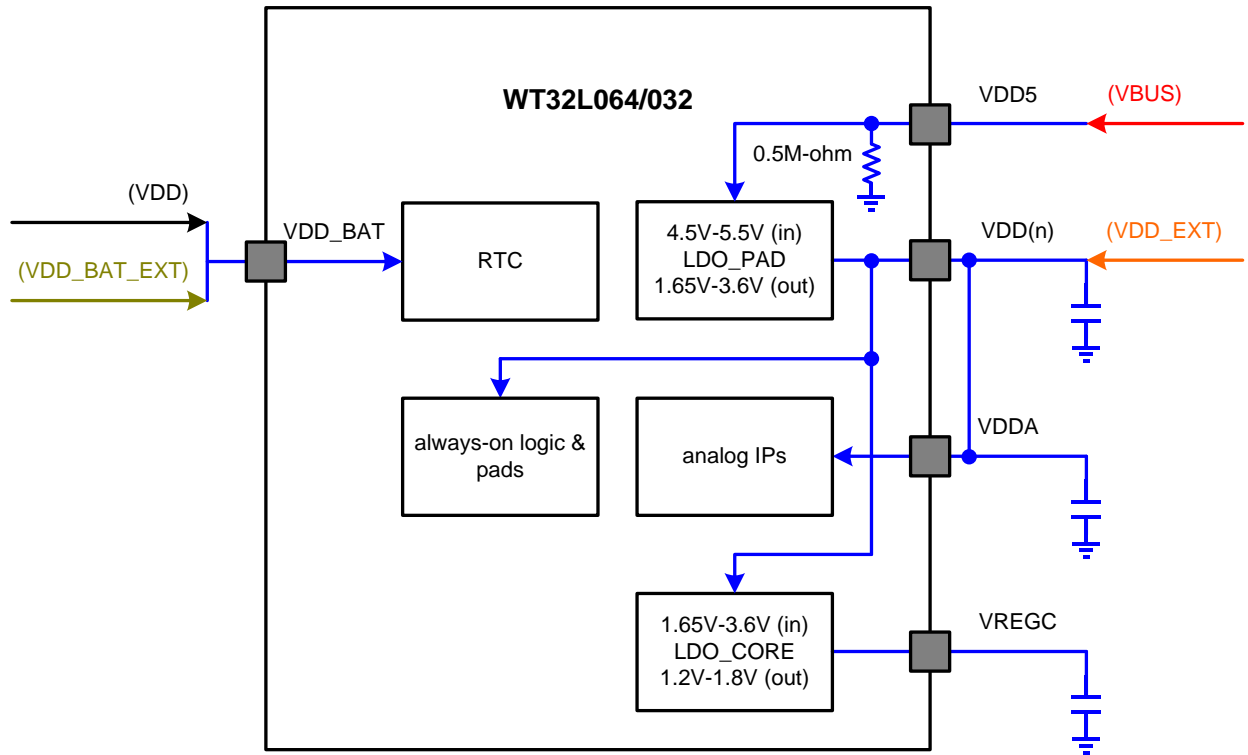


Figure 2 System Block Diagram

1.3 System Power Architecture



Notes:

1. blue circuits are fixed
2. only one of **VBUS** or **VDD_EXT** is supplied, the other is floating
3. only one of VDD or **VDD_BAT_EXT** is supplied for VDD_BAT, the other is floating

Figure 3 System Power Architecture

1.5 ARM® Cortex™-M0 Processor

The Cortex™-M0 processor is a configurable, multistage, 32-bit RISC processor. It has an AMBA AHB-Lite interface and includes a NVIC component. It also has optional hardware debug functionality. The processor can execute Thumb code and is compatible with other Cortex-M profile processor.

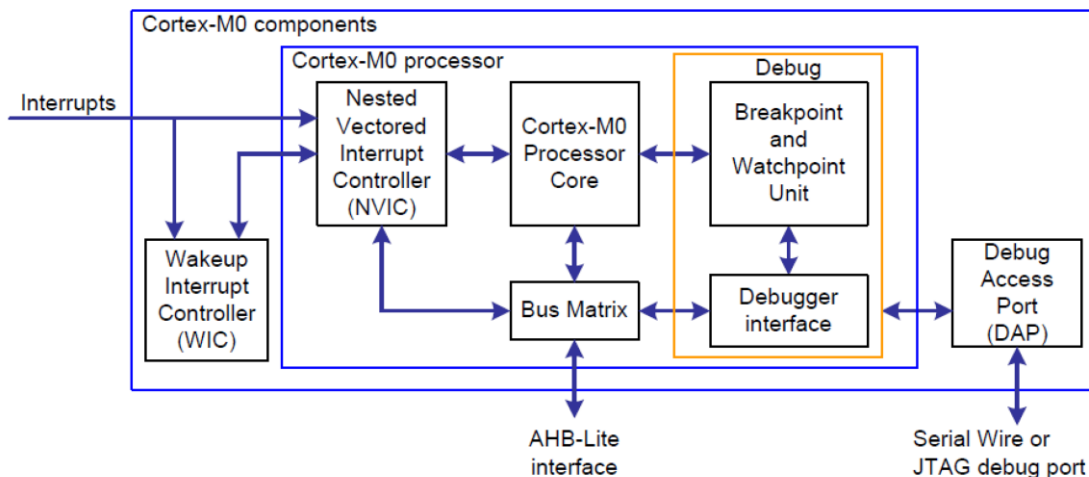


Figure 5 ARM® Cortex™-M0 Processor

1.5.1 Cortex M0 Processor Features

- Cortex-M0 processor includes
 - ✓ Cortex-M0 processor core
 - ✓ Nested Vectored Interrupt Controller (NVIC)
 - ✓ System Timer (SysTick)
- ARMv6-M Thumb instruction set
- NVIC: 32 external interrupt inputs
- Debug: 4 HD breakpoints, 2 watchpoints.
- Bus interfaces: 32-bit AMBA-3 AHB-Lite system interface
- ARMv6-M (which is a subset of ARMv7-M, upward compatible)
 - ✓ It supports only the Thumb instruction set. No Interworking code is required.
 - ✓ Total are 56 instructions. 6 are 32-bit length, the others are 16-bit length
 - ✓ 32-bit instructions: BL, DMB, DSB, ISB, MRS, MSR
- Supports byte (8-bit), halfword (16-bit) and word (32-bit) data types, each must be accessed with natural alignment.
- Built-in timer and interrupt controller
 - ✓ SysTick, NVIC
- Memory mapped I/O
 - ✓ Use same instructions to access memory and registers
 - ✓ Ex: LDR, STR

1.5.2 Nested Vectored Interrupt Controller (NVIC)

Cortex-M0 provides an interrupt controller as an integral part of the exception mode, named as “Nested Vectored Interrupt Controller (NVIC)”. It is closely coupled to the processor kernel and provides following features:

- Nested and Vectored interrupt support
- Automatic processor state saving and restoration
- Dynamic priority changing
- Reduced and deterministic interrupt latency
- An exception may be an interrupt or a hardware error
- Each exception has exception number, priority number and vector address
 - ✓ Vector address
 - ✓ Vector table base address is fixed at 0x00000000

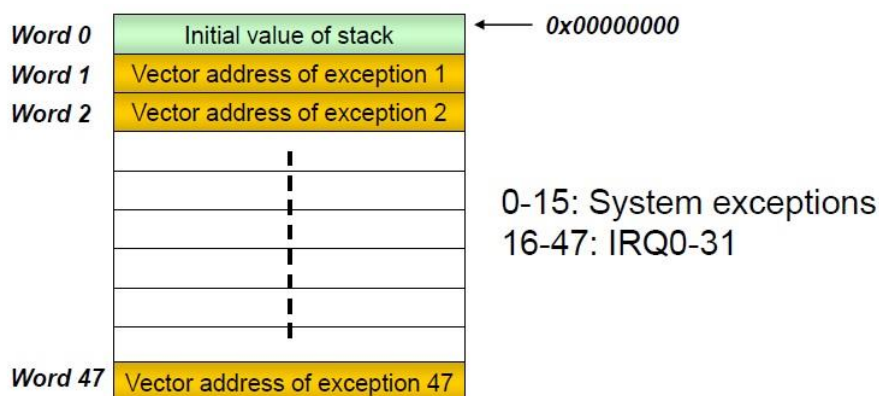


Figure 6 Cortex M0 NVIC Vector

The NVIC prioritizes and handles all supported exceptions. All exceptions are handled in “Handler Mode”. The NVIC architecture supports 32 (IRQ[31:0]) discrete interrupts with 4 levels of priority. All of the interrupts and most of the system exceptions can be configured to different priority levels. When an interrupt occurs, the NVIC will compare the priority of the new interrupt to the current running priority. If the priority of the new interrupt is higher than the current one, the new interrupt handler will override the current handler. Table1 shows WT32L064/032 interrupt and exception vectors.

Table 1 WT32L064/032 Interrupt and Exception Vectors

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-3	fixed	Reset	Reset	0x0000 0004
-	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
-	-1	fixed	HardFault	All class of fault	0x0000 000C
-	3	settable	SVCall	System service call via SWI instruction	0x0000 002C
-	5	settable	PendSV	Pendable request for system service	0x0000 0038
-	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	UART0	UART0 global interrupt	0x0000 0040
1	8	settable	UART1	UART1 global interrupt	0x0000 0044
2			Reserved		0x0000 0048
3	10	settable	I2C0	I2C0 global interrupt	0x0000 004C
4	11	settable	I2C1	I2C1 global interrupt	0x0000 0050
5	12	settable	PWM	PWM global interrupt	0x0000 0054
6	13	settable	SPI0	SPI0 global interrupt	0x0000 0058
7	14	settable	SPI1	SPI1 global interrupt	0x0000 005C
8	15	settable	WWDT	WWDT global interrupt	0x0000 0060
9	16	settable	TMR0	TMR0 global interrupt	0x0000 0064
10	17	settable	TMR1	TMR1 global interrupt	0x0000 0068
11	18	settable	I2S_RX	I2S RX global interrupt	0x0000 006C
12	19	settable	I2S_TX	I2S TX global interrupt	0x0000 0070
13	20	settable	USB0	USB0 global interrupt	0x0000 0074
14	21	settable	USB1	USB1 global interrupt	0x0000 0078
15	22	settable	ADC	ADC global interrupt	0x0000 007C
16	23	settable	RTC	RTC global interrupt	0x0000 0080
17			Reserved		0x0000 0084
18	25	settable	CRS	CRS global interrupt	0x0000 0088
19	26	settable	TMR2	TMR2 global interrupt	0x0000 008C
20	27	settable	PVD	PVD global interrupt	0x0000 0090
21	28	settable	CMP0	CMP0 global interrupt	0x0000 0094
22	29	settable	CMP1	CMP1 global interrupt	0x0000 0098
23	30	settable	eFlash_CTRL	eFlash controller global interrupt	0x0000 009C
24	31	settable	DMA0	DMA channel 0 interrupt	0x0000 00A0

Position	Priority	Type of priority	Acronym	Description	Address
25	32	settable	DMA1	DMA channel 1 interrupt	0x0000 00A4
26	33	settable	DMA2	DMA channel 2 interrupt	0x0000 00A8
27	34	settable	DMA3	DMA channel 3 interrupt	0x0000 00AC
28	35	settable	DMA4	DMA channel 4 interrupt	0x0000 00B0
29	36	settable	DMA5	DMA channel 5 interrupt	0x0000 00B4
30	37	settable	DMA6	DMA channel 6 interrupt	0x0000 00B8
31	38	settable	GPIO	GPIO global interrupt	0x0000 00BC

1.5.3 System Timer (SysTick)

The Cortex-M0 includes an integrated system timer, SysTick. SysTick provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used as a Real Time Operating System (RTOS) tick timer or as a simple counter.

- SysTick: 24-bit clear-on-write, decrementing, wrap-on-zero counter.
- Is used as a Real Time Operating System (RTOS) tick timer or as a simple counter.
- When enabled, count down from SysTick Current Value Register (SYST_CVR) to zero, and reload SysTick Reload Value Register (SYST_RVR), then continue decrement.
- When count down to zero, COUNTFLAG=1. COUNTFLAG=0, on reads.
- SYST_CVR value is UNKNOWN on reset.
- SYST_RVR=0, Timer=0. (Disable timer even if Timer is enabled)

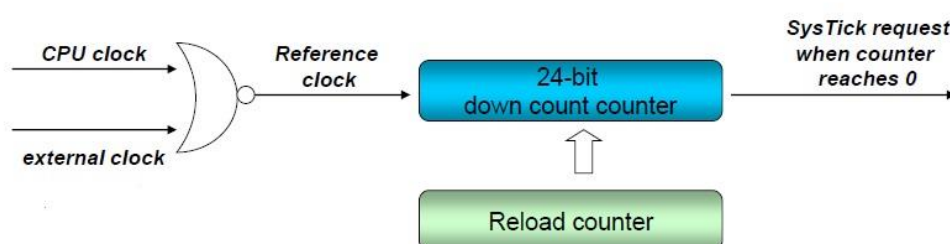
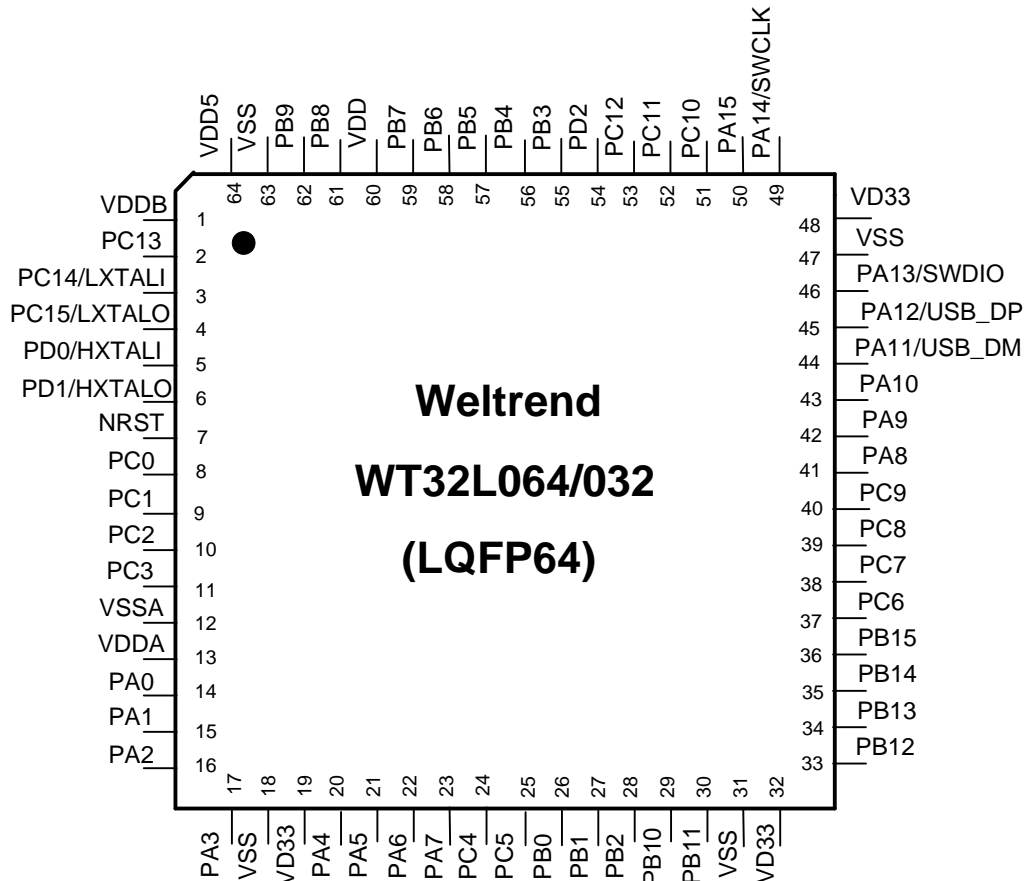


Figure 7 Cortex M0 SysTick

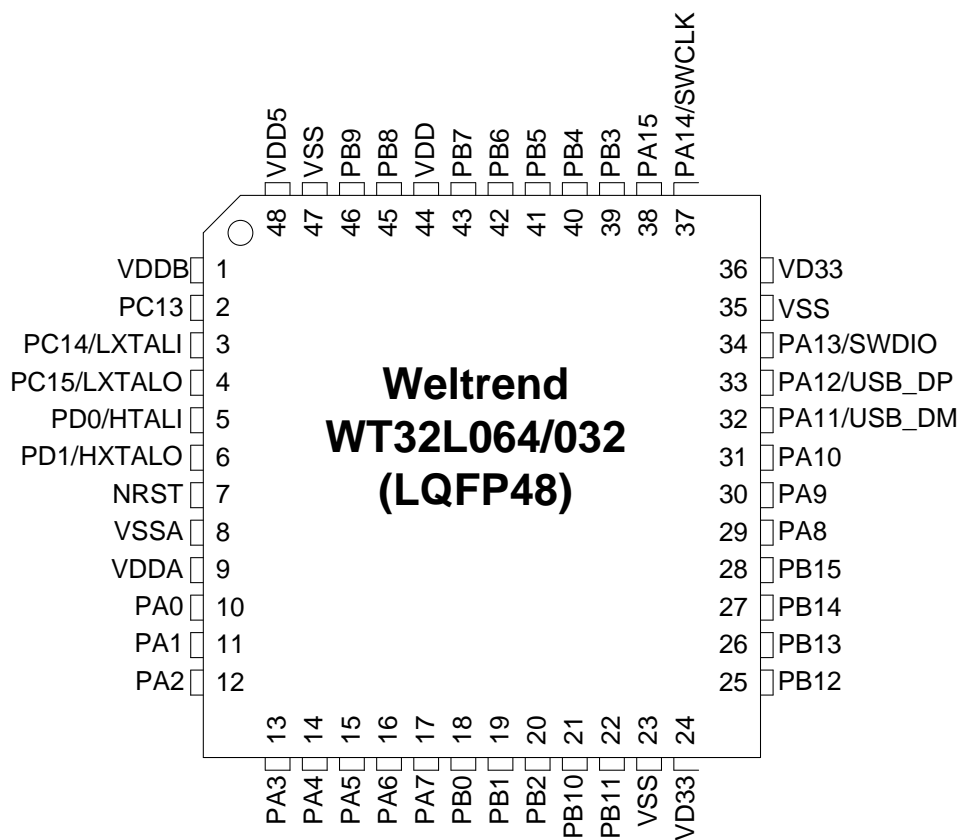
2 Pin Assignment

2.1 Package

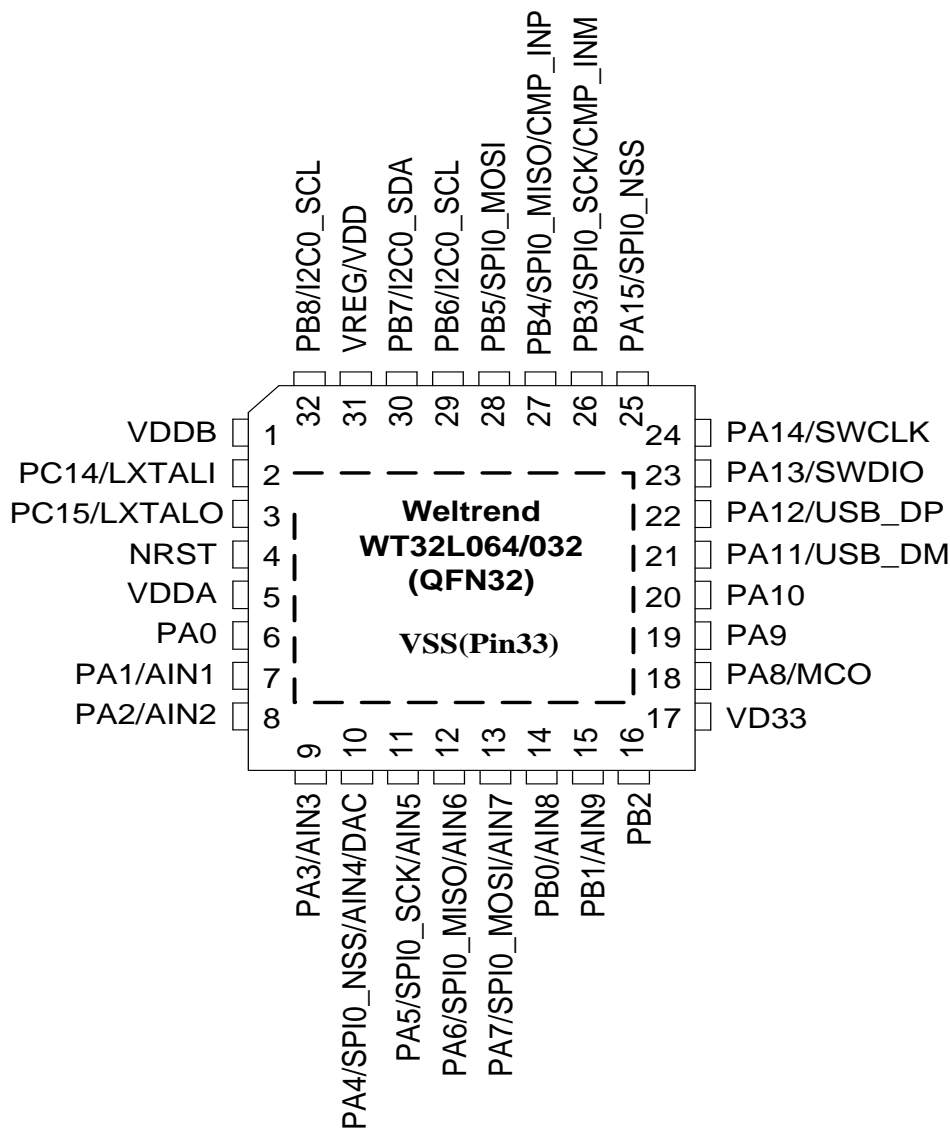
2.1.1 LQFP-64



2.1.2 LQFP-48



2.1.3 QFN-32



2.2 Pin Description

Table 2 Pin Count Table

Function	LQFP-64	LQFP-48	QFN-32
XTAL	2+2	2+2	1+1
ADC	16	10	10
DAC	1	1	1
comparator	2+2	2+2	2+2
USB	1+1	1+1	1+1
analog power	1+1	1+1	1+0
digital power	3+3	3+3	1+0
VBUS power	1+1	1+1	1+1
LDO_CORE	1	1	1
GPIO	51	37	27
serial wire	2	2	2
MCO (X1)	1 (with remap)	1 (with remap)	1 (with remap)
timer CH (X3)	12 (with remap)	12 (with remap)	12
timer ETR (X1)	1 (with remap)	1 (with remap)	1 (with remap)
UART (X2)	2+2 (with remap)	2+2 (with remap)	2+2 (with remap)
SPI (X2)	2+2+2+2 (with remap)	2+2+2+2 (with remap)	2+2+2+2 (with remap)
I2C (X2)	2+2+2 (with remap)	2+2+2 (with remap)	1+1+1
I2S (X1)	1+1+1+1 (with remap)	-	-
PWM (X4)	4 (with remap)	4 (with remap)	4 (with remap)
WKUP	2	2	1
PVD_IN	1	1	1

Table 3 Pin Assignment Table

LQFP 64	LQFP 48	QFN 32	Name	Type	Structure Type	Description
1	1	1	VDDDB	P	—	battery power pin (RTC)
2	2		PC13	I/O	—	WKUP1
3	3	2	PC14	I/O	A	LXTALI
4	4	3	PC15	I/O	A	LXTALO
5	5		PD0	I/O	B	HXTALI
6	6		PD1	I/O	B	HXTALO
7	7	4	NRST	I	C	active low external reset
8			PC0	I/O	A1	AIN10 / I2S_DIN
9			PC1	I/O	A1	AIN11 / I2S_DOUT
10			PC2	I/O	A1	AIN12 / I2S_BCLK
11			PC3	I/O	A1	AIN13 / I2S_LRCLK / PWM01A
12	8	33	VSSA	P	—	analog ground pin
13	9	5	VDDA	P	—	analog power pin
14	10	6	PA0	I/O	A1	AIN0 / TMR1_CH0_ETR / WKUP0 / PWM02A/ COMP0_INM
15	11	7	PA1	I/O	A1	AIN1 / TMR1_CH1 / PWM03A/ COMP0_INP
16	12	8	PA2	I/O	A1	AIN2 / UART1_TX / TMR1_CH2
17	13	9	PA3	I/O	A1	AIN3 / UART1_RX / TMR1_CH3
18		33	VSS	P	—	ground pin
19			VD33	P	—	power pin
20	14	10	PA4	I/O	B	AIN4 / SPI0_NSS / DAC
21	15	11	PA5	I/O	A1	AIN5 / SPI0_SCK
22	16	12	PA6	I/O	A1	AIN6 / SPI0_MISO / TMR2_CH0
23	17	13	PA7	I/O	A1	AIN7 / SPI0_MOSI / TMR2_CH1
24			PC4	I/O	A1	AIN14
25			PC5	I/O	A1	AIN15
26	18	14	PB0	I/O	B	AIN8 / TMR2_CH2/EXT_REF
27	19	15	PB1	I/O	A1	AIN9 / TMR2_CH3
28	20	16	PB2	I/O	A	MCO / PWM00A
29	21		PB10	I/O	A	I2C1_SCL / TMR1_CH2
30	22		PB11	I/O	A	I2C1_SDA / TMR1_CH3
31	23	33	VSS	P	—	ground pin
32	24	17	VD33	P	—	power pin
33	25		PB12	I/O	A	SPI1_NSS / I2C1_SMBA
34	26		PB13	I/O	A	SPI1_SCK
35	27		PB14	I/O	A	SPI1_MISO
36	28		PB15	I/O	A	SPI1_MOSI

LQFP 64	LQFP 48	QFN 32	Name	Type	Structure Type	Description
37			PC6	I/O	A	TMR2_CH0 / I2S_DIN
38			PC7	I/O	A	TMR2_CH1 / I2S_DOUT
39			PC8	I/O	A	TMR2_CH2 / I2S_BCLK
40			PC9	I/O	A	TMR2_CH3 / I2S_LRCLK
41	29	18	PA8	I/O	A	MCO / TMR0_CH0 / PWM01B
42	30	19	PA9	I/O	A	UART0_TX / TMR0_CH1 / PWM02B
43	31	20	PA10	I/O	A	UART0_RX / TMR0_CH2 / PWM03B
44	32	21	PA11	I/O	B	USB_DM / TMR0_CH3
45	33	22	PA12	I/O	B	USB_DP
46	34	23	PA13	I/O	A	SWDIO
47	35	33	VSS	P	—	ground pin
48	36		VD33	P	—	power pin
49	37	24	PA14	I/O	A	SWCLK
50	38	25	PA15	I/O	A	TMR1_CH0_ETR / SPI0_NSS
51			PC10	I/O	A	TMR0_CH0 / I2C1_SCL
52			PC11	I/O	A	TMR0_CH1 / I2C1_SDA
53			PC12	I/O	A	TMR0_CH2
54			PD2	I/O	A	TMR0_CH3
55	39	26	PB3	I/O	A1	TMR1_CH1 / SPI0_SCK / COMP1_INM
56	40	27	PB4	I/O	A1	SPI0_MISO / COMP1_INP
57	41	28	PB5	I/O	A	I2C0_SMBA / SPI0_MOSI
58	42	29	PB6	I/O	A	I2C0_SCL / UART0_TX / PWM00B
59	43	30	PB7	I/O	A1	I2C0_SDA / UART0_RX / PVD_IN
60	44	31	VDD	P	—	regulator core power output, connect to bypass capacitor.
61	45	32	PB8	I/O	A	I2C0_SCL / UART1_TX
62	46		PB9	I/O	A	I2C0_SDA / UART1_RX
63	47	33	VSS	P	—	ground pin
64	48		VDD5	P	—	VBUS power pin

GPIO Structure Type:

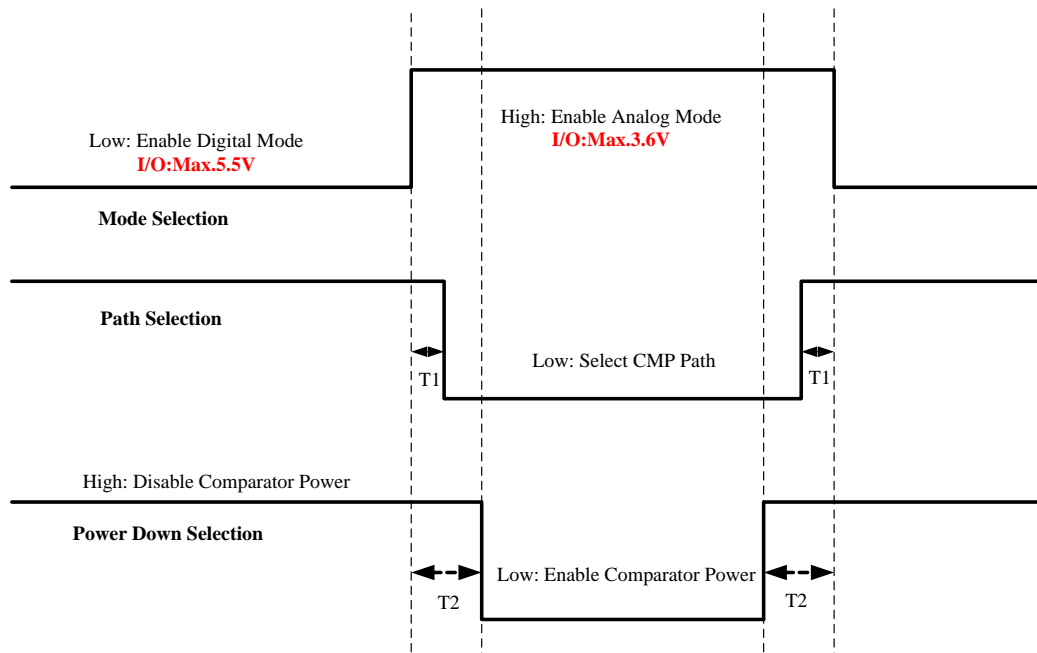
A: 5V Tolerant I/O

B: STD 3.3V I/O

C: Bidirection reset with embedded weak pull-up resistor

A1: 5V Tolerant I/O: Only at Digital Mode but exclude Analog Mode.

- ✓ Most **A1** type default is analog mode (3.3V) during **RESET period**, must avoid to connect 5V application and please choose **A** type I/O if necessary.
- ✓ If used in A1 Type, the timing procedure must follow below comparator example.
(Avoid hardware damage in 3.3V Analog mode caused by 5V input in Digital mode)



2.3 Alternate Function I/O Priority

Name	AF_SEL 0	AF_SEL 1	AF_SEL 2	AF_SEL 3	AF_SEL 4	AF_SEL 5	Analog
PA0					TMR1_CH0_ETR	PWM02A	WKUP0 / AIN0/ COM0_INM
PA1					TMR1_CH1	PWM03A	AIN1/COM0_INP
PA2			UART1_TX		TMR1_CH2		AIN2
PA3			UART1_RX		TMR1_CH3		AIN3
PA4				SPI0_NSS			AIN4 / DAC
PA5				SPI0_SCK			AIN5
PA6				SPI0_MISO	TMR2_CH0		AIN6
PA7				SPI0_MOSI	TMR2_CH1		AIN7
PA8		MCO			TMR0_CH0	PWM01B	
PA9			UART0_TX		TMR0_CH1	PWM02B	
PA10			UART0_RX		TMR0_CH2	PWM03B	
PA11					TMR0_CH3		USB_DM
PA12							USB_DP
PA13	SWDIO						
PA14	SWCLK						
PA15				(SPI0_NSS)	(TMR1_CH0_ETR)		
PB0					TMR2_CH2		AIN8/EXT_REF
PB1					TMR2_CH3		AIN9
PB2		MCO				PWM00A	
PB3				(SPI0_SCK)	(TMR1_CH1)		COMP1_INM
PB4				(SPI0_MISO)			COMP1_INP
PB5		I2C0_SMBA		(SPI0_MOSI)			
PB6		I2C0_SCL	(UART0_TX)			PWM00B	
PB7		I2C0_SDA	(UART0_RX)				PVD_IN
PB8		(I2C0_SCL)	(UART1_TX)				
PB9		(I2C0_SDA)	(UART1_RX)				
PB10		I2C1_SCL			(TMR1_CH2)		
PB11		I2C1_SDA			(TMR1_CH3)		
PB12		I2C1_SMBA		SPI1_NSS			
PB13				SPI1_SCK			
PB14				SPI1_MISO			
PB15				SPI1_MOSI			
PC0		I2S_DIN					AIN10
PC1		I2S_DOUT					AIN11
PC2		I2S_BCLK					AIN12
PC3		I2S_LRCLK				PWM01A	AIN13
PC4							AIN14
PC5							AIN15
PC6		(I2S_DIN)			(TMR2_CH0)		
PC7		(I2S_DOUT)			(TMR2_CH1)		
PC8		(I2S_BCLK)			(TMR2_CH2)		
PC9		(I2S_LRCLK)			(TMR2_CH3)		
PC10		(I2C1_SCL)			(TMR0_CH0)		
PC11		(I2C1_SDA)			(TMR0_CH1)		
PC12					(TMR0_CH2)		
PC13							WKUP1
PC14							LXTALI
PC15							LXTALO
PD0							HXTALI
PD1							HXTALO
PD2					(TMR0_CH3)		

Note: () means lower priority for port inputs

3 Memory Mapping

3.1 AMBA Bus Address Mapping

Index	Function	Description
0x0000_0000~0x1FEF_FFFF	reserved	
0x1000_0000~0x1000_FFFF	embedded flash	64KB
0x1FF0_0000~0x1FF0_07FF	reserved	
0x1FF0_0800~0x1FF0_FFFF	reserved	
0x1FF1_0000~0x1FF1_1FFF	boot ROM	8KB
0x1FF1_2000~0x1FFF_FFFF	reserved	
0x2000_0000~0x2000_1FFF	SRAM	8KB
0x2000_5000~0x2FFF_FFFF	reserved	
0x3000_0000~0x3000_0FFF	DMA buffer	4KB
0x3000_1000~0x3FFF_FFFF	reserved	
0x4000_0000~0x4003_FFFF	APB0 memory space	256KB
0x4004_0000~0x4007_FFFF	APB1 memory space	256KB
0x4008_0000~0x400B_FFFF	AHB memory space	256KB
0x400C_0000~0xDFFF_FFFF	reserved	
0xE000_0000~0xE00F_FFFF	reserved	
0xE010_0000~0xFFFF_FFFF	reserved	
0xF000_0000~0xF000_003FF	System ROM Table	
0xF000_0400~0xFFFF_FFFF	reserved	

3.2 APB Memory Space

Range	Index	Function	Description
APB0	0x4000_0000~0x4000_0FFF		
	0x4000_1000~0x4000_1FFF		
	0x4000_2000~0x4000_2FFF	-	
	0x4000_3000~0x4000_3FFF	-	
	0x4000_4000~0x4000_4FFF	UART1	
	0x4000_5000~0x4000_5FFF	-	
	0x4000_6000~0x4000_6FFF	SPI1	
	0x4000_7000~0x4000_7FFF	-	
	0x4000_8000~0x4000_8FFF	I2C0	
	0x4000_9000~0x4000_9FFF	I2C1	
	0x4000_A000~0x4000_AFFF	-	
	0x4000_B000~0x4000_BFFF	-	
	0x4000_C000~0x4000_CFFF		
	0x4000_D000~0x4000_DFFF	-	
	0x4000_E000~0x4000_EFFF	USB	
	0x4000_F000~0x4000_FFFF	-	
	0x4001_0000~0x4001_0FFF	DAC	
	0x4001_1000~0x4001_1FFF	-	
	0x4001_2000~0x4001_2FFF		
	0x4001_3000~0x4001_3FFF	-	

Range	Index	Function	Description
	0x4001_4000~0x4001_4FFF	TMR2	
	0x4001_5000~0x4001_5FFF	-	
	0x4001_6000~0x4001_6FFF	-	
	0x4001_7000~0x4001_7FFF	-	
	0x4001_8000~0x4001_8FFF	IWDT	
	0x4001_9000~0x4001_9FFF	WWDT	
	0x4001_A000~0x4001_AFFF	PMU	power management unit
	0x4001_B000~0x4001_BFFF	RTC	
	0x4001_C000~0x4001_CFFF	PWM	
	0x4001_D000~0x4001_DFFF	-	
	0x4001_E000~0x4001_EFFF	-	
	0x4001_F000~0x4001_FFFF	-	
APB1	0x4004_0000~0x4004_0FFF	-	
	0x4004_1000~0x4004_1FFF	-	
	0x4004_2000~0x4004_2FFF	-	
	0x4004_3000~0x4004_3FFF	-	
	0x4004_4000~0x4004_4FFF	UART0	
	0x4004_6000~0x4004_6FFF	SPI0	
	0x4004_7000~0x4004_7FFF	-	
	0x4004_8000~0x4004_8FFF	-	
	0x4004_9000~0x4004_9FFF	-	
	0x4004_A000~0x4004_AFFF	-	
	0x4004_B000~0x4004_BFFF	-	
	0x4004_C000~0x4004_CFFF	I2S	
	0x4004_D000~0x4004_DFFF	-	
	0x4004_E000~0x4004_EFFF	-	
	0x4004_F000~0x4004_FFFF	-	
	0x4005_0000~0x4005_0FFF	ADC	
	0x4005_1000~0x4005_1FFF	-	
	0x4005_2000~0x4005_2FFF	-	
	0x4005_3000~0x4005_3FFF	-	
	0x4005_4000~0x4005_4FFF	TMR0	
	0x4005_5000~0x4005_5FFF	TMR1	
	0x4005_6000~0x4005_6FFF	-	
	0x4005_7000~0x4005_7FFF	-	
	0x4005_9000~0x4005_9FFF	CRS	
	0x4005_A000~0x4005_AFFF	-	
	0x4005_B000~0x4005_BFFF	-	
	0x4005_C000~0x4005_CFFF	-	
	0x4005_D000~0x4005_DFFF	-	
	0x4005_E000~0x4005_EFFF	-	
	0x4005_F000~0x4005_FFFF	-	
AHB	0x4008_0000~0x4008_0FFF	FLASH	flash control
	0x4008_1000~0x4008_1FFF	-	
	0x4008_2000~0x4008_2FFF	DMA	
	0x4008_3000~0x4008_3FFF	-	

Range	Index	Function	Description
	0x4008_4000~0x4008_4FFF	RCC	
	0x4008_5000~0x4008_5FFF	-	
	0x4008_6000~0x4008_6FFF	-	
	0x4008_7000~0x4008_7FFF	-	
	0x4008_9000~0x4008_9FFF	-	
	0x4008_A000~0x4008_AFFF	CRC32	
	0x4008_B000~0x4008_BFFF	-	
	0x4008_C000~0x4008_CFFF	GPIO	
	0x4008_D000~0x4008_DFFF	-	
	0x4008_E000~0x4008_EFFF	-	
	0x4008_F000~0x4008_FFFF	-	
	0x5001_F000~0x5001_FFFF	system control	

4 Reset and Clock Control

4.1 Main Features

The Reset and Clock Control (RCC) is used to control miscellaneous resets and clocks of system.

- Reset outputs:
 - Flash reset
 - Global reset
- Clock outputs:
 - AHB, APB0 & APB1 clock
 - ADC clock
 - USB clock
 - Microcontroller Clock Output (MCO)

4.2 Block Diagram

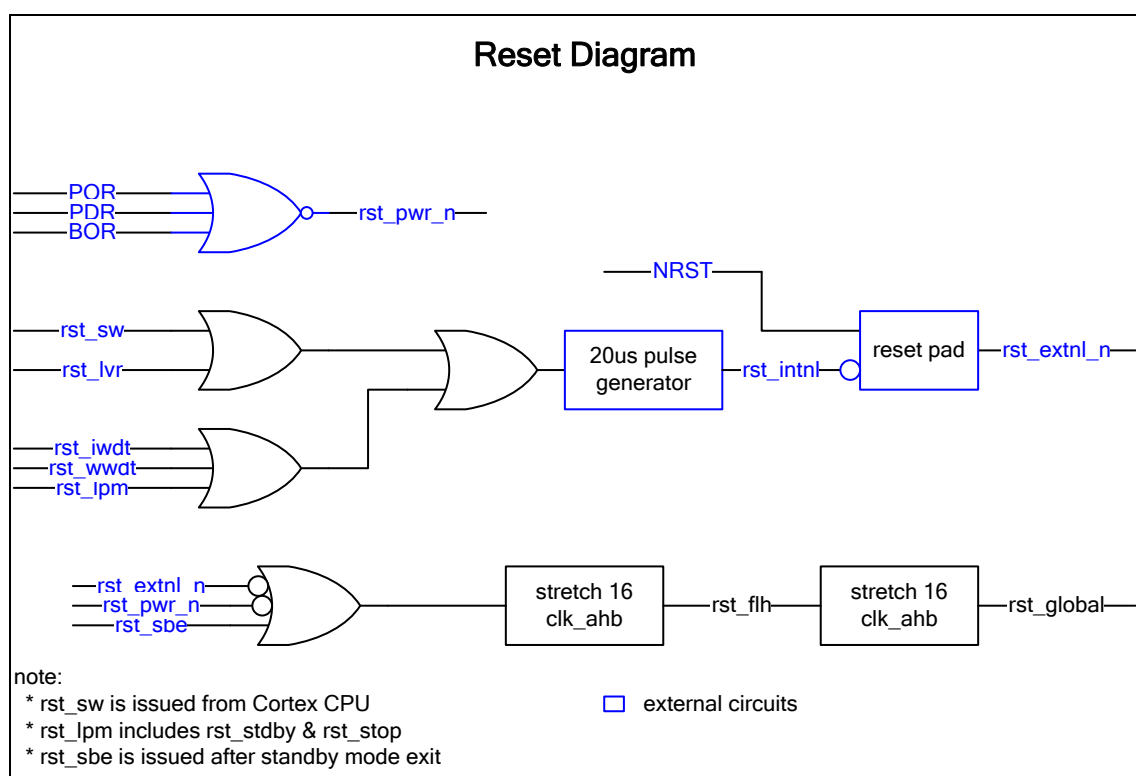


Figure 8 RCC Module Reset Diagram

Clock Diagram

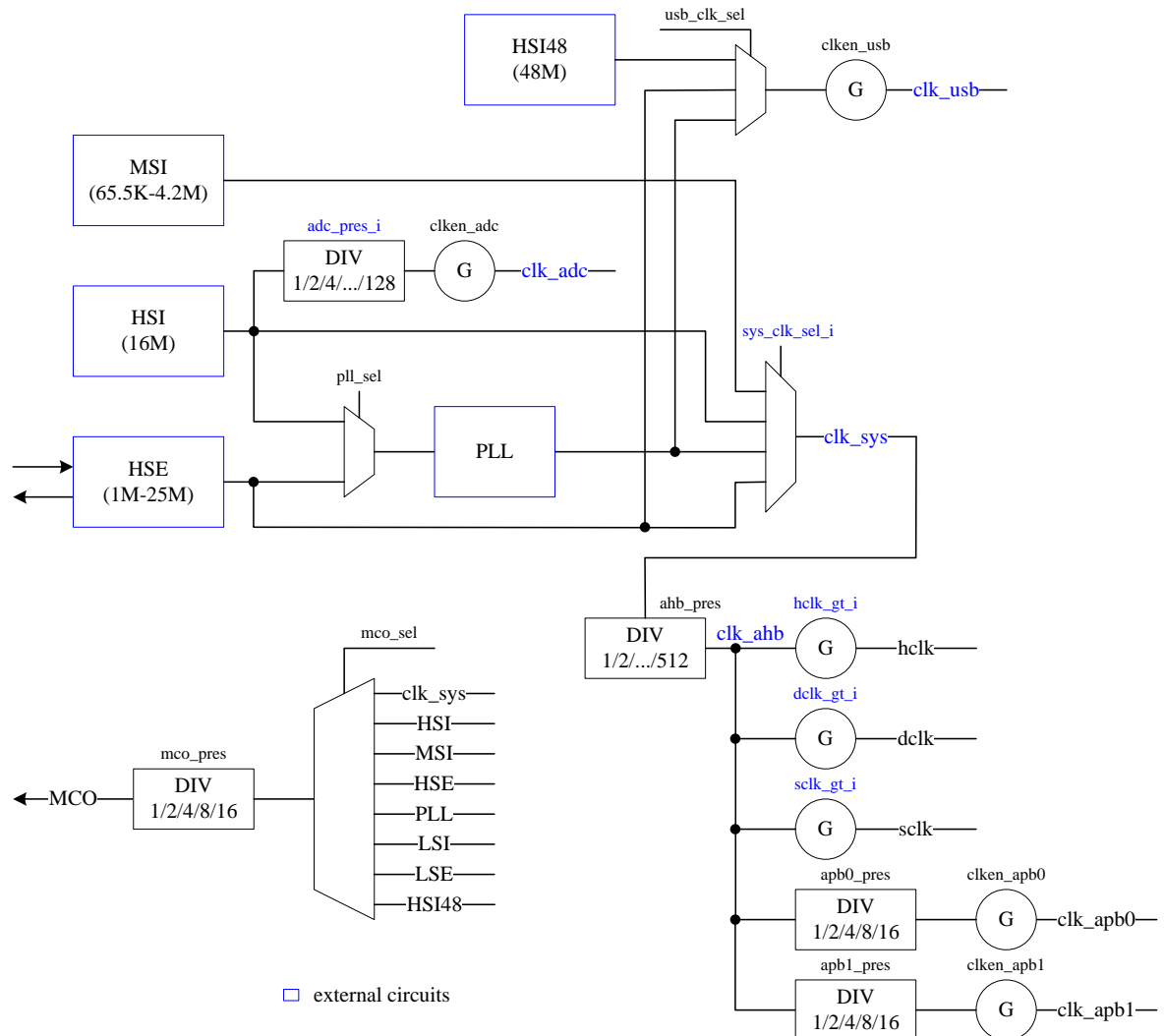


Figure 9 RCC Module Clock Diagram

4.3 Register Definition

Table 4 RCC Control Register

Index	Bit	R/W	Default	Name	Description
RCC_CCR: Clock control settings					
00h	28:24	R/W	03h	pll_prediv	input reference clock pre-divider $F_{int} = F_{in} / (pll_prediv + 1)$
	17:16	R/W	0h	pll_postdiv	the output clock divider $F_{out} = F_{vco} / (2^{**} (pll_postdiv + 1))$
	14:8	R/W	0fh	pll_mul	feedback clock multiplier $F_{fb} = F_{vco} / (pll_mul + 1)$
	0	R/W	0	pll_sel	PLL clock source selection: 0: HSI oscillator clock selected as PLL input clock 1: HSE oscillator clock selected as PLL input clock
RCC_CSCR: Source control settings					
04h	9:8	R/W	0	usb_clk_sel	USB clock selection: 0: HSI48 1: PLL 2: HSE
	1	R/W	1	clken_apb1	APB1 clock enable
	0	R/W	1	clken_apb0	APB0 clock enable
RCC_CCFGR: Source control settings					
08h	30:28	R/W	0	mco_pres	MCO prescaler: $MCO_PAD = MCO / 2^{**} mco_pres$ valid range: 0 - 4
	27:24	R/W	0	mco_sel	microcontroller clock output selection: 0000: MCO output disabled, no clock on MCO 0001: SYSCLK clock selected 0010: HSI oscillator clock selected 0011: MSI oscillator clock selected 0100: HSE oscillator clock selected 0101: PLL clock selected 0110: LSI oscillator clock selected 0111: LSE oscillator clock selected 1000: HSI48 oscillator clock selected
	13:11	R/W	0	apb0_pres	to control the division factor of the APB low speed clock: 0: HCLK not divided 1: HCLK divided by 2 2: HCLK divided by 4 3: HCLK divided by 8 4: HCLK divided by 16
	10:8	R/W	0	apb1_pres	to control the division factor of the APB high speed clock: 0: HCLK not divided 1: HCLK divided by 2 2: HCLK divided by 4 3: HCLK divided by 8 4: HCLK divided by 16
	7:4	R/W	0	ahb_pres	to control the division factor of the AHB clock: 0: SYSCLK not divided 1: SYSCLK divided by 2

Index	Bit	R/W	Default	Name	Description
					2: SYSCLK divided by 4 3: SYSCLK divided by 8 4: SYSCLK divided by 16 5: SYSCLK divided by 32 6: SYSCLK divided by 64 7: SYSCLK divided by 128 8: SYSCLK divided by 256 9: SYSCLK divided by 512
RCC_RAPB0R: Module Resets for APB0					
10h	28	R/W	0	rst_pwm	PWM reset: 0: no effect 1: PWM reset
	27	R/W	0	rst_rtc	RTC reset, excluded by global reset
	25	R/W	0	rst_wwdt	WWDT reset
	24	R/W	0	rst_iwdt	IWDT reset
	20	R/W	0	rst_tmr2	timer-2 reset
	16	R/W	0	rst_dac	DAC reset
	14	R/W	0	rst_usb	USB reset
	9	R/W	0	rst_i2c1	I2C1 reset
	8	R/W	0	rst_i2c0	I2C0 reset
	6	R/W	0	rst_spi1	SPI1 reset
	4	R/W	0	rst_uart1	UART1 reset
	1	-	-	-	Reserved
	0	-	-	-	Reserved
RCC_RAPB1R: Module Resets for APB1					
14h	24				Reserved
	21	R/W	0	rst_tmr1	timer-1 reset
	20	R/W	0	rst_tmr0	timer-0 reset
	16	R/W	0	rst_adc	ADC reset
	12	R/W	0	rst_i2s	I2S reset
	6	R/W	0	rst_spi0	SPI0 reset
	5				Reserved
	4	R/W	0	rst_uart0	UART0 reset
RCC_RAHBR: Module Resets for AHB					
18h	12	R/W	0	rst_gpio	GPIO reset
	10	R/W	0	rst_crc32	CRC32 reset
	8				Reserved
	2	R/W	0	rst_dma	DMA reset
RCC_CEAPB0R: Module clock enables for APB0					
20h	28	R/W	0	clken_pwm	PWM clock enable: 0: PWM clock disable 1: PWM clock enable
	27	R/W	0	clken_rtc	RTC clock enable
	26	R/W	1	clken_pmu	PMU clock enable
	25	R/W	0	clken_wwdt	WWDT clock enable
	24	R/W	0	clken_iwdt	IWDT clock enable
	20	R/W	0	clken_tmr2	Timer-2 clock enable
	16	R/W	0	clken_dac	DAC clock enable
	14	R/W	0	clken_usb	USB clock enable
	12	R/W	0	clken_intc	Interrupt controller clock enable
	9	R/W	0	clken_i2c1	I2C1 clock enable

Index	Bit	R/W	Default	Name	Description
	8	R/W	0	clken_i2c0	I2C0 clock enable
	6	R/W	0	clken_spi1	SPI1 clock enable
	4	R/W	0	clken_uart1	UART1 clock enable
	1	-	-	-	Reserved
	0	-	-	-	Reserved
RCC_CEAPB1R: Module clock enables for APB1					
24h	25	R/W	0	clken_crs	CRS clock enable: 0: CRS clock disable 1: CRS clock enable
	24				Reserved
	21	R/W	0	clken_tmr1	Timer-1 clock enable
	20	R/W	0	clken_tmr0	Timer-0 clock enable
	16	R/W	0	clken_adc	ADC clock enable
	12	R/W	0	clken_i2s	I2S clock enable
	6	R/W	0	clken_spi0	SPI0 clock enable
	5				Reserved
	4	R/W	0	clken_uart0	UART0 clock enable
RCC_CEAHBR: Module clock enables for AHB					
28h	15	R/W	0	clken_sys	SYS clock enable: 0: SYS clock disable 1: SYS clock enable
	12	R/W	0	clken_gpio	GPIO clock enable
	10	R/W	0	clken_crc32	CRC32 clock enable
	8				Reserved
	2	R/W	0	clken_dma	DMA clock enable
	0	R/W	1	clken_fih_prog	Flash programmer clock enable
RCC_LCEAPB0R: Low power module clock enables for APB0					
30h	28	R/W	0	lpclken_pwm	PWM low power clock enable: 0: PWM low power clock disabled 1: PWM low power clock enabled
	27	R/W	0	lpclken_rtc	RTC low power clock enable 0: RTC low power clock disable 1: RTC low power clock enable
	26	R/W	1	lpclken_pmu	PMU low power clock enable
	25	R/W	0	lpclken_wwdt	WWDT low power clock enable
	24	R/W	0	lpclken_iwdt	IWDT low power clock enable
	20	R/W	0	lpclken_tmr2	Timer-2 low power clock enable
	16	R/W	0	lpclken_dac	DAC low power clock enable
	14	R/W	0	lpclken_usb	USB low power clock enable
	12	R/W	0	lpclken_intc	interrupt controller low power clock enable
	9	R/W	0	lpclken_i2c1	I2C1 low power clock enable
	8	R/W	0	lpclken_i2c0	I2C0 low power clock enable
	6	R/W	0	lpclken_spi1	SPI1 low power clock enable
	4	R/W	0	lpclken_uart1	UART1 low power clock enable
	1	-	-	-	Reserved
	0	-	-	-	Reserved
RCC_LCEAPB1R: Low power module clock enables for APB1					
34h	25	R/W	0	lpclken_crs	CRS low power clock enable: 0: CRS low power clock disable 1: CRS low power clock enable
	24				Reserved

Index	Bit	R/W	Default	Name	Description
	21	R/W	0	lpclken_tmr1	Timer-1 low power clock enable
	20	R/W	0	lpclken_tmr0	Timer-0 low power clock enable
	16	R/W	0	lpclken_adc	ADC low power clock enable
	12	R/W	0	lpclken_i2s	I2S low power clock enable
	6	R/W	0	lpclken_spi0	SPI0 low power clock enable
	5				Reserved
	4	R/W	0	lpclken_uart0	UART0 low power clock enable
RCC_LCEAHBR: Low power module clock enables for AHB					
38h	15	R/W	0	lpclken_sys	SYS low power clock enable: 0: SYS low power clock disable 1: SYS low power clock enable
	12	R/W	0	lpclken_gpio	GPIO low power clock enable
	10	R/W	0	lpclken_crc32	CRC32 low power clock enable
	8				Reserved
	2	R/W	0	lpclken_dma	DMA low power clock enable
	0	R/W	1	lpclken_fih_prog	Flash programmer low power clock enable
RCC_CSR: Reset flag					
40h	31	R/W1c	0	rstf_lpm	Low power management reset flag
	30	R/W1c	0	rstf_wwdt	WWDT reset flag
	29	R/W1c	0	rstf_iwdt	IWDT reset flag
	28	R/W1c	0	rstf_lockup	Lockup reset flag
	27	R/W1c	1	rstf_pwr	PWR (POR/PDR/BOR) reset flag
	26	R/W1c	0	rstf_lvr	LVR reset flag
	25	R/W1c	0	rstf_sbe	Standby-exit reset flag
	24	R/W1c	0	rstf_sys	System reset flag, SW reset flag
	23	R/W1c	0	rstf_ext	External reset flag

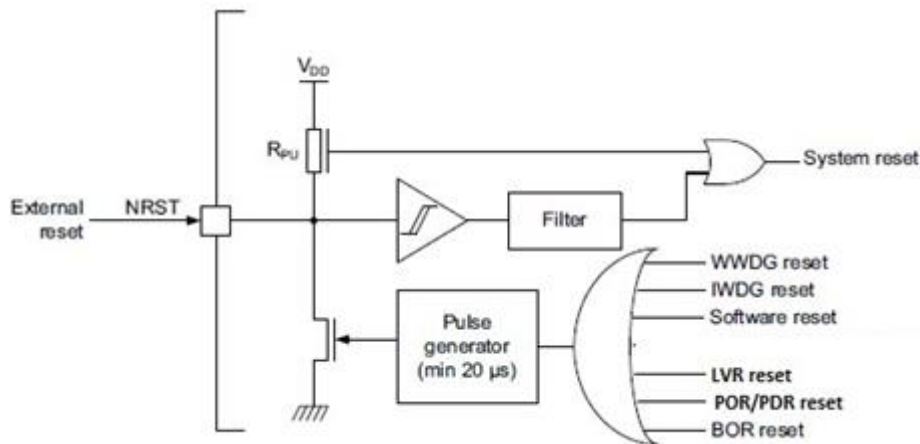
R/W1C: read & write one clear

- Notice: Before setting RTC function, **RTC clock** needs to be enabled or else system/IC will fail due to no APB0 clock.

4.4 Functional Description

4.4.1 Reset

Three types of reset were contained in RCC module, defined as System Reset, Power Reset and Standby Exit Reset.



4.4.1.1 System Reset

A system reset is generated when one of the following events occurs:

1. A low level on the RST_N pin (External Reset)
2. Window Watchdog end-of-count condition (WWDG Reset)
3. Independent Watchdog end-of-count condition (IWDG Reset)
4. Software reset (SW reset)
5. LVR reset

These sources act on the NRST pin and it is always kept low during the delay phase. The RESET service routine vector is fixed at address 0x0000_0004 in the memory map. The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 μ s for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

Software Reset

The SYSRESETREQ bit in Cortex®-M0 Application Interrupt and Reset Control Register must be set to force a software reset on the device. Please refer to the Cortex®-M0 technical reference manual for more details.

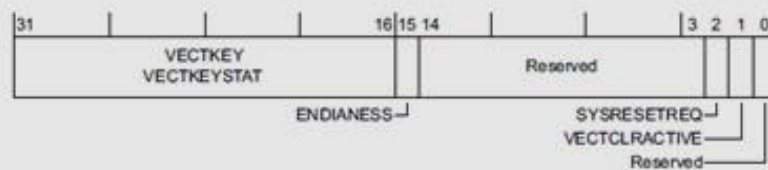
Application Interrupt and Reset Control Register, AIRCR

The AIRCR Register characteristics are:

Purpose Sets or returns interrupt control data.

Usage constraints There are no usage constraints.

Configurations Always implemented.



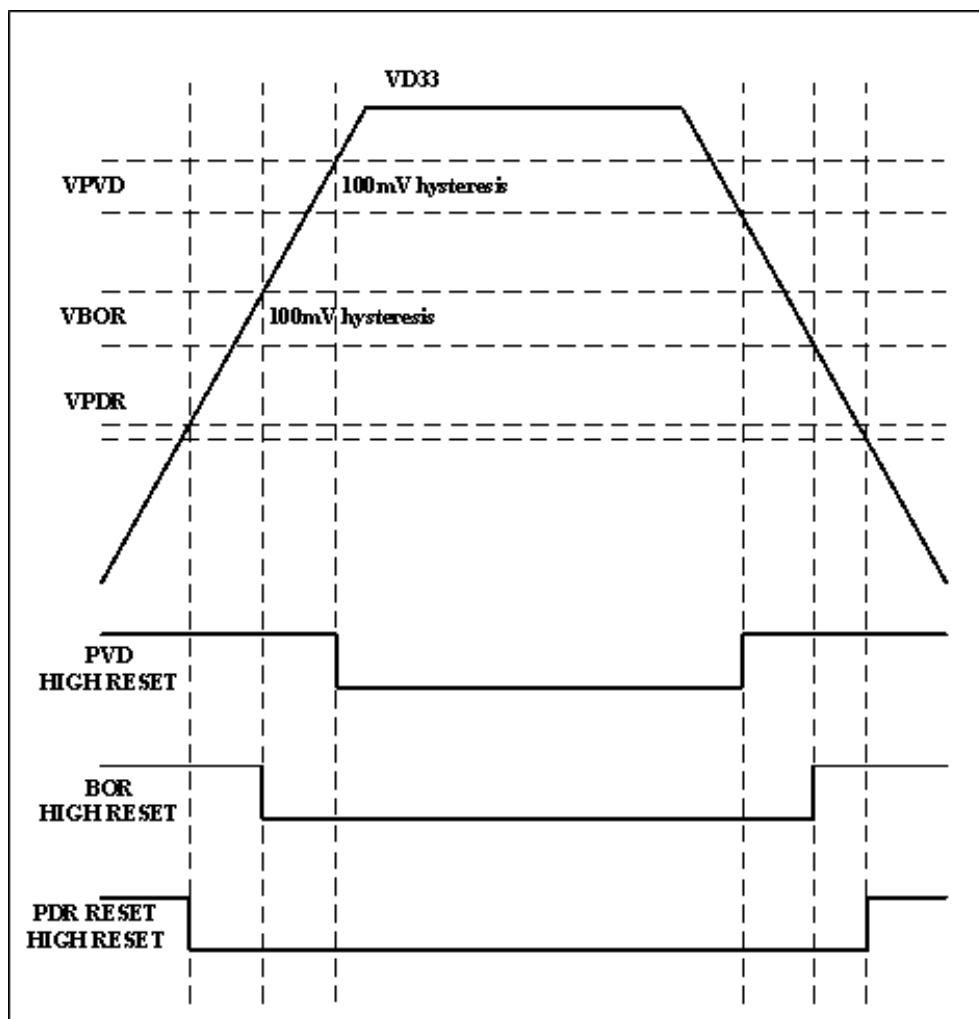
Bits	Type	Name	Function
[2]	WO	SYSRESETREQ	<p>System Reset Request:</p> <p>0 do not request a reset.</p> <p>1 request reset.</p> <p>Writing 1 to this bit asserts a signal to request a reset by the external system. The system components that are reset by this request are IMPLEMENTATION DEFINED. A Local reset is required as part of a system reset request.</p> <p>A Local reset clears this bit to 0.</p>

4.4.1.2 Power Reset

A power reset is generated when one of the following events occurs:

1. Power-on/power-down reset (POR/PDR reset)
2. BOR reset

A power reset sets all registers to their reset values.



1. PVD is enabled or disabled by software.
2. BOR is enabled of operating from 1.8V to 3.6V.
3. PDR reset level is around at 1.0V.
4. There is no BOR when **VD33** operating from 1.65V to 3.6V and the reset is enabled when **VD33** goes below PDR level.
5. BOR reset level setting: refer to PMU chapter 16.6.
6. PVD setting: please refer to Chapter 16.6 PMU section for more details.

4.4.1.3 Standby Exit Reset

A Standby Exit Reset is generated when system is resumed from standby mode.

4.4.2 Clock

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI ((high-speed internal) oscillator clock
- HSE (high-speed external) oscillator clock
- PLL clock
- MSI (multispeed internal) oscillator clock

The MSI is used as system clock source after startup from reset, wake-up from stop or standby low-power modes.

The devices have the following two secondary clock sources:

- 37 kHz low speed internal RC (LSI RC) which drives the independent Watchdog
- 32.768 kHz low speed external crystal (LSE crystal) which drives the real-time clock (RTCCLK)

Each clock source can be switched on or off independently when it is not used, to optimize power consumption. Several prescalers allow the configuration of the different clock sources.

4.4.2.1 PLL Clock

The internal PLL can be clocked by the HSI RC or HSE crystal. It is used to drive the system clock and to generate the 48 MHz clock for the USB peripheral. The PLL input clock frequency must be between 2 MHz and 24 MHz. The desired frequency is obtained by using the input division, multiplication factor and output division embedded in the PLL.

The PLL configuration (selection of the source clock, output division factor, multiplication factor and output division factor) must be performed before enabling the PLL. Once the PLL is enabled, these parameters cannot be changed.

PLL Diagram

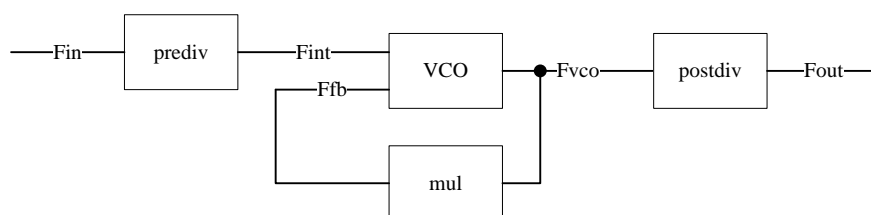


Figure 10 PLL Diagram

Notes:

1. $8\text{MHz} \leq F_{vco} \leq 192\text{MHz}$, if $V_{DDA} \geq 1.8\text{V}$; $8\text{MHz} \leq F_{vco} \leq 96\text{MHz}$, if $1.65\text{V} \leq V_{DDA} < 1.8\text{V}$;
2. $F_{out} = F_{in} * (pll_mul + 1) / (pll_prediv + 1) / (2 ** (pll_postdiv + 1))$
3. $F_{vco} = F_{in} * (pll_mul + 1) / (pll_prediv + 1)$.

5 Power Manage Unit

5.1 Power Management Unit Features

Power management unit (PMU) controls all power of core and peripheral components, and it also controls source of system clock and frequency range of MSI. The device has an integrated zeropower power-on reset (POR)/power-down reset (PDR), coupled with a brown out reset (BOR) circuitry. (Refer to BOR) The device features an embedded programmable voltage detector (PVD) that monitors the VDD/VDDA power supply and compares it to the VPD threshold. Seven different PVD levels can be selected by software between 1.85V and 3.05 V, with a 200 mV step. (Please refer to PVD)

- Three different voltage regulator modes:
 - Main (MR)
 - Low-power (LPR)
 - Power-down
- Dynamic core voltage:
 - 1.8V
 - 1.2V
- Five power saving modes:
 - Low-power run mode (LPR)
 - Sleep mode (MR)
 - Stop mode (MR)
 - Low -power stop mode (LPR)
 - Standby mode (Power-down)

5.2 Power Domain Overview

The device can be divided into two blocks and it is provided by two voltages. Core voltage can be powered off in Standby mode as the following figure 11.

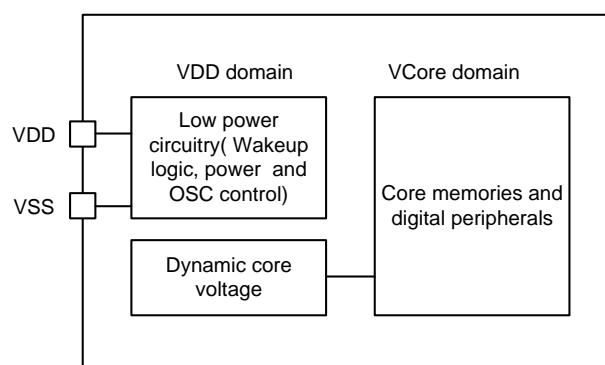


Figure 11 Power Domain Overview

5.3 Voltage Regulator

An embedded linear voltage regulator supplies all the digital circuitries except for the Standby circuitry. The regulator output voltage (V_{CORE}) can be programmed by software to two different ranges within 1.2V or 1.8V (see section Dynamic core voltage).

The voltage regulator is always enabled after Reset. It works in three different modes: Main (MR), Low-Power (LPR) and Power-Down, depending on the application modes.

- **Run mode**, the regulator is in main (MR) mode and it supplies full power to the V_{CORE} domain (core, memories and digital peripherals).
- **Low-power run mode**, the regulator is in low-power (LPR) mode and supplies low-power to the V_{CORE} domain, preserving the contents of the registers and internal SRAM.
- **Sleep mode**, the regulator is in main (MR) mode and it supplies full power to the V_{CORE} domain, preserving the contents of the registers and internal SRAM. Disable system clock but preserve peripheral clock for NVIC.
- **Stop mode**, the regulator is in main (MR) mode and it supplies full power to the V_{CORE} domain, preserving the content of registers and internal SRAM. All internal clocks powered off, only wake up by extended interrupt (EXTI). (Refer section EXTI)
- **Low-power stop mode**, the regulator is in low-power (LPR) mode and it supplies low-power to the V_{CORE} domain, preserving the contents of the registers and internal SRAM. All internal clocks powered off, only wake up by extended interrupt (EXTI). (Refer section EXTI).
- **Standby mode**, the regulator is powered off. The content of the registers and SRAM are lost except for the Standby circuitry

5.4 Dynamic Core Voltage

The dynamic voltage scaling is a power management technique which consists in increasing or decreasing the voltage used for the digital peripherals (V_{CORE}), according to the circumstances.

Range 1

Range 1 is the “CPU performance” range. The voltage regulator outputs a 1.8 V voltage as long as the V_{DD} input voltage is above 2.3 V. Flash program and erase operations can be performed in this range.

Range 2

Range 2 is the “power performance” range. The voltage regulator outputs a 1.2 V voltage without any limitations on V_{DD} but system clock can't exceed 4.2MHz.

Please refer to Table 5 for details on the performance for each range.

Table 5 Performance versus V_{CORE} Ranges

CPU performance	Power performance	Vcore Range	Typical Value (V)	Max frequency (MHz)	VDD range
High	Low	1	1.8	System Clock _(MAX)	2.3V-VDD _{MAX}
Low	High	2	1.2	4.2	VDD _{MIN} -VDD _{MAX}

➔ If $VD33 \leq 2.2V$, this unit will automatically switch V_{CORE} power from 1.8V to 1.2V. Therefore, the user needs to use PVD (programmable voltage detector) to monitor VD33, and once VD33 drops below 2.2V, the clock source must be switched to lower frequency MSI (4.2MHz) by software in advance.

➔ While VD33 was operated in high speed clock and alternatively turns on/off rapidly, user must turn on BOR to prevent MCU failure caused by $VD33 \leq 2.2V$.

5.5 Power Saving Mode

By default, the microcontroller is in Run mode after a system or a power-on reset. In Run mode, the CPU is clocked by HCLK and the program code is executed. Several low-power modes are available to save power when the CPU does not need to be kept running, for example when waiting for an external event. It is up to the user to select the mode that gives the best compromise between low-power consumption, performance, short startup time and available wakeup sources.

The devices feature five power saving modes:

- **Low-power run mode:** regulator in low-power mode, limited clock frequency, limited number of peripherals running.
- **Sleep mode:** Cortex®-M0 core stopped, peripherals kept running.
- **Stop mode:** all clocks are stopped, regulator running
- **Low-power stop mode:** all clocks are stopped, regulator in low-power mode
- **Standby mode:** V_{CORE} domain powered off

In addition, the power consumption in Run mode can be reduced by one of the following means:

- Slowing down the system clocks
- Gating the clocks to the APBx and AHBx peripherals when they are unused.

Table 6 Summary of Saving Power Modes

Mode name	Entry	Wakeup	Effect on V _{CORE} domain clocks	Effect on V _{DD} domain clocks	Voltage regulator
Low-power run	bldo_pd and LPRUN bits + Clock setting	The regulator is forced in Main regulator (1.8V)	None	None	In low-power mode
Sleep mode	WFI or Return from ISR	Any interrupt	CPU CLK OFF no effect on other clocks or analog clock sources	None	ON
	WFE	Wakeup event			
Stop mode	SLEEPDEEP bit + WFI or Return from ISR or WFE	Any extended interrupt (EXTI) and IWDG reset	All V _{CORE} domain clocks OFF	HSI16 ⁽¹⁾ , HSE and MSI oscillators OFF	ON
Low-power stop mode	LPDSR bits + SLEEPDEEP bit + WFI or Return from ISR or WFE				In low-power mode
Standby mode	PDDS bits + SLEEPDEEP bit + WFI or Return from ISR or WFE				OFF

5.5.1 Extended Interrupt (EXTI) for wake up

The EXTI allows the management of up to more than 20 event lines which can wake up the device from Stop mode.

When using I/O Falling or Rising Edge trigger to wake up in STOP Mode, please pay attention to the following two points to avoid abnormalities, or you can use Falling & Rising Edges trigger or level trigger instead:

1. Falling edge trigger: If the I/O is already in the low level and enters the STOP mode on time, the I/O can no longer generate falling edge and cannot be woken up.
2. Rising edge trigger: If the I/O is already in the high level and enters the STOP mode on time, the I/O can no longer generate rising edge and cannot be woken up.

Table 7 EXTI Lines Connections

EXTI line	Line source
each port 0-15	All GPIO
16	PVD
17	RTC alarm
18	RTC periodic timer
19	COMP output

5.5.2 Behavior of Clocks in Power Saving Modes

APB peripheral clocks can be disabled by software.

Sleep modes

The CPU clock is stopped in Sleep modes. The memory interface clocks (Flash memory and RAM interfaces) and all peripherals clocks can be stopped by software during Sleep. The AHB to APB Bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.

Stop and Standby modes

The system clock and all high speed clocks are stopped in Stop and Standby modes:

- PLL is disabled
- Internal RC 16 MHz (HSI16) oscillator is disabled, except if hsi16keron bit is set in Stop Mode
- External 1-24 MHz (HSE) oscillator is disabled
- Internal 65 kHz - 4.2 MHz (MSI) oscillator is disabled

When exiting this mode by an interrupt (Stop mode), the internal MSI or HSI16 can be selected as system clock by PMU control bit (stopwuck). For both oscillators, their individual configuration (range and trimming) value is kept on Stop mode exit.

When exiting this mode by a reset (Standby mode), the individual MSI oscillator is selected as system clock. The range and the trimming value are reset to the default 2.1 MHz.

If a Flash program operation or an access to APB domain is ongoing, the Stop/Standby mode entry is delayed until the Flash memory or the APB access has completed.

Before entering STOP Mode, please turn off all TIMERS and SYSTICK, and then turn them on again after waking up to avoid abnormal wake-up.

5.6 Register Definition

Table 8 PMU Control Register

Index	Bit	R/W	Default	Name	Description
PMU_EN: PMU Enable					
00h	[0]	R/W	1	pmu_enable	0: Disable PMU for saving more power 1: Enable PMU
PMU_PVD_IE: Interrupt signals					
04h	[1]	R/W	1	pvd_int_en	Enable PVD interrupt 0: Disable 1: Enable
	[0]	R	0	pvd_int	PVD interrupt flag
PMU_RGLTR_CR: Regulator control signals					
08H	[8]	R/W	1	ch1p2v_n	Core power selection. 0: 1.2V 1: 1.8V
	[5]	R/W	0	Lpsdsr	Disable BLDO when enter into sleep mode or stop mode
	[4]	R/W	0	Pdds	When MCU assert deep sleep, the status goes into 1: Standby mode; 0: Stop mode
	[1]	R/W	0	sldo_pd	Low power regulator power down 0: Power up 1: Power down
	[0]	R/W	0	bldo_pd	Main regulator power down 0: Power up 1: Power down
PMU_VD_CR: Voltage detector control signals					
0Ch	[10:8]	R/W	000	bor_sel	BOR level selection [000]: setting VBOR0=1.7V [001]: setting VBOR1=1.93V [010]: setting VBOR2=2.3V [011]: setting VBOR3=2.55V [100]: setting VBOR4=2.8V
	[6:4]	R/W	000	pls_sel	PVD level selection [000]: setting VPVD0=1.85V [001]: setting VPVD1=2.04V [010]: setting VPVD2=2.24V [011]: setting VPVD3=2.44V [100]: setting VPVD4=2.64V [101]: setting VPVD5=2.83V [110]: setting VPVD6=3.05V [111]: setting VPVD7: External input analog voltage (Compare internally to VBGC)
	[3:2]	R/W	11	cmp_hs	Select the operation mode of the CMP High Speed Mode: 1 Low Speed Mode: 0 cmp_hs[1]: CMP2 cmp_hs[0]: CMP1
	[1]	R/W	1	bor_pd	BOR power down 0: Power up

Index	Bit	R/W	Default	Name	Description
					1: Power down
	[0]	R/W	1	pvd_pd	PVD power down 0: Power up 1: Power down
PMU_AN_PD: Analog power control					
14h	[11]	R/W	1	dac_pd	DAC power down 0: Power up 1: Power down
	[10]	R/W	1	r2r_pd	Rail-to-Rail power down 0: Power up 1: Power down
	[9]	R/W	1	adc_pd	ADC power down 0: Power up 1: Power down
	[8:7]	R/W	1	cmp_pd	CMP1 & CMP2 power down [7]:CMP1,[8]:CMP2 0: Power up 1: Power down
	[6]	R/W	0	flash_keep_off	Flash keep power down when exit from Stop mode
	[5]	R/W	0	run_pd_flg	Flash power down when exit sleep mode to run mode
	[4]	R/W	0	sleep_pd_flg	Flash power down when enter into sleep mode
	[1]	R/W	0	rom_pd	ROM power down 0: Power up 1: Power down
	[0]	R/W	0	flash_pd	Flash power down 0: Power up 1: Power down
PMU_CLK_PD: Clock power signals					
18h	[5]	R/W	1	usb48m_pd	USB 48M power down 0: Power up 1: Power down
	[4]	R/W	1	hse_pd	HSE power down 0: Power up 1: Power down
	[3]	R/W	1	pll_pd	PLL power down 0: Power up 1: Power down
	[2]	R/W	1	hsi_pd	HSI power down 0: Power up 1: Power down
	[1]	R/W	1	lsi_pd	LSI power down 0: Power up 1: Power down
	[0]	R/W	0	msi_pd	MSI power down 0: Power up 1: Power down
PMU_CLK_CR: Clock control signals					
1Ch	[11]	R/W	0	dclk_gt	1: Disable debug mode DCLK for power saving 0: Enable debug mode DCLK, no power saving
	[9]	R/W	0	stopwuck	when exit from Stop mode 1: Trun on HSI; 0:Trun on MSI

Index	Bit	R/W	Default	Name	Description
	[8]	R/W	0	hsi16keron	Keep HSI power on when enter into STOP mode
	[6:4]	R/W	101	msi_range	Frequency output select pin 3'b000: range0 65.5K 3'b001: range1 131K 3'b010: range2 262K 3'b011: range3 524K 3'b100: range4 1.05M 3'b101: range5 2.1M 3'b110: range6 4.2M 3'b111: range6 4.2M
	[1:0]	R/W	00	sw_sysclk	select system clock. 00: MSI 01: HSI16 10: HSE 11: PLL
PMU_WKUP_CR: Wakeup control signals					
20h	[3]	R/W	0	rst_stop	generated reset when entering the Stop mode
	[2]	R/W	0	rst_stdby	generated reset when entering the Standby mode
	[1]	R/W	0	wukp1_en	Wakeup pin 1 enable
	[0]	R/W	0	wukp0_en	Wakeup pin 0 enable
PMU_CSR: Status Flag					
24h	[4]	R/W	0	boot_sel	Boot select
	[1]	R/W1C	0	stop_flag	Stop flag if wakeup from Stop, write 1to clear
	[0]	R/W1C	0	stby_flag	Standby flag if wakeup from Standby, write 1to clear
PMU_ATPD_STOP: Auto power down at Stop					
28h	[8]	R/W	0	stop_lvr22_pd	set lvr22_pd when PMU going to stop mode
	[7]	R/W	0	stop_lvr18_pd	set lvr18_pd when PMU going to stop mode
	[4]	R/W	1	stop_r2r_pd	set r2r_pd when PMU going to stop mode
	[3]	R/W	1	stop_dac_pd	set dac_pd when PMU going to stop mode
	[2]	R/W	1	stop_adc_pd	set adc_pd when PMU going to stop mode
	[1]	R/W	1	stop_cmp_pd	set cmp_pd when PMU going to stop mode
	[0]	R/W	1	stop_lsi_pd	set lsi_pd when PMU going to stop mode
PMU_ATPD_STBY: Auto power down at Standby					
2Ch	[11]	R/W	0	stby_pdvbg_dig	set pdvbg_dig when PMU going to standby mode
	[10]	R/W	0	stby_bor_pd	set bor_pd when PMU going to standby mode
	[9]	R/W	0	stby_pvd_pd	set pvd_pd when PMU going to standby mode
	[8]	R/W	0	stby_lvr22_pd	set lvr22_pd when PMU going to standby mode
	[7]	R/W	0	stby_lvr18_pd	set lvr18_pd when PMU going to standby mode
	[6]	R/W	0	stby_pdr_v18_pd	set pdr_v18_pd when PMU going to standby mode
	[5]	R/W	0	stby_pdr_vda_pd	set pdr_vda_pd when PMU going to standby mode
	[4]	R/W	1	stby_r2r_pd	set r2r_pd when PMU going to standby mode
	[3]	R/W	1	stby_dac_pd	set dac_pd when PMU going to standby mode
	[2]	R/W	1	stby_adc_pd	set adc_pd when PMU going to standby mode
	[1]	R/W	1	stby_cmp_pd	set cmp_pd when PMU going to standby mode
	[0]	R/W	1	stby_lsi_pd	Set lsi_pd when PMU going to standby mode

R/W1C: read & write one clear

5.7 State Machine

5.7.1 Run mode

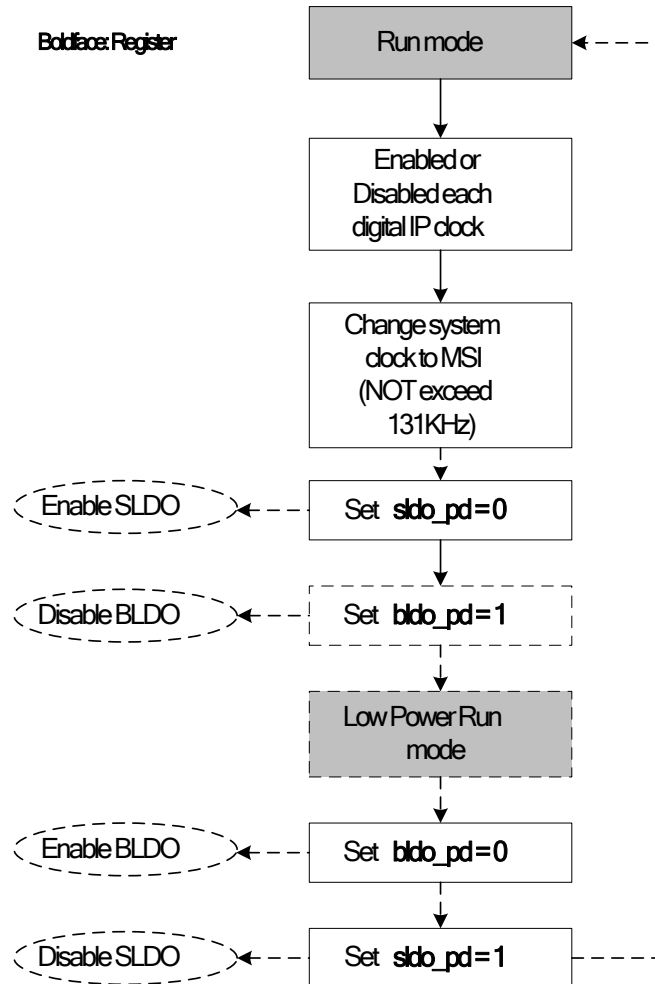


Figure 12 Low Power Run mode state machine

5.7.2 Sleep Mode

Boldface: Register

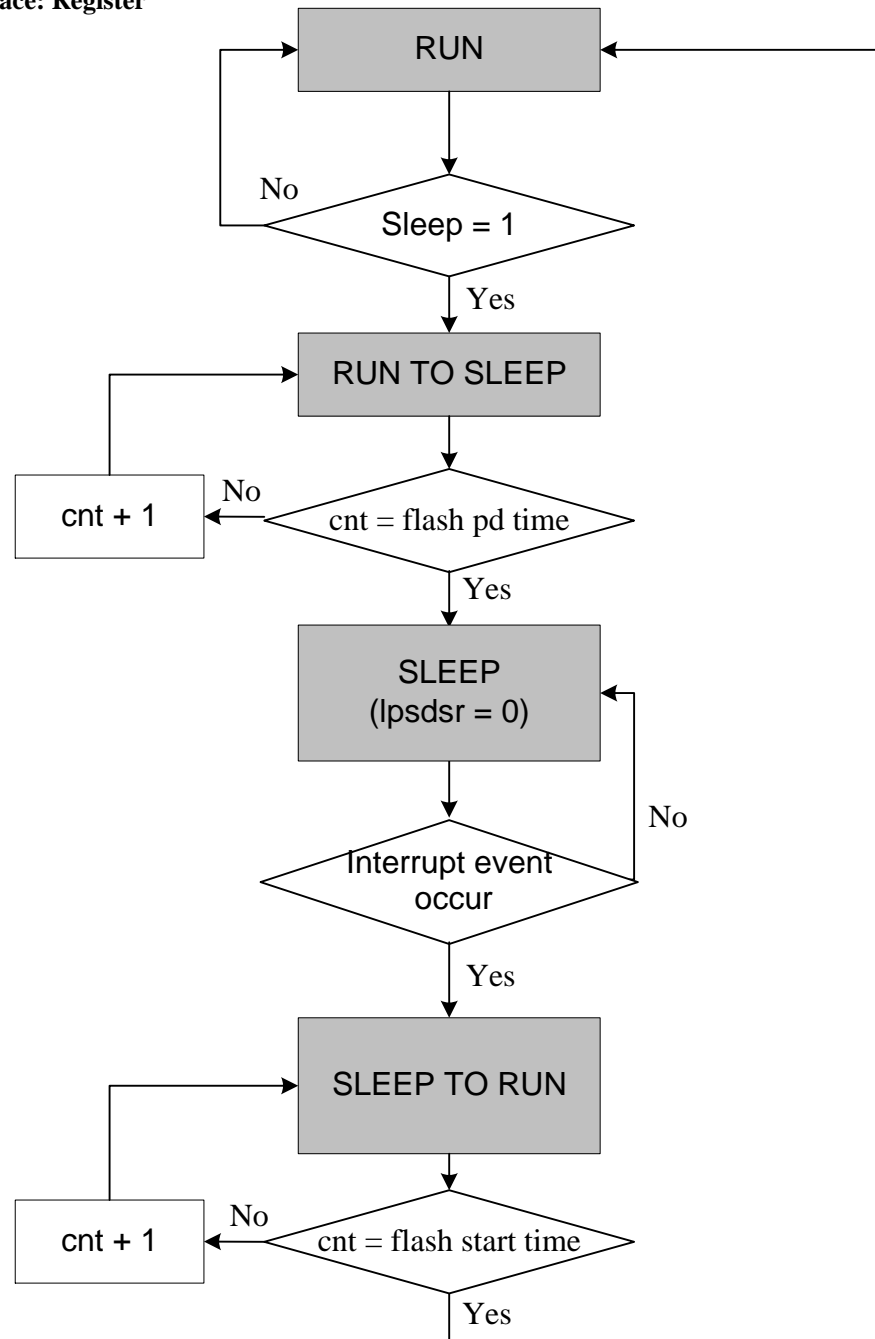


Figure 13 Sleep state machine

Sleep modes

The CPU clock is stopped in Sleep modes. The memory interface clocks (Flash memory and RAM interfaces) and all peripherals clocks can be stopped by **software** during Sleep. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.

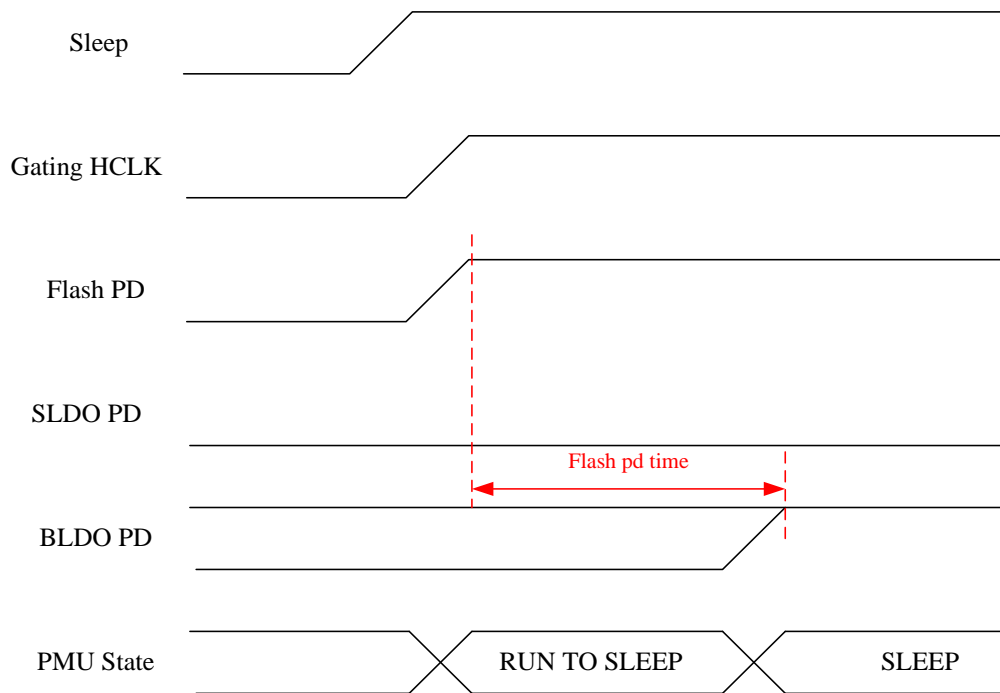


Figure 14 Timing Diagram of Entering Sleep Mode

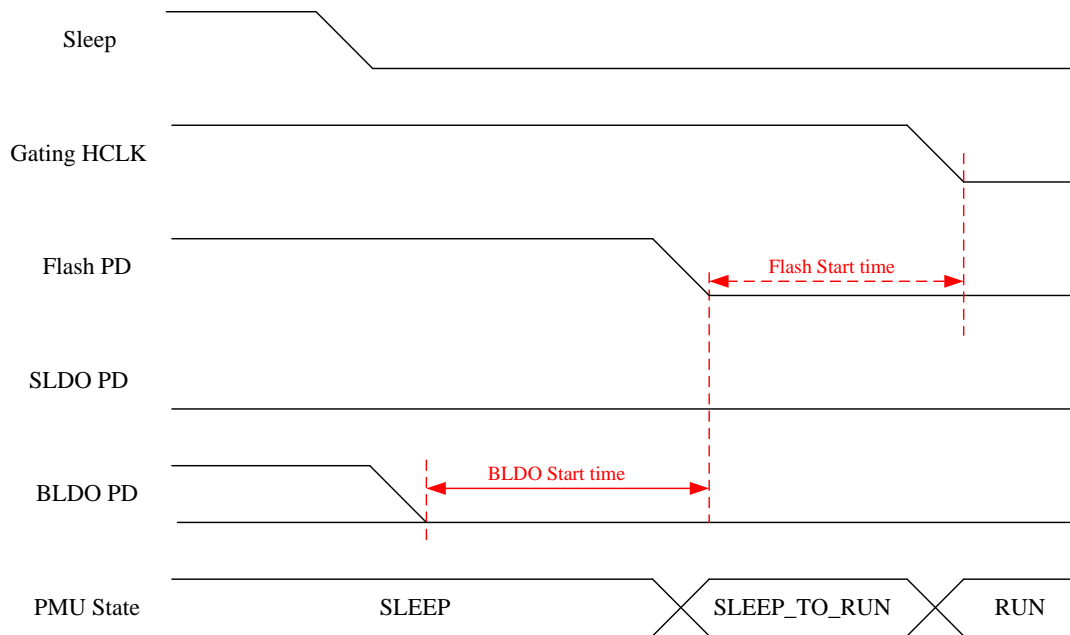


Figure 15 Timing Diagram of Exit Sleep mode

5.7.3 Stop mode

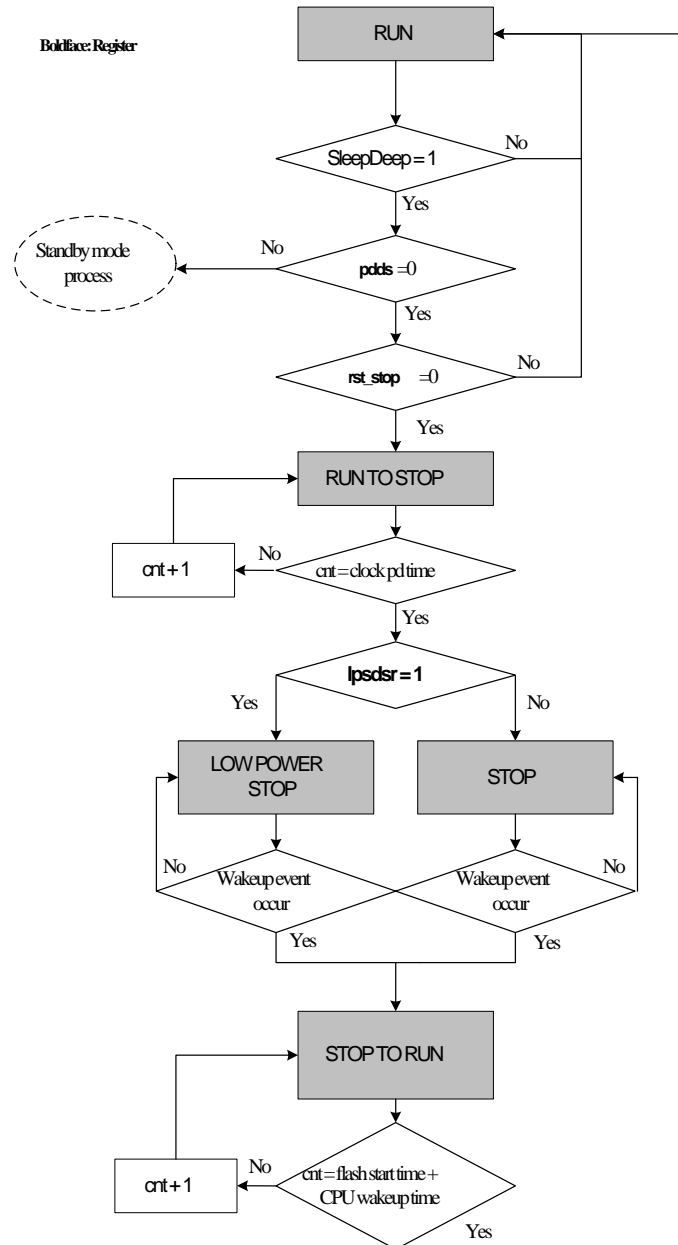


Figure 16 Stop Mode State Machine

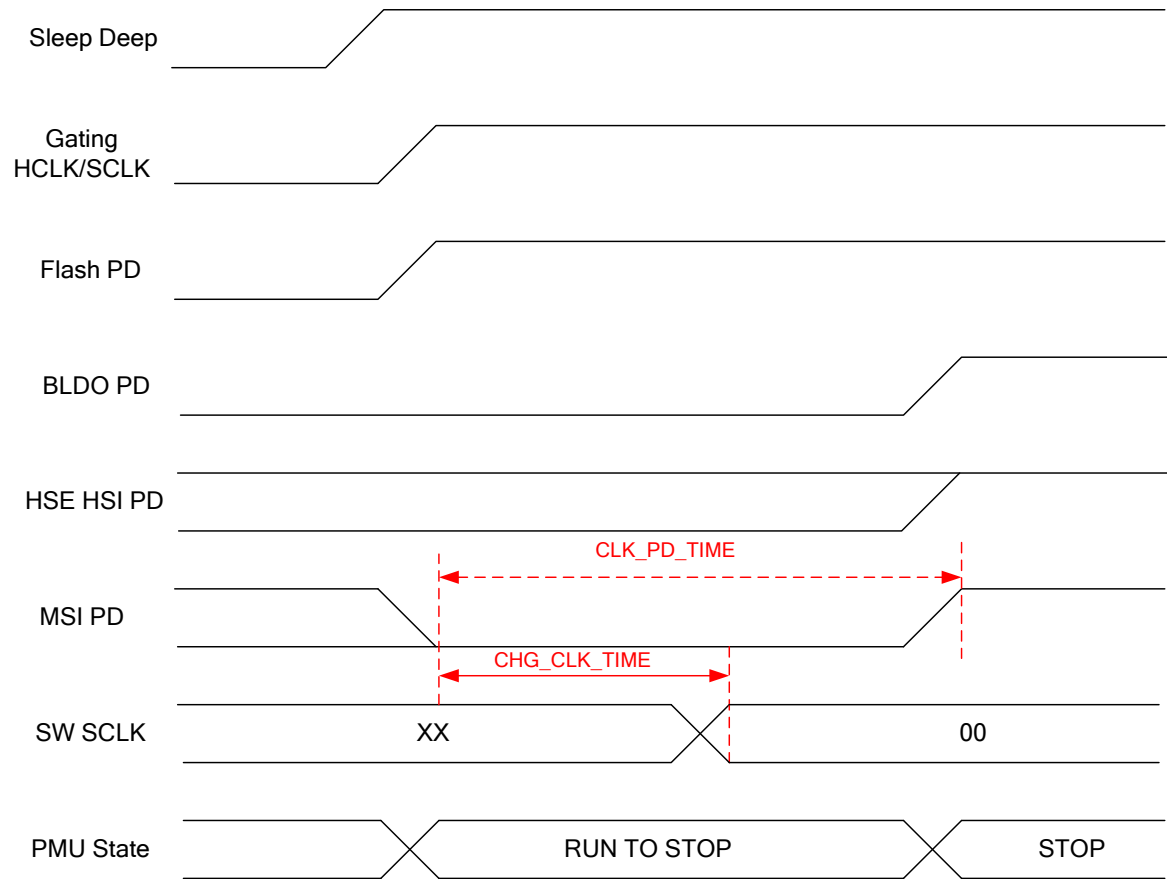


Figure 17 Timing Diagram of Entering Stop Mode

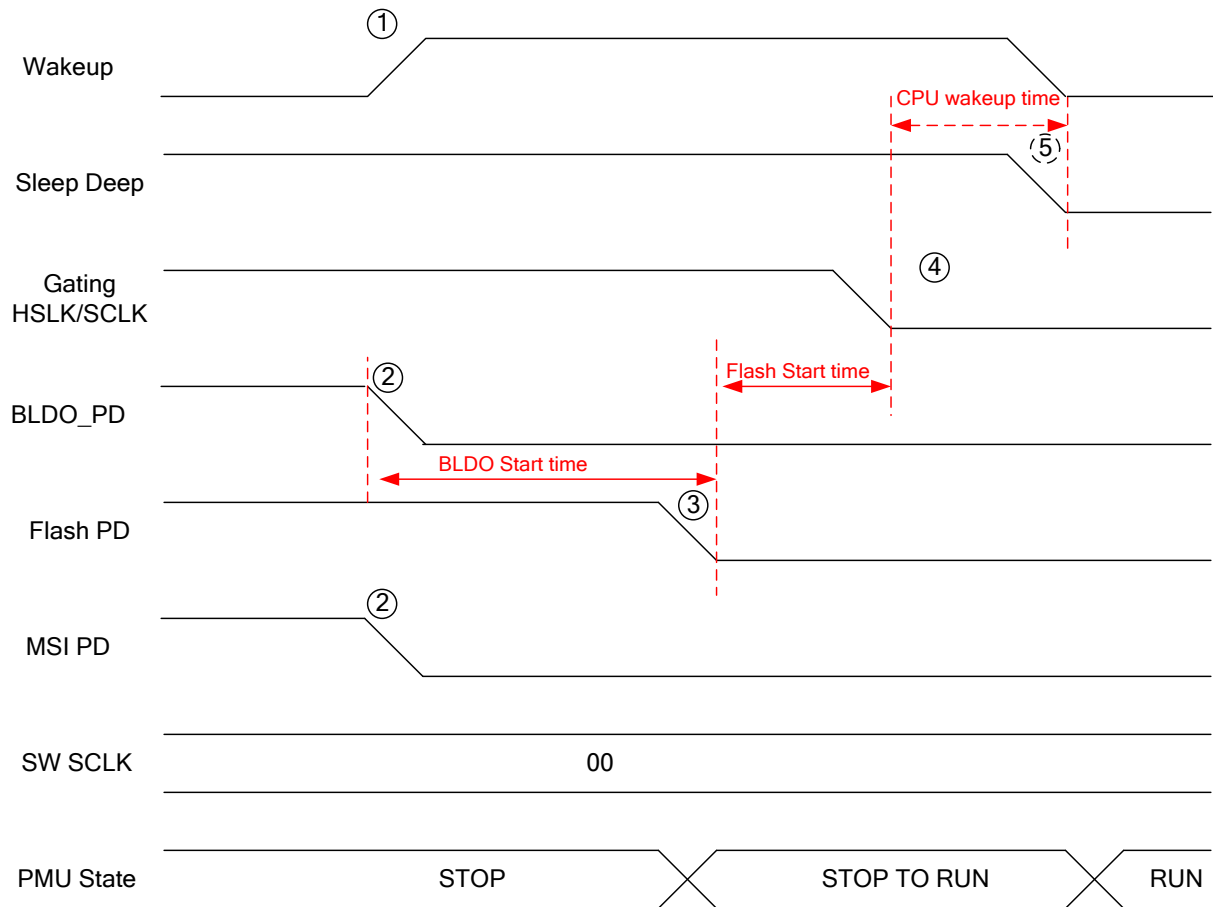


Figure 18 Timing Diagram of Exiting Stop Mode

5.7.4 Standby mode

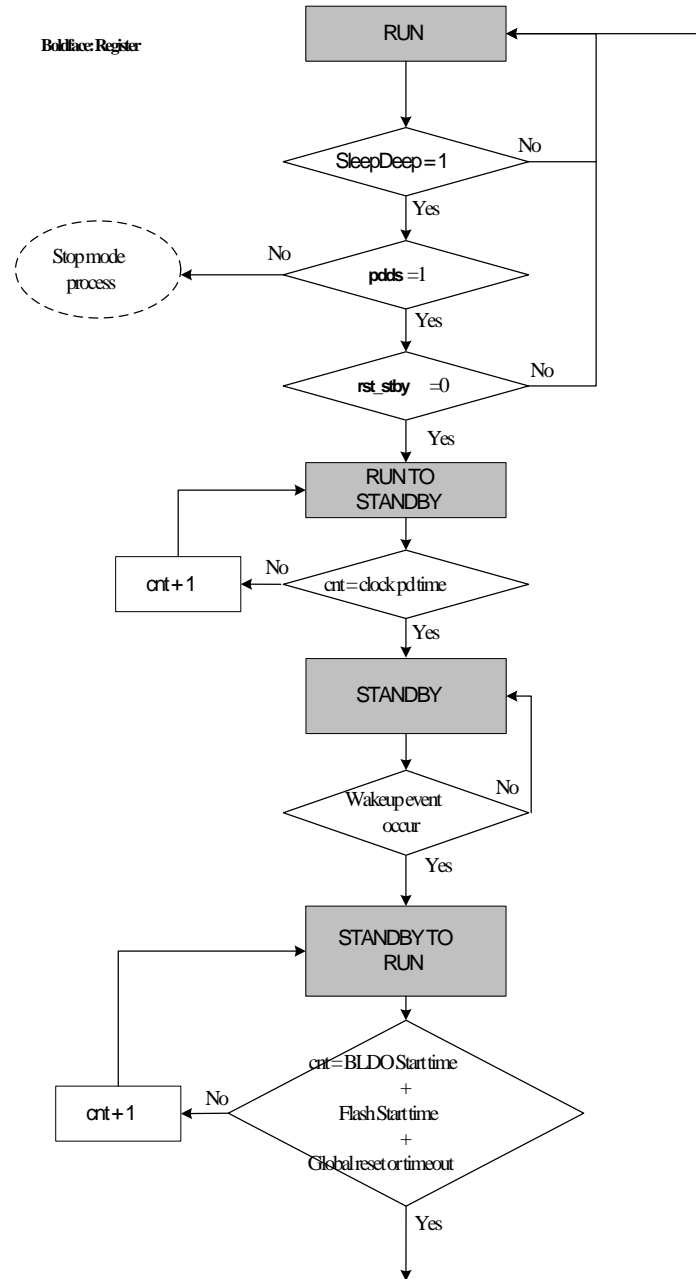


Figure 19 Standby Mode State Machine

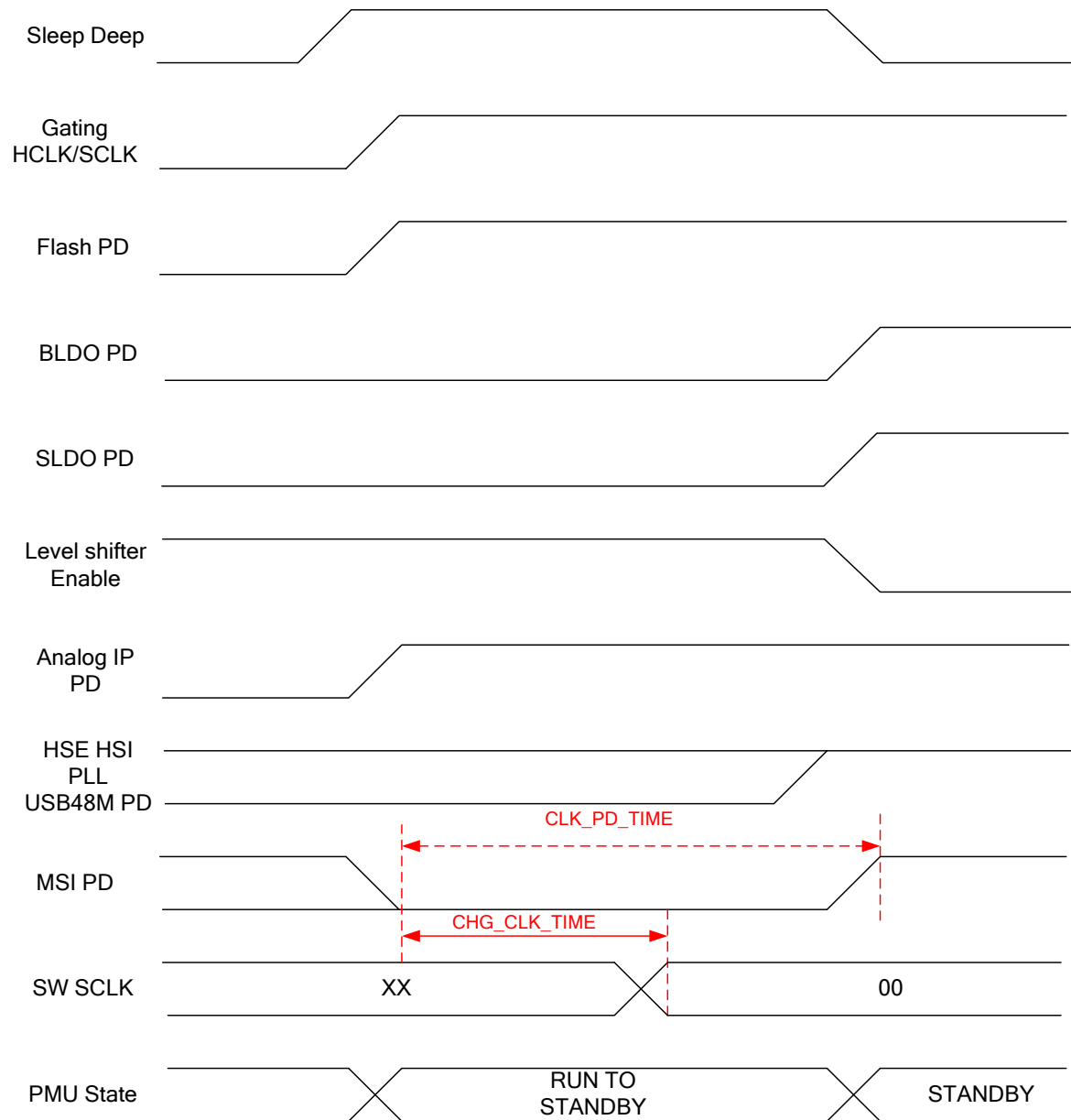


Figure 20 Timing Diagram of Enter Standby Mode

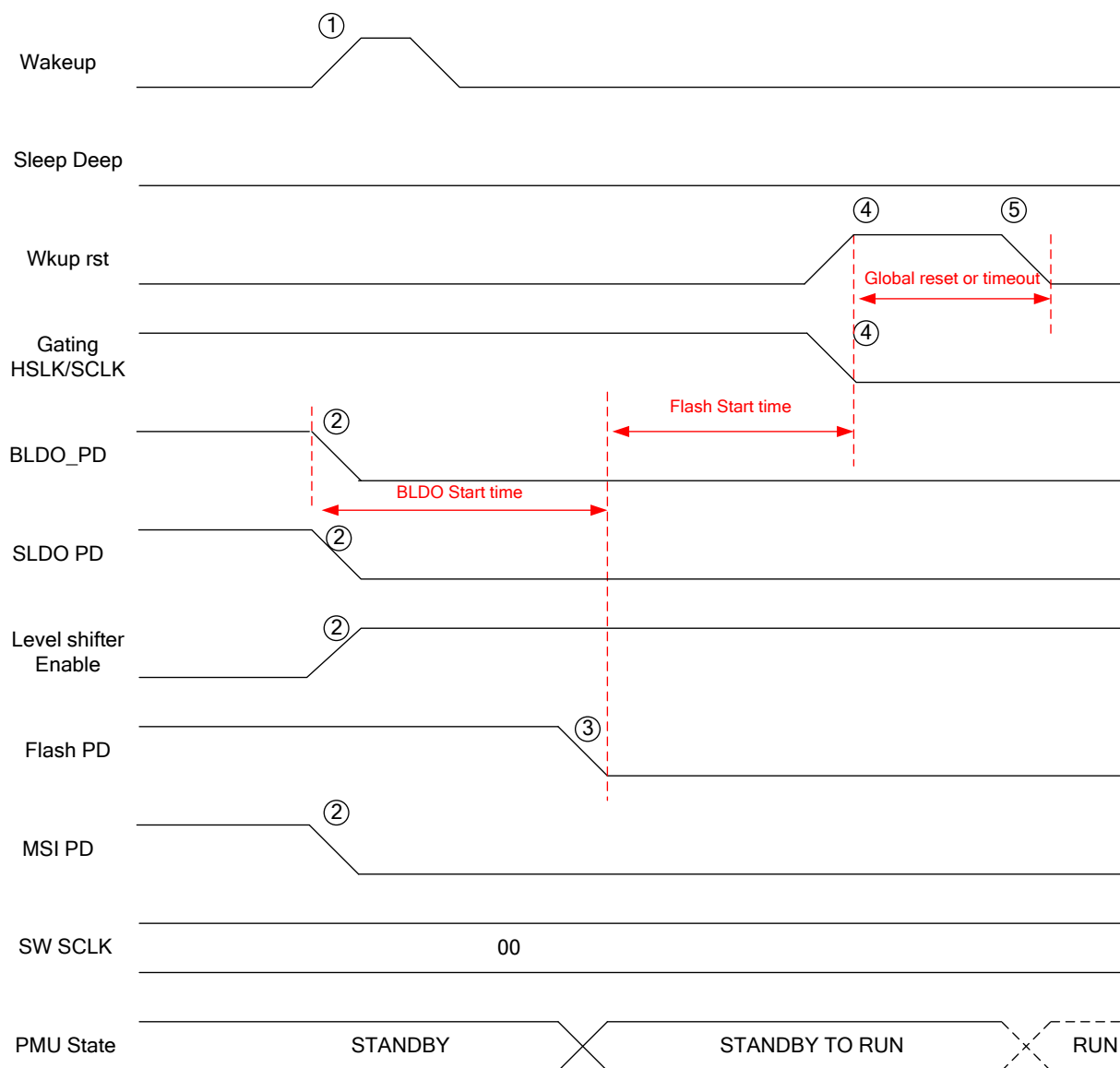


Figure 21 Timing Diagram of Exit Standby Mode

5.8 Setting Examples

Setting GPIO PA[0] as a wakeup pin and PA[3:1] are represent PMU status. User can measure current at any power mode (red square as the flowing diagram).

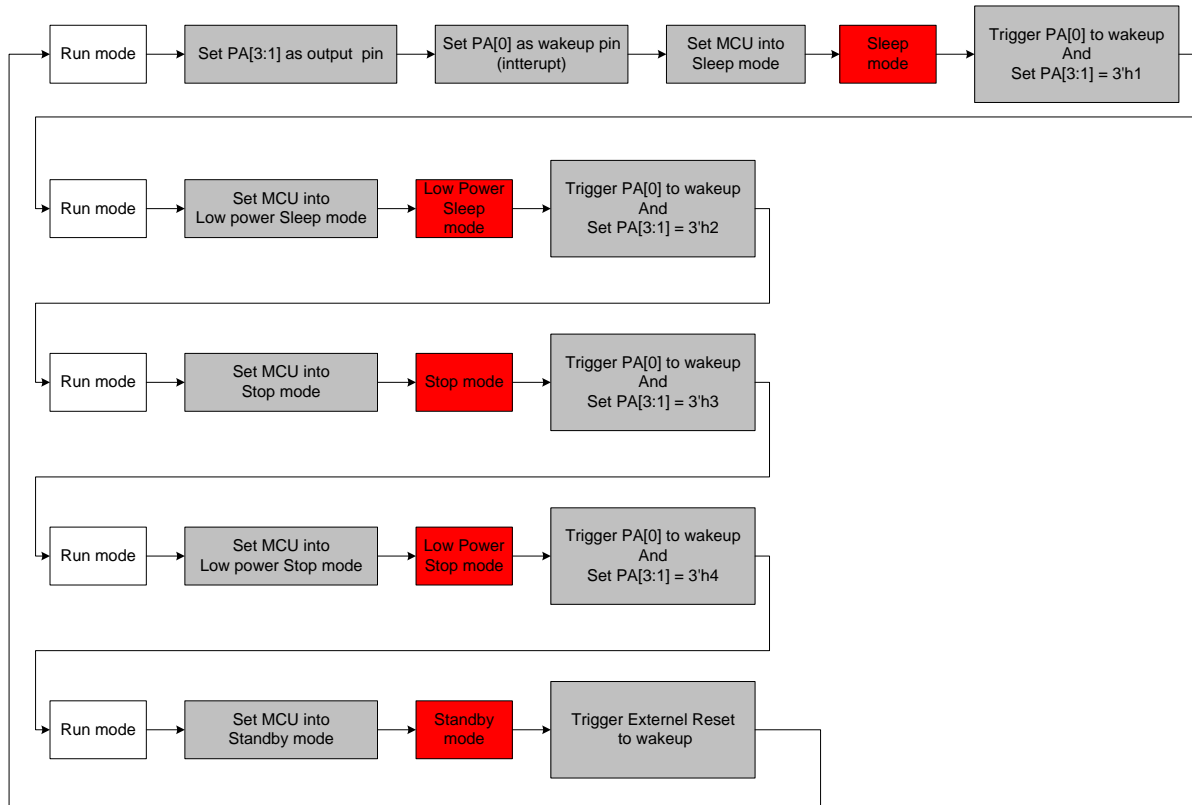


Figure 22 Power Down Diagram

5.9 Operating mode & Wake up Source

Operating Mode:

Operating Mode	BLDO	SLDO	LowFreq. OSC.	HighFreq. OSC	PLL	CPU	Peripherals	SRAM Data Retention
NORMAL	On	On	On	On	NA	On	NA	V
Low-Power RUN	Off	On	On	Off	Off	On	NA	V
SLEEP	On	On	On	On	Off	Off	NA	V
STOP	On	On	Off	On/Off(*1)	Off	Off	NA	V
Low-Power STOP	Off	On	Off	Off	Off	Off	Off	V
STANDBY	Off	Off	Off	Off	Off	Off	Off	X

*1: If BLDO on to let HSI on to keep peripherals work

*2: High Freq. OSC should be off by software, due to BLDO is off

Wake up sources:

Power Down Mode	2 lines Wake up I/O PA0 & PC13	GPIO Wake up	COMP Wake up	DAC Wake up	ADC Wake up	RTC Wake up	IWDT Wake up	USB Resume
SLEEP	V(INT)	V(INT)	V(INT)	V(INT)	V(INT)	V(INT)	V (RESET)	V V(INT)
Low Power STOP	V(INT)	V(INT)	V(INT)	NA	NA	V(INT)	V (RESET)	V V(INT)
STANDBY	V(high level trigger) (RESET)	NA	NA	NA	NA	V (RESET)	V (RESET)	

Function overview

Table 9 Peripherals on the Operating Mode

IPs	Run / Active	Sleep [wk up]	Low-power run	Low-power sleep[wk]	Stop		Low- power stop		Standby	
						Wakeup capability		Wakeup capability		Wakeup capability
USB	O	O	--	--	--	O	--	O	--	--
DAC	O	O	O	O	O	--	O	--	--	--
ADC	O	O	--	--	--	--	--	--	--	--
CMP	O	O	O	O	O	O	O	O	--	--
WWDT	O	O	O	O	--	--	--	--	--	--
IWDT	O	O	O	O	O	O	O	O	O	O
SysTick Timer	O	O	O	O	--	--	--	--	--	--
PVD	O	O	O	O	O	O	O	O	--	--
GPIOs	O	O	O	O	O	O	O	O	--	--
RTC	O	O	O	O	O	O	O	O	O	O
2 lines Wake up I/O PA0 & PC13 (high level trigger)	O	O	O	O	O	O	O	O	O	O
Wakeup time to Run mode	0μs	~7.2μs	0μs	~7.2μs	~42μs		~42μs		~610μs	
Consumption VDD=3.0 V (Typ.)	~8.9mA (32 MHz)	~330μA (MSI 4.2 MHz)	~202μA (MSI 1 MHz)	~265μA (MSI 4.2 MHz)	33μA (TYP.) (No RTC)		0.6μA (TYP.) (No RTC)		0.3 μA (TYP.) (No RTC)	
					33.7μA (TYP.) (with RTC)		1.3μA (TYP.) (with RTC)		0.9 μA (TYP.) (with RTC)	

"O" = Optional can be enabled/disabled by software).

"TBD" = not measured yet.

6 eFlash Control

6.1 Main Features

The NVM (embedded flash) is organized as 64-bit memory cells that can be used to store code, data, boot code or Option bytes. The memory array is divided into pages. A page is composed of 128x64-bit double words (or 1024 bytes) and a sector is composed of 4K bytes in Flash program memory and Option bytes areas (factory).

- Read interface organized by word, half-word or byte in every area
- Programming in the Flash memory performed by word
- Programming in the Option bytes area performed by word
- Erase operation performed by page (in Flash memory and Option bytes)
- Option byte Loader (init_load)
- Mass erase operation
- Read / Write protection
- PCROP (Proprietary Code Read-Out Protection) protection

6.2 Block Diagram

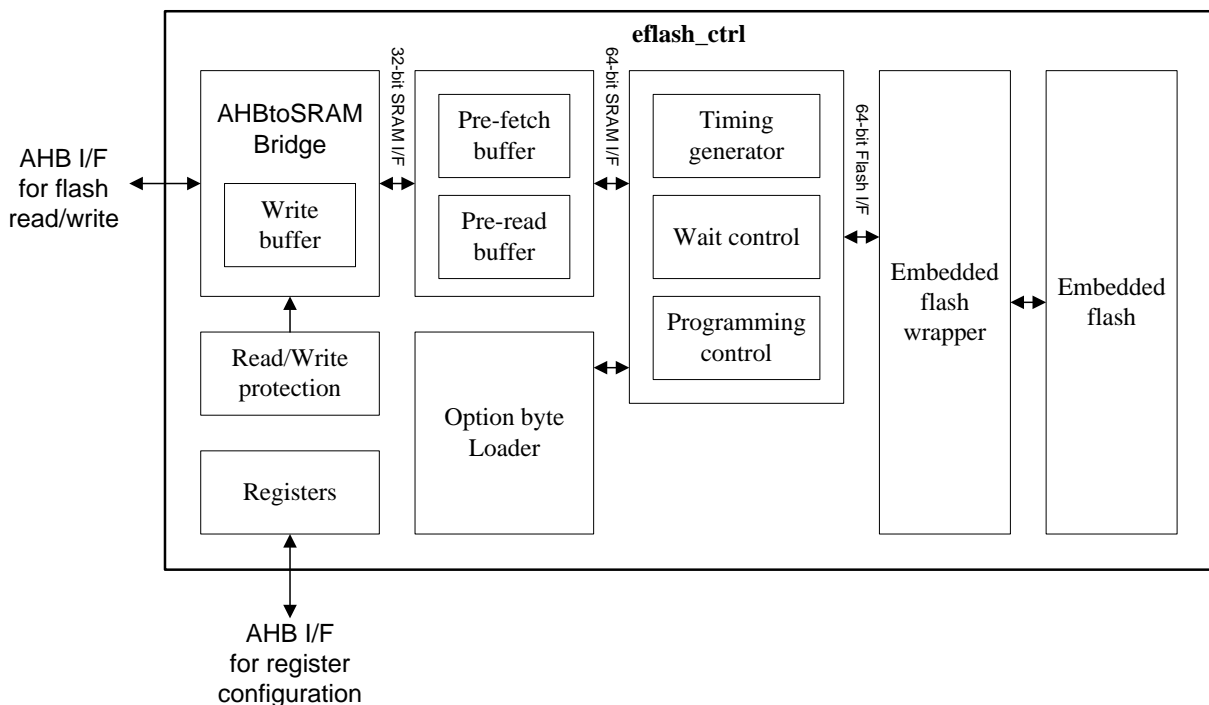


Figure 23 eFlash Control Block Diagram

6.3 Register Definition

Table 10 eFlash Control Register Table

Index	Bit	R/W	Default	Name	Description
FLASH_ACR: Flash access control					
00h	31:7	-	-	-	Reserved
	6	R/W	0	prften	This bit enables the prefetch. It is automatically reset every time the disab_buf bit (in this register) is set to 1. To know how the prefetch works, see the "Fetch and prefetch" section. 0: The prefetch is disabled. 1: The prefetch is enabled. The memory interface stores the last address fetched and tries to read the next one when no other read or write operation is ongoing.
	5	R/W	0	pre_read	This bit enables the pre-read. 0: The pre-read is disabled 1: The pre-read is enabled. The memory interface stores the last address read as data and tries to read the next one when no other read or write or prefetch operation is ongoing. Note: It is automatically reset every time the disab_buf bit (in this register) is set to 1.
	4	R/W	1	disab_buf	This bit disables the buffers used as a cache during a read. This means that every read will access the NVM even for an address already read (for example, the previous address). When this bit is reset, the prften and pre_read bits are automatically reset, too. 0: The buffers are enabled 1: The buffers are disabled. Every time one NVM value is necessary, one new memory read sequence has do be done.
	3:2	-	-	-	Reserved
	1:0	R/W	0	latency	The value of these bits specifies if a 0~3 wait-state is necessary to read the NVM. The user must write the correct value relative to the core frequency and the operation mode (power). The correct value to use can be found in Table 12 is done to verify if the configuration is correct. To increase the clock frequency, the user has to change these bits, then to increase the frequency. To reduce the clock frequency, the user has to decrease the frequency, then to change these bits . 0: 0 wait state is used to read a word in the NVM. 1: 1 wait state is used to read a word in the NVM. 2: 2 wait state is used to read a word in the NVM. 3: 3 wait state is used to read a word in the

Index	Bit	R/W	Default	Name	Description
					NVM.
FLASH_PECR: Flash program erase control					
04h	31:18	-	-	-	Reserved
	17	R/W	0	errie	Error interrupt enable 0: Error interrupt disable. 1: Error interrupt enable. Note: This bit can only be modified when pelock is 0. Locking pelock does not reset this bit; the interrupt remains enabled.
	16	R/W	0	eopie	End of programming interrupt enable 0: End of program interrupt disable. 1: End of program interrupt enable. Note: This bit can only be modified when pelock is 0. Locking pelock does not reset this bit; the interrupt remains enabled.
	15:12	-	-	-	Reserved
	11	-	-	-	Reserved
	10	R/W	0	mass	0: no mass erase operation requested 1: mass erase operation requested Note: This bit can be modified when pelock is 0. It is reset when pelock is set.
	9	-	-	-	Reserved
	8	R/W	0	erase	0: No erase operation requested. 1: Erase operation requested. Note: This bit can be modified when pelock is 0. It is reset when pelock is set.
	7:3	-	-	-	Reserved
	2	-	-	-	Reserved
	1	R/W	1	prglock	Program memory lock This bit blocks the write/erase operations to the Flash program memory. It can only be written to 1 to re-lock. To reset to 0, a correct sequence of unlock with flash_prgkeyr[31:0] register is necessary (see Unlocking the Flash program memory), with pelock bit at 0. The keys to unlock are: - First key: 8C9D_AEBFh - Second key: 1314_1516h 0: The write and erase operations in the Flash program memory are unlocked. 1: The write and erase operations in the Flash program memory are locked. Note: This bit is set when pelock is set.
	0	R/W	1	pelock	Flash_pecr register lock This bit locks the flash_pecr register. It can only be written to 1 to re-lock. To reset to 0, a correct sequence of unlock with pekeyr[31:0] register (see "Unlocking the flash_pecr register" section) is necessary. The keys to unlock are: - First key: 89AB_CDEF - Second key: 0203_0405 0: The flash_pecr register is unlocked; it can be

Index	Bit	R/W	Default	Name	Description
					modified and the other bits unlocked. Data write/erase operations are enabled. 1: The flash_pecr register is locked and no write/erase operation can start.
Reserved					
08h	31:0	-	-	-	Reserved
FLASH_PEKEYR: Flash program erase key					
0ch	31:0	W	-	flash_pekeyr	This is a write-only register. With a sequence of two write operations (the first one with 89AB_CDEFh and the second one with 0203_0405h), the write size being that of a word, it is possible to unlock the flash_pecr[31:0] register. For more details, refer to “Unlocking and the FLASH_PECR register” section.
FLASH_PRGKEYR: Flash program memory key					
10h	31:0	W	-	flash_prgkeyr	This is a write-only register. With a sequence of two write operations (the first one with 8C9D_AEBFh and the second one with 1314_1516h), the write size being that of a word, it is possible to unlock the Flash program memory. The sequence can only be executed when pelock is already unlocked. For more details, refer to “Unlocking the Flash program memory” section.
FLASH_SR: Flash status					
18h	31:14	-	-	-	Reserved
	13	R/W1c	0	rderr	This bit is set by hardware when the user tries to read an area protected by PcROP. It is cleared by writing 1. 0: No read protection error happened. 1: One read protection error happened.
	12	-	-	-	Reserved
	11	R/W1c	0	optverr	Option valid error This bit is set by hardware when, during an Option byte loading, there was a mismatch for one or more configurations. It means that the configurations loaded may be different from what the user wrote in the memory. It is cleared by writing 1. If an error happens while loading the protections (wprmod, rdprot[7:0], wrprot[x]), the source code in the Flash program memory may not execute correctly. 0: No error happened during the Option bytes loading. 1: One or more errors happened during the Option bytes loading
	10:9	-	-	-	Reserved
	8	R/W1c	0	wrperr	Write protection error This bit is set by hardware when an address to be programmed or erased is write-protected. It

Index	Bit	R/W	Default	Name	Description
					is cleared by writing 1. 0: No protection error happened. 1: One protection error happened.
	7:4	-	-	-	Reserved
	4	R	0	lve	Low voltage read 0: NVM in normal read mode 1: NVM in low voltage read mode
	3	-	-	-	Reserved
	2	R	1	endhvh	This bit is set and reset by hardware. 0: High voltage is executing a write/erase operation in the NVM. 1: High voltage is off, no write/erase operation is ongoing.
	1	R/W1c	0	eop	End of program (EOP) This bit is set by hardware at the end of a write or erase operation when the operation has not been aborted. It is reset by software (writing 1). 0: No EOP operation occurred 1: An EOP event occurred. An interrupt is generated if eopie bit is set.
	0	R	0	bsy	Memory interface busy Write/erase operations are in progress. 0: No write/erase operation is in progress. 1: A write/erase operation is in progress.
Reserved					
1ch	31:0			-	Reserved
FLASH_INIT0: Analog initial load					
20h	31:0			-	Reserved
FLASH_INIT1: Analog initial load					
24h	31:0			-	Reserved
FLASH_OPTR: Boot ROM option					
28h	31:30	R/W	-	nboot	This bit is used to select the device boot mode. If there is a mismatch on this configuration during the option bytes loading, it is loaded with 0. 0: remap 0x00000000 to ROM 1: remap 0x00000000 to flash 2: remap 0x00000000 to system ram After standby reset, this bit will be asserted if it is matched during Option bytes loading.
	29:9	-	-	-	Reserved
	8	R	-	wprmod	This bit selects between write and read protection of Flash program memory sectors. If there is a mismatch on this configuration during the option bytes loading, it is loaded with 1. PCROP=0. The wrprot[x] bits are used as a write protection on a sector. PCROP=1. The wrprot[x] bits are used as a read protection on a sector. (also write protection)
	7:0	R/W	-	rdprot	Read protection

Index	Bit	R/W	Default	Name	Description
					These bits contain the protection level loaded during the Option byte loading. If there is a mismatch on this configuration during the option bytes loading, it is loaded with 0x00. AAh: Level 0 Others: Level 1
FLASH_WRPROT0: Write protection					
2ch	31:0	R/W	0	flash_wrprot[31:0]	<p>Write protection for sector 0~31 If wprmod = 0 in the FLASH_OPTR register, these bits contain the write protection configuration for the flash memory (every bit protects a 4-Kbyte sector: the first bit protects the first sector, the second bit protects the second sector and so on). In this case, 1 = page protected, 0 = no protection. It is possible to set or reset these bits without any limitation changing the relative option bytes.</p> <p>If wprmod = 1, these bits are used to protect from reading as data (see “Read as data and pre-read” section), and then also from writing, with the same granularity and with the same combination of bits and sectors. The read protection does not protect against a fetch. In this case, 1 = no protection, 0 = sector protected. It is only possible to increase the protection, which means that the user can add zeros but cannot add ones.</p> <p>The mass erase deletes the wprmod bit but does not delete the content of this register. After a mass erase, the user must write the relative option bytes with zeros to remove completely the write protections.</p> <p>If there is a mismatch on this configuration during the option bytes loading, and the content of wprmod in the FLASH_OPTR register is: 1, this configuration is loaded with 0000h. 0, this configuration is loaded with FFFFh. If there was a mismatch when wprmod was loaded in the FLASH_OPTR register (thus loaded with ones), the register is loaded with 0000h.</p>
FLASH_WRPROT1: Write protection					
30h	31:0	R/W	0	flash_wrprot[63:32]	Write protection for sector 32~63
Reserved					
40h	31:0				reserve
Reserved					
44h	31:0				reserve
Reserved					
48h	31:0				reserve

Index	Bit	R/W	Default	Name	Description
Reserved					
4ch	31:0				reserve
Reserved					
50h	31:0				reserve

R/W1C: read & write one clear

6.4 Functional Description

6.4.1 NVM Functional Description

The NVM is organized as 64-bit memory cells that can be used to store code, data, boot code or Option bytes. The memory array is divided into pages. A page is composed of 1024 bytes (128x64-bit) and a sector is composed of 4K bytes in Flash program memory and Option bytes areas (user and factory). Table 11 shows the NVM organization (including 64 pages in main memory block)

Table 11 NVM organization

NVM	NVM address	Page	Sector	Description
Main memory block	1000_0000h - 1000_03FFh	0	0	Flash program memory
	1000_0400h - 1000_07FFh	1		
	1000_0800h - 1000_0BFFh	2		
	1000_0C00h - 1000_0FFFh	3		
		
	1000_F000h - 1000_F3FFh	60	15	
	1000_F400h - 1000_F7FFh	61		
	1000_F800h - 1000_FBFFh	62		
	1000_FC00h - 1000_FFFFh	63		

6.4.2 Reading the NVM

➤ Protocol to read

Depending on the clock frequency, a 0~1 wait-state can be necessary to read the NVM. The user must set the correct number of wait states (latency[1:0] bits in the FLASH_ACR register). No control is done to verify if the frequency used is correct, with respect to the number of wait states. A wrong number of wait states can generate wrong read values (high frequency and 0 wait states) or a long time to execute a code (low frequency with 3 wait-state).

You can read the NVM by word (4 bytes), half-word (2 bytes) or byte. It is not possible to read during a write/erase operation. If a write/erase operation is ongoing, the reading will be in a wait state until the write/erase operation completes, stalling the master that requested the read operation, except when the address is read-protected. In this case, the error is sent to the master by a hard fault or a memory interface flag; no stall is generated and no read is waiting. Table12 shows the delays to read a word in the NVM.

Table 12 Number of wait states

Frequency range	Wait state required
≤ 16MHz	0
> 16MHz, ≤ 32MHz	1

➤ Change the CPU Frequency

After reset, the clock used is the MSI (2.1 MHz) and 0 wait-state is configured in the FLASH_ACR register. Please follow below software sequences to tune the number of wait states needed to access the

NVM with the CPU frequency. A CPU clock or a number of wait-state configuration changes may take some time before being effective. Checking the AHB pre-scaler factor and the clock source status values is a way to ensure that the correct CPU clock frequency is the configured one. Similarly, the read of FLASH_ACR register is a way to ensure that the number of programmed wait-states is effective.

■ Increasing the CPU frequency

1. Program 1 wait state in latency[1:0] bits of FLASH_ACR register, if necessary.
2. Check that the new number of wait-states is taken into account by reading the FLASH_ACR register. When the number of wait-states changes, the memory interface modifies the way the read access is done to the NVM. The number of wait states cannot be modified when a read operation is ongoing, so the memory interface waits until no read is done on the NVM. If the master reads back the content of the FLASH_ACR register, this reading is stopped (and also the master which requested the reading) until the number of wait-states is really changed. If the user does not read back the register, the following access to the NVM may be done with 0 wait-states, even if the clock frequency has been increased, and consequently the values are wrong.
3. Modify the CPU clock source and/or the AHB clock pre-scaler in the Reset & Clock Controller (RCC).
4. Check that the new CPU clock source and/or the new CPU clock pre-scaler value is taken into account by reading respectively the clock source status and/or the AHB pre-scaler value in the Reset & Clock Controller (RCC). This check is important as some clocks may take time to get available.

■ Decreasing the CPU frequency

1. Modify the CPU clock source and/or the AHB clock pre-scaler in the Reset & Clock Controller (RCC).
2. Check that the new CPU clock source and/or the new CPU clock pre-scaler value is taken into account by reading respectively the clock source status and/or the AHB pre-scaler value in the Reset and Clock Controller (RCC).
3. Program 0 wait state in latency[1:0] bits of the FLASH_ACR register, if needed.
4. Check that the new number of wait states is taken into account by reading FLASH_ACR register. It is necessary to read back the register for the reasons explained in the previous paragraph.

➤ Data buffering

In the NVM, six buffers can impact the performance (and in some conditions help to reduce the power consumption) during read operations, both for fetch and data. The structure of one buffer is shown on Figure 24. Each buffer stores 3 different types of information: address, data and history. In a read operation, if the address is found, the memory interface can return data without accessing the NVM. Data in the buffer is 64 bit wide (even if the master only reads 8 or 16 bits), so that a value can be returned whatever the size used in a previous reading. The history is used to know if the content of a buffer is valid and to delete (with a new value) the older one.



Figure 24 Structure of one internal buffer

The buffers are used to store the value received by the NVM and for speculative readings during normal read operations. Disabling the speculative reading makes that only the data requested by masters is stored in buffers, if enabled (default). This can increase the performance as no wait-state is necessary if the value is already available in buffers, and reduce the power consumption as the number of reads in memory is reduced and all combinatorial paths from memory are stable.

The buffers are divided in groups to manage different tasks. The number of buffers in every group can change starting from the configuration selected by the user (see Table 13). The total number of buffers used is always 6 (if enabled). The history is always managed by group. The memory interface always searches if a particular address is available in all buffers without checking the group of buffers and if the read is fetch or data.

At reset or after a write/erase operation that changes several addresses, all buffers are empty and the history is set to EMPTY. After a program by word, half-word or byte, only the buffer with the concerned address is cleaned.

Table 13 Pre-fetch & Pre-buffer Management

disab_buf	prften	pre_read	Buffers for fetch			Buffers for data	
			Buffers for jumps	Buffers for prefetch	Buffers for last value	Buffers for pre-read	Buffers for last value
1	-	-	0	0	0	0	0
0	0	0	3	0	1	0	2
0	1	0	2	1	1	0	2
0	0	1	3	0	1	1	1
0	1	1	2	1	1	1	1

If a value in a buffer is not empty, the history will show the time taken to read or write the value. The history is organized as a list of values from the latest to the oldest one. At a given instant, only one buffer in a group can have a particular value of history (except the empty value). Moving a buffer to the latest position, all other buffers in the group move one step further, thus maintaining the order. The history is changed to the latest position when the buffer is read (the master requests for the buffer content) or written (with a new value from the NVM). The memory interface always writes the oldest buffer (or one empty buffer, if any) of the right group when a new address is required in memory.

Three configuration bits of the FLASH_ACR register are used to manage the buffering:

■ **disab_buf**

Setting this bit disables all buffers. When this bit is 1, the prefetch or the pre-read operations cannot be enabled and if, for example, the master requests the same address twice, two readings are generated in the NVM.

■ **prften**

Setting this bit to 1 (with **disab_buf** to 0) enables the prefetch. When the memory interface does not have any operation in progress, the address following the last address fetched is read and stored in a buffer.

■ **pre_read**

Setting this bit to 1 (with **disab_buf** to 0) enables the pre-read. When the memory interface does not have any operation in progress or prefetch to execute, the address following the last data address is read and stored in a buffer.

➤ **Fetch and prefetch**

A memory interface fetch is a read from the NVM to execute the operation that has been read. The memory interface does not check the master who performs the read operation, or the location it reads from, but it only verifies if the read operation is done to execute what has been read. It means that a fetch can be performed:

■ **in all areas**

■ **with any size (16 or 32 bits)**

The memory interface stores in the buffers:

- The address of jumps so that, in a loop, it is only necessary to access the NVM the first time, because then the jump address is already available.
- The last read address so that, when performing a fetching on 16 bits, the other 16 bits are already available.

To manage the fetch, the memory interface uses 4 buffers: at reset (**disab_buf** = 0, **prften** = 0, **pre_read** = 0). 3 buffers are used to manage the jumps and 1 buffer to store the last value fetched. With this configuration, the 4 buffers for fetch are organized in 2 groups with separate histories: the group for loops and the group for the last value fetched.

Setting the **prften** bit to 1 enables the prefetch. The prefetch is a speculative read in the NVM, which is executed when no read is requested by masters, and where the memory interface reads from the last address fetched increased by 4 (one word). This read is with a lower priority and it is aborted if a master requests a read (data or fetch) to a different address than the prefetch one. When the prefetch is enabled, one buffer for loops is moved to a new group (of only one buffer) to store the prefetched value: 2 buffers continue to store the jumps, 1 buffer is used for prefetch and 1 buffer is used for the last value.

The memory interface can only prefetch one address, so the function is temporarily disabled when no fetch is done and the prefetch is already completed. After a prefetch, if the master requests the prefetched value, the content of the prefetch buffer is copied to the last value buffer and a new prefetch is enabled. If, instead, the master requests a different address, the content of the prefetch buffer is lost, a read in the NVM is started (if necessary) and, when it is complete, a new prefetch is enabled at the new address fetched increased by 4.

The prefetch can only increase the performance when reading with 1 wait state and for mostly linear codes: the user must evaluate the pros and cons to enable or not the prefetch in every situation. The prefetch increases the consumption because many more readings are done in the NVM (and not all of them will be used by the master).

➤ Read as data and pre-read

A data read from the memory interface, corresponds to any read operation that is not a fetch. The master reads operation constants and parameters as data. At reset, (disab_buf = 0, prften = 0, pre_read = 0), the memory interface uses 2 buffers organized in one group to store the last two values read as data.

In some particular cases, it can be useful to enable the pre-read (pre_read = 1 with disab_buf = 0). The pre-read works exactly like the prefetch: it is a speculative reading at the last data address increased by 4 (one word). With this configuration, one buffer of data is moved to a new group to store the pre-read value, while the second buffer continues to store the last value read. For a prefetch, the pre-read value is copied in the last read value if the master requests it, or is lost if the master requests a different address.

The pre-read has a lower priority than a normal read or a prefetch operation: this means that it will be launched only when no other type of read is on-going. Pay attention to the fact that a pre-read used in a wrong situation can be harmful: in a code where a data read is not done linearly, reducing the number of buffers (from 2 to 1) used for the last read value can increase the number of accesses to the NVM (and the time to read the value). Moreover, this can generate a delay on prefetch. Table 14 is a summary of the possible configurations.

Table 14 Configurations for buffers and speculative reading

disab_buf	prften	pre_read	Description
1	X	X	Buffers disabled
0	0	0	Buffer enabled: no speculative reading is done
0	1	0	Prefetch enabled: speculative reading on fetch enabled
0	0	1	pre-read enabled: speculative reading on data enabled
0	1	1	Prefetch and pre-read enabled: speculative reading on fetch and data enabled

6.4.3 Writing/Erasing the NVM

There are many ways to modify the contents of NVM. The memory interface helps to reduce the possibility of unwanted modification to the NVM contents, and to implement all procedures required to erase or write in different memory areas by hardware.

➤ **Write/erase protocol**

To write/erase memory contents during protections removing, please do the following:

1. Configure the operation to execute.
2. Send the right number of data to the memory interface, writing one or several addresses in the NVM.
3. Wait for the operation to complete.

During the waiting time, the user can prepare for the next operation (except in very particular cases) writing the new configuration and starting to write data for the next write/erase operation.

➤ **Single programming operation**

With this protocol, the software has to write a value in a not-protected address of the NVM. When the memory interface receives this writing request, it stops the master for some pulses of clock while it checks the protections and the previous value and it latches the new value inside the NVM. The software can then start to configure the next operation. The operation will complete when the eop bit of FLASH_SR register rises (if it was 0 at the beginning of operation).

6.4.4 Unlocking/Locking Operations

It is necessary to enable the unlocking function before performing a write/erase operation. The user can write into the Flash memory register, program memory, and Option bytes areas. **To perform a write/erase operation, please unlock the pelock bit of the FLASH_PECR register.** When this bit is unlocked (its value is 0), the other bits of the same register can be modified.

To write/erase the Flash program memory, unlock prglock[0] bit of the FLASH_PECR register. The bit can only be unlocked when pelock is 0.

➤ Unlocking the FLASH_PECR register

After a reset, the flash_pecr register are not accessible in write mode because pelock bit in the FLASH_PECR register is set. The same unlocking sequence unprotects both of them at the same time. The following sequence is used to unlock the FLASH_PECR register:

1. **Write PEKEY1 = 0x89ABCDEF to the FLASH_PEKEYR register**
2. **Write PEKEY2 = 0x02030405 to the FLASH_PEKEYR register**

Any wrong key sequence will generate a hard fault. Even if the master tries to write another register between the two key sequences or use the wrong key. A reading access does not generate an error and does not interrupt the sequence. A hard fault is returned in any of the four cases below:

- After the first write access if the PEKEY1 value entered is erroneous.
- During the second write access if PEKEY1 is correctly entered but the value of PEKEY2 does not match.
- If there is any attempt to write a third value to flash_pekeyr (pay attention: this is also true for the debugger).
- If there is any attempt to write a different register of the memory interface between PEKEY1 and PEKEY2.

When properly executed, the unlocking sequence clears pelock bit in the FLASH_PECR register.

To lock flash_pecr again, the software only needs to set pelock bit in flash_pecr. When locked again, pelock bit needs a new sequence to return to 0.

➤ Unlocking the Flash program memory

An additional protection is implemented to write/erase the Flash program memory. After a reset, the Flash program memory is no more accessible in write mode: prglock[0] bit is set in the FLASH_PECR register. A write access to the Flash program memory is granted by clearing prglock[0] bit. The following sequence is used to unlock the Flash program memory:

1. Unlock the FLASH_PECR register (see the FLASH_PECR register" section).
2. **Write PRGKEY1 = 0x8C9DAEBF to the FLASH_PRGKEYR register.**
3. **Write PRGKEY2 = 0x13141516 to the FLASH_PRGKEYR register.**

If the keys are written with pelock set to 1, no error is generated and prglock[0] remains at 1. It will be unlocked while re-executing the process with pelock = 0. It will return to hard fault in any of the four cases below:

- After the first write access if the entered PRGKEY1 value is erroneous.
- During the second write access if PRGKEY1 is correctly entered but the PRGKEY2 value does not match.
- If there is any attempt to write a third value to flash_prgkeyr (this is also true for the debugger).

- If there is any attempt to write a different register of the memory interface between PRGKEY1 and PRGKEY2.

When properly executed, the unlocking sequence clears the prglock bit and the Flash program memory is write-accessible

To lock the Flash program memory again, the software only needs to set prglock[0] bit in FLASH_PECR register. When locked again, prglock[0] bit needs a new sequence to return to 0. If pelock returns to 1 (locked), prglock[0] is automatically locked, too.

6.4.5 Status Register

The FLASH_SR Status Register gives some information on the memory interface or the NVM status (operation(s) ongoing) and about errors that happened.

➤ **bsy**

This flag is set and reset by hardware. It is set to 1 every time the memory interface executes a write/erase operation, and it informs that no other operation can be executed. If a new operation is requested, different conditions can occur:

- Waiting for read, or waiting for write/erase, or waiting for option loading.
If the software requests a write operation while a write/erase operation is executing (hvoff = 0), the memory interface stalls the master and has the pending operation execute as soon as the write/erase operation is complete.
- rderr error:
If the software requests a read operation while a write/erase operation is executing (hvoff = 0) but the address is protected, the memory interface rises the flag and continues to wait for the end of the write/erase operation.

➤ **eop**

This flag is set by hardware and reset by software. The software can reset it writing 1 in the status register. This bit is set when the write/erase operation is completed and the memory interface can work on other operations (or start to work on pending operations).

It is useful to clear it before starting a new write/erase operation, in order to know when the actual operation is complete. It is very important to wait for this flag to rise when a mass erase is on-going, before requesting a new operation.

➤ **hvoff**

This flag is set and reset by hardware and it is a memory interface information coming from the NVM: it informs when the High-Voltage Regulators are on (= 0) or off (= 1).

6.5 Memory Protection

The user can protect part of the NVM (Flash program memory and Option bytes areas) from unwanted write and against code hacking (unwanted read). **Three types of protections are implemented.**

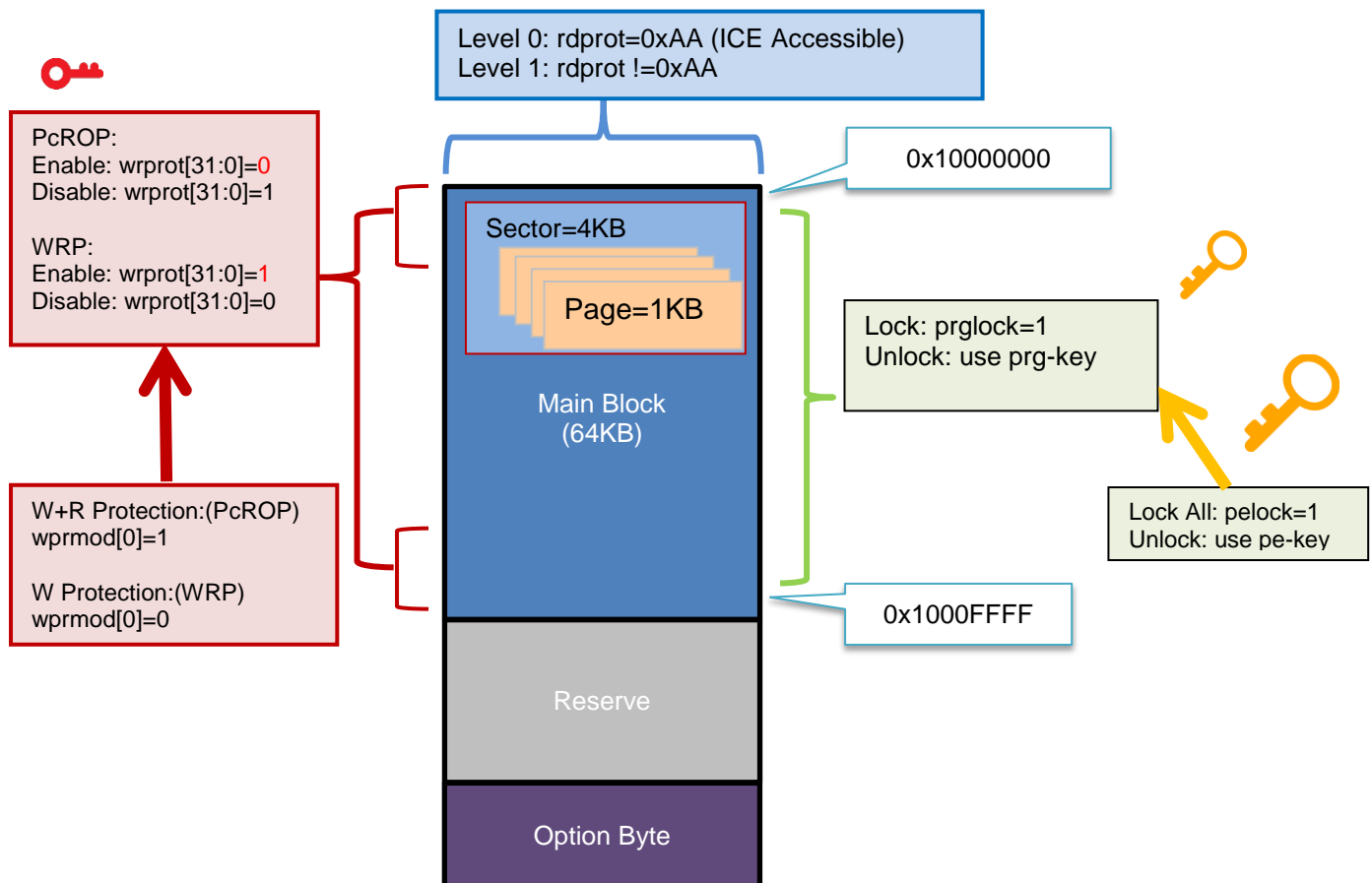


Figure 25 Memory Protection

6.5.1 RDP (Read Out Protection)

This type of protection aims at protecting against unwanted read (hacking) of the NVM content. This protection is managed by rdprot[7:0] bit field in the FLASH_OPTR register. The value is loaded from the Option bytes area during a boot and copied in the read-only register.

Two protection levels are defined:

- Level 0: no protection

Level 0 is set when rdprot[7:0] is set to 0xAA. When this level is enabled, and if no other protection is enabled, read and write can be done in the Flash program memory and Option bytes areas without restrictions.

➤ Level 1: memory read protection

Level 1 is set when rdprot[7:0] is set to any value except 0xAA, respectively used for Level 0. This is the default protection level after an Option bytes erase or when there is a mismatch in the rdprot[7:0] field.

It protects the Flash program memory. When protection Level 1 is set through debugger-is required to execute the user code.

When this level is enabled:

- No access to the Flash program memory (read both for fetch and data and write) is performed if the debug features (single-wire) is connected. No restriction is present on other areas: it is possible to read and write/erase the Option bytes area.
- All operations are possible when the boot is done in the Flash program memory.
- The mass erase deletes the Flash program memory and changes protection level to level 0, deletes the third Option byte and then rewrites it to enable Level 0 and disable PCROP (wprmod = 0).

6.5.2 PcROP (Proprietary Code Read-Out Protection)

The Flash program memory can be protected from being read by a hacking code: the read data are blocked (not for a fetch). The protected code must not access data in the protected zone, including the literal pool. The Flash program memory can be protected against a hacking code read: this blocks the data read (not for a fetch), assuming that the native code is compiled according to the PcROP option. This mode is activated setting wprmod = 1 in the FLASH_OTPR register.

The protection granularity is the sector (1 sector = 4 pages = 4 KB). To protect a sector, set to 0 the right bit in the wrprot[63:0] configuration: 0 means read and write protection, 1 means no protection. Table 14 shows the link between the bits of the wrprot[15:0] configuration and the address of the 64KB Flash memory 16 sectors.

Any read access performed as data (see “Read as data and pre-read” section) in a protected sector will trigger the rderr[0] flag in the FLASH_SR register. Any read-protected sector is also write-protected and any write access to one of these sectors will trigger the wrperr[0] flag in the FLASH_SR register.

Table 15 PcROP & wrprot Flash Memory Protected Range

Bit	Start address	End address	Bit	Start address	End address
0	1000_0000h	1000_0FFFh	16	1001_0000h(Reserve)	1001_0FFFh(Reserve)
1	1000_1000h	1000_1FFFh	17	1001_1000h(Reserve)	1001_1FFFh(Reserve)
2	1000_2000h	1000_2FFFh	18		
3	1000_3000h	1000_3FFFh	19		
4	1000_4000h	1000_4FFFh	20		
5	1000_5000h	1000_5FFFh	21		
6	1000_6000h	1000_6FFFh	22		
7	1000_7000h	1000_7FFFh	23		
8	1000_8000h	1000_8FFFh	24		
9	1000_9000h	1000_9FFFh	25		
10	1000_A000h	1000_AFFFh	26		
11	1000_B000h	1000_BFFFh	27		
12	1000_C000h	1000_CFFFh	28		
13	1000_D000h	1000_DFFFh	29		
14	1000_E000h	1000_EFFFh
15	1000_F000h	1000_FFFFh	63	1003_F000h(Reserve)	1003_FFFFh(Reserve)

When wprmod = 1 (PcROP enabled), it is not possible to reduce the protection on a sector: new zeros (to protect new sectors) can be set, but new ones (to remove the protection from sectors) cannot be added.

This is valid regardless of the protection level (RDPROT configuration). When wprmod is active, if the user tries to reset wprmod or to remove the protection from a sector, the programming is launched but wprmod or protected sectors remain unchanged.

The only way to remove a protection from a sector is to request a mass erase (which changes the protection level to 0 and disables PcROP): when PcROP is disabled, the protection on sectors can be changed freely.

6.5.3 Protections Against unwanted Write/Erase Operations

The memory interface implements two ways to protect against unwanted write/erase operations which are valid for all matrix or only for specific sectors of the Flash program memory.

As explained in the “Unlocking/locking operations” section, the user can:

- Write/erase to the Option bytes area only when pelock = 0 and optlock[0] = 0 in the FLASH_PECR register.
- Write/erase to the Flash program memory only when pelock = 0 and prglock[0] = 0 in the FLASH_PECR register.

To see the sequences to set pelock, prglock[0] and optlock[0], refer to the “Unlocking the FLASH_PECR register,” “Unlocking the Flash program memory”-sections.

In the Flash program memory, it is possible to add another write protection with the sector granularity. When PcROP is disabled (wprmod = 0), the bits of wrprot[15:0] are used to enable the write protection on the sectors. The polarity is opposed relatively to PcROP: to protect a sector, it is necessary to set the bit to 1; to remove the protection, it is necessary to set the bit to 0. Table 15 is valid for a write protection as well. As explained, when PcROP is enabled, the sectors protected against read are also protected against write/erase. It is always possible to change the write protection on sectors both in Level 0 and Level 1. Table 16 resumes the protections.

Table 16 Memory access vs mode

Flash program memory sectors	Mode		
	User mode (including In Application Programming)	Debug mode	
Flash program memory (RDP Level Setting)	R/W (Level 1 or 0)	Level 0 (rdprot[7:0]=0XAA)	Level 1 (rdprot[7:0]≠0XAA)
		R/W	Protected (no access)
Flash program memory (prglock[0] = 1)	R	R	Protected (no access)
Flash program memory (prglock[0] = 0)	R/W	R/W	Protected (no access)
Flash program memory in WRP pages	R	R	Protected (no access)
Flash program memory in PcROP pages	Fetch	Fetch	Protected (no access)

6.5.4 Reading the NVM

Here is a summary of the rules to change all previous protections:

- When in Level 1, the protection can be reduced to Level 0 after issuing mass erase.
- PcROP can be removed on requesting a mass erase.
- When PcROP is enabled, protected sectors can be added (writing 0) but cannot be removed. A mismatch concerns all read- and write-protected sectors (if PcROP is enabled).

6.5.5 Protection Errors

- Write protection error flag (wrperr[0])

If an erase/program operation to a write-protected page of the Flash program memory, the Write Protection Error flag (wrperr[0]) is set in the FLASH_SR register. Consequently, the wrperr[0] flag is set when the software tries to:

- Write to a WRP page.
- Write to factory option bytes.
- Write to the Flash program memory or Option bytes if they are not unlocked by PEKEY, PRGKEY

A write-protection error aborts the write/erase operation and an interrupt can be generated (if errie[0] = 1 in the FLASH_PECR register).

To reset this flag, the software needs to write it to 1.

- Read error (rderr[0])

If the software tries to read a sector protected by PcROP, the rderr[0] flag of flash_pecr is raised. The data received on the bus is at 0. If the error interrupt is enabled (errie[0] = 1 in the FLASH_PECR register), an interrupt is generated.

To reset this flag, the software needs to write it to 1.

6.6 NVM Interrupts

Setting the End of programming interrupt enable bit (eopie[0]) in the FLASH_PECR register enables an interrupt generation when an erase or a programming operation ends successfully. In this case, the End of programming (eop) bit in the FLASH_SR register is set. To reset it, the software needs to write it to 1.

Setting the Error interrupt enable bit (errie[0]) in the FLASH_PECR register enables an interrupt generation if an error occurs during a programming or an erase operation request. In this case, one or several error flags are set in the FLASH_SR register:

- rderr[0] (PCROP Read protection error flags)
- wrperr[0] (Write protection error flags)
- optverr[0] (Option validity error flag)

To reset the error flag, the software needs to write the right flag to 1.

Table 17 Flash interrupt request

Interrupt event	Event flag	Enable control bit
End of operation	eop	eopie[0]
Error	rderr[0] wrperr[0] optverr[0]	errie[0]

6.6.1 Hard Fault

A hard fault will be generated on the following conditions:

- Trying to read any area that has been set as read protection on the memory bus
- Trying to write an incorrect value in FLASH_PEKEYR, FLASH_PRGKEYR

6.7 Memory Interface Management

The purpose of this section is to clarify what happens when one operation is requested while another is ongoing: the way the different operations work together and are managed by the memory interface.

6.7.1 Operation Priority and Evolution

There are three types of operations and each of them has different flows:

- Read
 - If no operation is ongoing and the read address is not protected, the read is executed without delaying and with the actual configurations.
 - If the read address is protected, the operation is filtered (the read requested is never sent to the memory) and an error is raised.
 - If the read address is not protected but the memory interface is busy and cannot perform the operation, the read is put on hold to be executed as soon as possible.
- Write/erase
 - If no operation is ongoing and the write address is not protected, the write/erase will start immediately; after some clock pulses during which the bus and the master are blocked, the memory interface continues the operation freeing the bus and the master.
 - If the address is protected, the write/erase is filtered (the write/erase requested is never sent to the memory) and an error is raised.
 - If the address is not protected but one or several conditions are not met, the operation is aborted (the abort needs more time to be executed because the NVM need to return to default configuration) and an error is raised.
 - If the address to write/erase is not protected and all rules are respected, and if the memory interface is busy, the operation is put on hold to be executed as soon as possible.

6.7.2 Sequence of Operations

- Read as data while write

If the master requests a read as data (see “Read as data and pre-read” section) while a write operation is ongoing, there are three different cases:

- If the read is in a protected area, the `rderr[0]` flag is raised and the write operation continues.
- If the write operation uses a Single programming operation, the read is put on hold and will be executed when the write operation is complete. Most important of all, during the time of waiting read to be executed, the master is blocked and no other operation can be executed until the write and read operations are complete.

➤ Fetch while write

If the master fetches an instruction while a write is ongoing, the situation is similar to a read as data (see first two conditions in “Read as data while write” section).

➤ Write while another write operation is ongoing

If the master requests a write operation while another one is ongoing, there are different cases:

- If the previous write uses a Single programming operation, and if the new write is in a protected area, the WRPERR flag is triggered, the previous write continues and the new write is deleted.
- If the previous write uses a Single programming operation, and if the new Single programming operation is not in a protected area, the new write is put on hold and will be executed when the first write operation is complete. Please note that the master who requested the second write is blocked until the first write completes and the second has stored the address and data internally.
- It is forbidden to request a new write when a mass erase is ongoing: during all the steps of the mass erase, the data is not stored internally and the new data can change the value stored as a protection, adding unwanted protections.
- It is possible to modify configurations to prepare a new write operation when the first operation uses a Single programming operation.

6.7.3 Change the Number of Wait-States while Reading

To change the number of wait states, it is necessary to write to the FLASH_ACR register. The read/write of a register uses a different interface than the memory read/write. The number of wait-states cannot be changed while the memory interface is reading and the memory interface cannot be stopped if a request is sent to the register interface.

Therefore, while a master is reading the memory and another master is changing the number of the wait-states, the register interface will be locked until the change takes effect (until the readings stop). To stop the master which is changing the number of wait-states, be sure to read back the content of the FLASH_ACR register again. Because it is impossible to know the number of clock cycles required to change the number of wait-states which are depended on the customer code.

6.8 Option Bytes

On the NVM, an area is reserved to store a set of Option bytes which are used to configure the product. Some option bytes are written in factory while others can be configured by the end user.

The configuration managed by an end user is stored in the Option bytes area. Please pay attention to the execution of the boot program. This boot program occurs after a power-on reset, or by reloading the option bytes by software. The Option bytes are automatically loaded during the boot, and they are used to set the content of the FLASH_OPTR and FLASH_WRPROT registers.

6.8.1 API for Option Bytes

Several important configurations in Option bytes area can be read or modified via a set of API library. The all API functions are listed in Table 18.

Table 18 API Functions

API Functions	Description	Example Code
FLASH_OB_LevelUpdate	Update the configuration of RDP (Read Out Protection) level in the Option Bytes area.	FLASH_OB_LEVEL
FLASH_OB_PcropUpdate	Update the configuration of PcROP (Read Out Protection) in the Option Bytes area.	FLASH_OB_WRITE_PROTECTION & FLASH_OB_READ_PROTECTION
FLASH_OB_EepromWrite	Write data to the EEPROM data in the Option Bytes area (max 512 bytes).	FLASH_OB_EEPROM
FLASH_OB_EepromRead	Read data to the EEPROM data in the Option Bytes area (max 512 bytes).	FLASH_OB_EEPROM

The end user must select the software component 'FLASHEXT' in the IDE before uses them. The category of this software component is the Standard-Peripheral-Driver. It consists of two files – 'WT32L064/032_flashext.h' and 'WT32L064/032_flashext.lib', and they can be found under the CMSIS PACK installation path.

About how to using these functions, please refer to the related example code in the CMSIS PACK

6.8.2 Mismatch when loading protection flags

When there is a mismatch during an Option byte loading, the memory interface sets the default value in registers. In the Option byte area, there are three kinds of protection information:

➤ rdprot[7:0]

This configuration sets the Protection Level. As explained in the next section, changing this level changes the possibility to access the NVM and the product. The default value is Level 1. It is possible to return to Level 0 from Level 1 but all contents of the Flash program memory will be deleted (mass erase).

➤ wprmod

This flag is independent from rdprot[7:0] and set if the Flash program memory is protected from reading or writing. When this flag is 1 (read protection), the only way to reset it is to request a mass erase. The default value is 1 (read protection) and a mismatch on this bit also generates the default value for the wrprot[63:0] configuration.

➤ wrprot[63:0]

This configuration sets which pages of the Flash program memory are read- or write-protected. If the read protection is disabled (wprmod = 0), 1 must be set in the right bit to protect a sector. If the read protection is enabled (wprmod = 1), 0 must be in the right bit to protect a sector. If during boot there is a mismatch on wprmod, this configuration is loaded with zeros so that all sectors of the Flash program memory are protected from reading. If wprmod has been read correctly but there is a mismatch reading wrprot[63:0], the register will be loaded with zeros if wprmod = 1, and with ones if wprmod = 0.

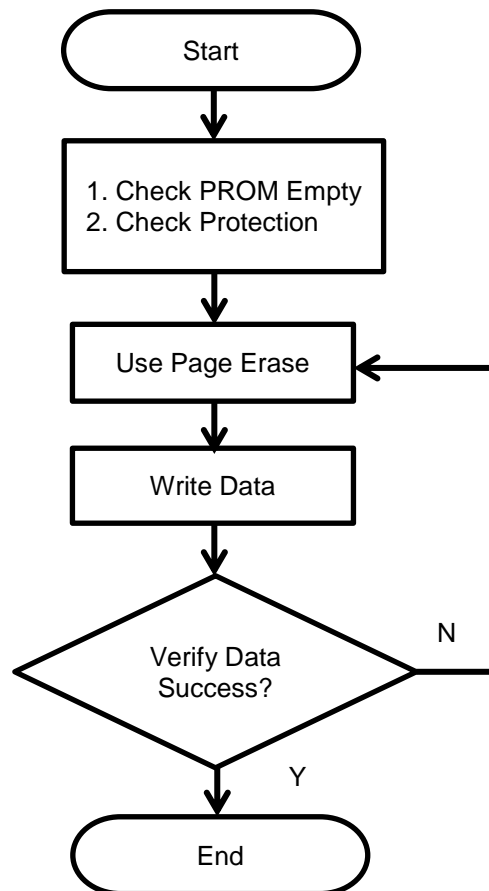
Thus, a mismatch on a protection can have a serious impact on the normal execution of code (if it is in the Flash program memory): when there is a read protection, only a fetch is possible. In the Flash program memory, some values are read as data (the constants, for example) during a code execution; protecting all sectors from read prevents the execution of the application code from the Flash program memory.

6.8.3 Simulated EEPROM

6.8.3.1 Description

The WT32L064/032 can use Flash PROM space to emulate E²PROM. @ VDD33=1.8V~3.6V
Operating Process:

1. Selected PROM area without program code.
2. Confirm that there is no write or anti-read protection in this area (see section 6.5 introduction).
3. Perform Page Erase to clear the areas.
4. Write with Byte, Half-Word, or Word data length.
5. Read and verify with Byte, Half-Word, or Word data length.
6. EEPROM data can be sent and received through peripherals such as UART and I2C.



7 DMA

7.1 Features

- Compliant with AMBA v2.0
 - AHB slave interface for DMA controller configuration.
 - AHB master interface for data transfers.
 - Transfer type – Single Mode
 - 32-bits (word), 16-bits (half-word), 8-bits (byte) wide data transaction
- 7 configurable DMA channels
 - Support memory-to-peripheral transfer
 - Support peripheral-to-memory transfer
 - Support peripheral-to-peripheral transfer
- Peripherals supported (**Note: The peripheral hardware system clock APB0 or APB1 only supports AHB/1, AHB/2, instead of AHB/4, AHB/8 or AHB/16**)
 - 15 sets requests/acknowledges hardware handshake
 - UARTs (Tx/Rx)
 - Timers
 - IICs(Tx/Rx)
 - ADC
 - SPI(Tx/Rx)
 - USB
 - I2S(Tx/Rx)
- Arbitration scheme
 - Round-robin arbitration.
 - Configurable 4-level priority.
- Circular Mode

7.2 Function description

The Direct Memory Access Controller is enhanced the system performance and reduce the processor-interrupt generation. There are 7 configurable channels for memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral transfers. Each channel is connected to dedicated hardware handshake signal.

7.2.1 Block Diagram

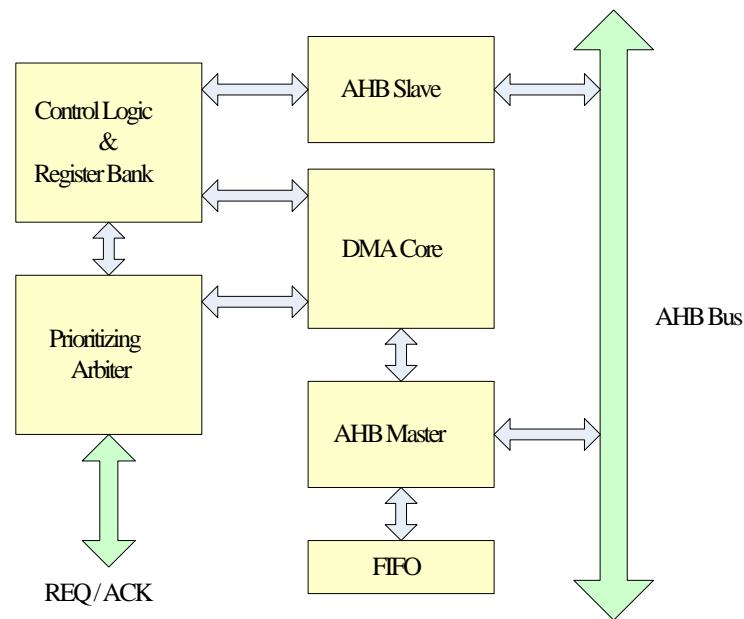


Figure 26 DMA Block Diagram

7.2.2 AHB Master Interface

The system can transfer data on AHB bus through this AHB master interface.

7.2.3 AHB Slave Interface

The system can configure the DMA controller or access the devices on AHB bus through this AHB slave interface.

7.2.4 FIFO Buffer

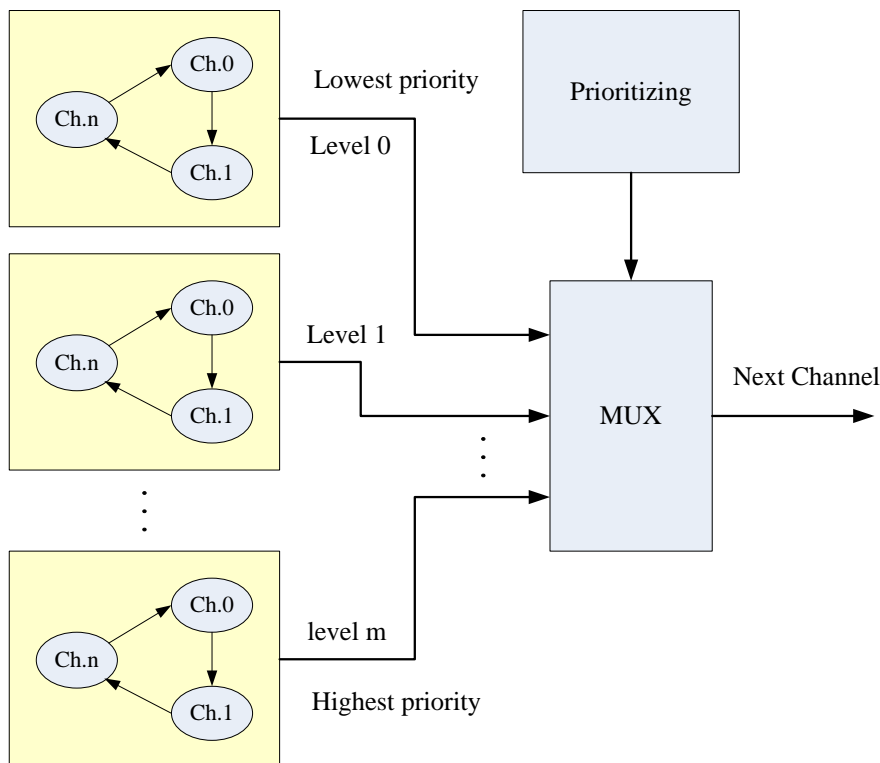
The FIFO buffer provides the data transfer buffer between the source and the destination.

7.2.5 DMA Core

The DMA data transfer engine.

7.2.6 Prioritizing Arbiter

To handle hardware handshake signals to start DMA transfer and configure up to 7-channels. They can group the round-robin arbitration scheme into 4 priority levels.



7.2.7 Control Logic & Register bank

Register set from AHB slave interface and generate some control logic for DMA transfer.

7.3 DMA Register Table

7.3.1 Global setting for interrupt status / clear, Channel Busy

Base Address: 0x0030_0000h

Index	Default	R/W	Bit	Name	Description
DMA_BUSY: DMA busy					
04			31:7		Reserved
	0h	RO	6	dma_busy6	1: Channel 6 busy 0: Channel 6 available
	0h	RO	5	dma_busy5	1: Channel 5 busy 0: Channel 5 available
	0h	RO	4	dma_busy4	1: Channel 4 busy 0: Channel 4 available
	0h	RO	3	dma_busy3	1: Channel 3 busy 0: Channel 3 available
	0h	RO	2	dma_busy2	1: Channel 2 busy 0: Channel 2 available
	0h	RO	1	dma_busy1	1: Channel 1 busy 0: Channel 1 available

Index	Default	R/W	Bit	Name	Description
	0h	RO	0	dma_busy0	1: Channel 0 busy 0: Channel 0 available
DMA_ISR: Interrupt status					
08			31:28		Reserved
	0	RO	27	err_int6	Set by hardware, clear by software 0: No transfer error event 1: Transfer error
	0	RO	26	half_int6	0: No half_int event on channel 6 1: A half_int event on channel 6
	0	RO	25	total_int6	0: No total_int event on channel 6 1: A total_int event on channel 6
	0	RO	24	dma_int6	0: No half_int or total_int event on channel 6 1: A err_int, half_int or total_int event on channel 6
		RO	23	err_int5	Set by hardware, clear by software 0: No transfer error event 1: Transfer error
	0	RO	22	half_int5	0: No half_int event on channel 5 1: A half_int event on channel 5
	0	RO	21	total_int5	0: No total_int event on channel 5 1: A total_int event on channel 5
	0	RO	20	dma_int5	0: No half_int or total_int event on channel 5 1: A err_int, half_int or total_int event on channel 5
		RO	19	err_int4	Set by hardware, clear by software 0: No transfer error event 1: Transfer error
	0	RO	18	half_int4	0: No half_int event on channel 4 1: A half_int event on channel 4
	0	RO	17	total_int4	0: No total_int event on channel 4 1: A total_int event on channel 4
	0	RO	16	dma_int4	0: No half_int or total_int event on channel 4 1: A err_int, half_int or total_int event on channel 4
		RO	15	err_int3	Set by hardware, clear by software 0: No transfer error event 1: Transfer error
	0	RO	14	half_int3	0: No half_int event on channel 3 1: A half_int event on channel 3
	0	RO	13	total_int3	0: No total_int event on channel 3 1: A total_int event on channel 3
	0	RO	12	dma_int3	0: No half_int or total_int event on channel 3 1: A err_int, half_int or total_int event on channel 3
		RO	11	err_int2	Set by hardware, clear by software 0: No transfer error event 1: Transfer error
	0	RO	10	half_int2	0: No half_int event on channel 2 1: A half_int event on channel 2
	0	RO	9	total_int2	0: No total_int event on channel 2 1: A total_int event on channel 2
	0	RO	8	dma_int2	0: No half_int or total_int event on channel 2 1: A err_int, half_int or total_int event on channel 2
		RO	7	err_int1	Set by hardware, clear by software 0: No transfer error event 1: Transfer error

Index	Default	R/W	Bit	Name	Description
	0	RO	6	half_int1	0: No half_int event on channel 1 1: A half_int event on channel 1
	0	RO	5	total_int1	0: No total_int event on channel 1 1: A total_int event on channel 1
	0	RO	4	dma_int1	0: No half_int or total_int event on channel 1 1: A err_int, half_int or total_int event on channel 1
		RO	3	err_int0	Set by hardware, clear by software 0: No transfer error event 1: Transfer error
	0	RO	2	half_int0	0: No half_int event on channel 0 1: A half_int event on channel 0
	0	RO	1	total_int0	0: No total_int event on channel 0 1: A total_int event on channel 0
	0	RO	0	dma_int0	0: No half_int or total_int event on channel 0 1: A err_int, half_int or total_int event on channel 0
DMA_CLR_F: Clear flags					
0C			31:28		Reserved
		WO	27	Clr_err_int6	0: No effect 1: Clear err_int6 flag
	0	WO	26	clr_half_int6	0: No effect 1: Clear half_int6 flag
	0	WO	25	clr_total_int6	0: No effect 1: Clear total_int6 flag
	0	WO	24	clr_int6	0: No effect 1: Clears dma_int6, err_int6, total_int6 and half_int6 flags
		WO	23	Clr_err_int5	0: No effect 1: Clear err_int5 flag
	0	WO	22	clr_half_int5	0: No effect 1: Clear half_int5 flag
	0	WO	21	clr_total_int5	0: No effect 1: Clear total_int5 flag
	0	WO	20	clr_int5	0: No effect 1: Clears dma_int5, err_int5, total_int5 and half_int5 flags
		WO	19	Clr_err_int4	0: No effect 1: Clear err_int4 flag
	0	WO	18	clr_half_int4	0: No effect 1: Clear half_int4 flag
	0	WO	17	clr_total_int4	0: No effect 1: Clear total_int4 flag
	0	WO	16	clr_int4	0: No effect 1: Clears dma_int4, err_int4, total_int4 and half_int4 flags
		WO	15	Clr_err_int3	0: No effect 1: Clear err_int3 flag
	0	WO	14	clr_half_int3	0: No effect 1: Clear half_int3 flag
	0	WO	13	clr_total_int3	0: No effect 1: Clear total_int3 flag
	0	WO	12	clr_int3	0: No effect 1: Clears dma_int3, err_int3, total_int3 and half_int3 flags
		WO	11	Clr_err_int2	0: No effect 1: Clear err_int2 flag
	0	WO	10	clr_half_int2	0: No effect 1: Clear half_int2 flag

Index	Default	R/W	Bit	Name	Description
	0	WO	9	clr_total_int2	0: No effect 1: Clear total_int2 flag
	0	WO	8	clr_int2	0: No effect 1: Clears dma_int2, err_int2, total_int2 and half_int2 flags
		WO	7	Clr_err_int1	0: No effect 1: Clear err_int1 flag
	0	WO	6	clr_half_int1	0: No effect 1: Clear half_int1 flag
	0	WO	5	clr_total_int1	0: No effect 1: Clear total_int1 flag
	0	WO	4	clr_int1	0: No effect 1: Clears dma_int1, err_int1, total_int1 and half_int1 flags
		WO	3	Clr_err_int0	0: No effect 1: Clear err_int0 flag
	0	WO	2	clr_half_int0	0: No effect 1: Clear half_int0 flag
	0	WO	1	clr_total_int0	0: No effect 1: Clear total_int0 flag
	0	WO	0	clr_int0	0: No effect 1: Clears dma_int0, err_int0, total_int0 and half_int0 flags

7.3.2 DMA Channel x source AddrESS register

X = 0 ~ 6, where x is channel number

Channel X Address index: $0x10 + 0x10 * (\text{channel number } X)$

Index	Default	R/W	Bit	Name	Description
DMA_SR_ADDR0: Source address of channel 0					
10	0	R/W	31:0	dma_saddr0	Channel 0 source address
DMA_SR_ADDR1: Source address of channel 1					
20	0	R/W	31:0	dma_saddr1	Channel 1 source address
DMA_SR_ADDR2: Source address of channel 2					
30	0	R/W	31:0	dma_saddr2	Channel 2 source address
DMA_SR_ADDR3: Source address of channel 3					
40	0	R/W	31:0	dma_saddr3	Channel 3 source address
DMA_SR_ADDR4: Source address of channel 4					
50	0	R/W	31:0	dma_saddr4	Channel 4 source address
DMA_SR_ADDR5: Source address of channel 5					
60	0	R/W	31:0	dma_saddr5	Channel 5 source address
DMA_SR_ADDR6: Source address of channel 6					
70	0	R/W	31:0	dma_saddr6	Channel 6 source address

7.3.3 DMA channel x destination address register

X = 0 ~ 6, where x is channel number

Channel X Address index: $0x14 + 0x10 * (\text{channel number } X)$

Index	Default	R/W	Bit	Name	Description
DMA_DT_ADDR0: Destination address of channel 0					
14	0	R/W	31:0	dma_daddr0	Channel 0 destination address

Index	Default	R/W	Bit	Name	Description
DMA_DT_ADDR1: Destination address of channel 1					
24	0	R/W	31:0	dma_daddr1	Channel 1 destination address
DMA_DT_ADDR2: Destination address of channel 2					
34	0	R/W	31:0	dma_daddr2	Channel 2 destination address
DMA_DT_ADDR3: Destination address of channel 3					
44	0	R/W	31:0	dma_daddr3	Channel 3 destination address
DMA_DT_ADDR4: Destination address of channel 4					
54	0	R/W	31:0	dma_daddr4	Channel 4 destination address
DMA_DT_ADDR5: Destination address of channel 5					
64	0	R/W	31:0	dma_daddr5	Channel 5 destination address
DMA_DT_ADDR6: Destination address of channel 6					
74	0	R/W	31:0	dma_daddr6	Channel 6 destination address

7.3.4 DMA channel x number of data register

X = 0 ~ 6, where x is channel number

Channel X Address index: 0x18 + 0x10 * (channel number X)

Index	Default	R/W	Bit	Name	Description
DMA_LENGTH0: Total data length of channel 0					
18			31:9		Reserved
	0	R/W	8:0	dma_lenth0	Channel 0 The total size transfer length: 1 ~ 511 0: DMA transfer stop
DMA_LENGTH1: Total data length of channel 1					
28			31:9		Reserved
	0	R/W	8:0	dma_lenth1	Channel 1 The total size transfer length: 1 ~ 511 0: DMA transfer stop
DMA_LENGTH2: Total data length of channel 2					
38			31:9		Reserved
	0	R/W	8:0	dma_lenth2	Channel 2 The total size transfer length: 1 ~ 511 0: DMA transfer stop
DMA_LENGTH3: Total data length of channel 3					
48			31:9		Reserved
	0	R/W	8:0	dma_lenth3	Channel 3 The total size transfer length: 1 ~ 511 0: DMA transfer stop
DMA_LENGTH4: Total data length of channel 4					
58			31:9		Reserved
	0	R/W	8:0	dma_lenth4	Channel 4 The total size transfer length: 1 ~ 511 0: DMA transfer stop
DMA_LENGTH5: Total data length of channel 5					
68			31:9		Reserved
	0	R/W	8:0	dma_lenth5	Channel 5 The total size transfer length: 1 ~ 511 0: DMA transfer stop
DMA_LENGTH6: Total data length of channel 6					
78			31:9		Reserved
	0	R/W	8:0	dma_lenth6	Channel 6 The total size transfer length: 1 ~ 511 0: DMA transfer stop

7.3.5 DMA channel x configuration register

X = 0 ~ 6 where x is channel number

Channel X Address index: $0x1C + 0x10 * (\text{channel number } X)$

Index	Default	R/W	Bit	Name	Description
DMA_CFG0: Configuration of channel 0					
1C			31:16		Reserved
	0	RW	15:14	Src_width0	Channel 0 source transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved
	0	RW	13:12	Dest_width0	Channel 0 destination transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved
			11		Reserved
	0	RW	10	circ_mode0	Channel 0 Circular Buffer Mode 1: Circular buffer mode enabled 0: Circular buffer mode disabled
	0	R/W	9:8	pri_ch0	Channel 0 priority level: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (Default)
	0	R/W	7:6	src_adr0_ctl	Channel 0 source address control: 3: Reserved 2: Fixed source address 1: Reserved 0: Increment source address (Default)
	0	R/W	5:4	dst_adr0_ctl	Channel 0 destination address control: 3: Reserved 2: Fixed destination address 1: Reserved 0: Increment destination address (Default)
	0	R/W	3	en_err_int0	Channel 0 error interrupt enable 1: Error Interrupt enable 0: Error Interrupt disable
	0	R/W	2	en_half_int0	Channel 0 half interrupt enable: 1: half Interrupt enable 0: half Interrupt disable
	0	R/W	1	en_total_int0	Channel 0 total interrupt enable: 1: total Interrupt enable 0: total Interrupt disable
	0	R/W	0	dma_chen0	Channel 0 write 1 enable.
DMA_CFG1: Configuration of channel 1					
2C			31:16		Reserved
	0	RW	15:14	Src_width1	Channel 1 source transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved

Index	Default	R/W	Bit	Name	Description
	0	RW	13:12	Dest_width1	Channel 1 destination transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved
			11		Reserved
	0	RW	10	circ_mode1	Channel 1 Circular Buffer Mode 1: Circular buffer mode enabled 0: Circular buffer mode disabled
	0	R/W	9:8	pri_ch1	Channel 1 priority level: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (Default)
	0	R/W	7:6	src_adr1_ctl	Channel 1 source address control: 3: Reserved 2: Fixed source address 1: Reserved 0: Increment source address (Default)
	0	R/W	5:4	dst_adr1_ctl	Channel 1 destination address control: 3: Reserved 2: Fixed destination address 1: Reserved 0: Increment destination address (Default)
	0	R/W	3	en_err_int1	Channel 1 error interrupt enable 1: Error Interrupt enable 0: Error Interrupt disable
	0	R/W	2	en_half_int1	Channel 1 half interrupt enable: 1: half Interrupt enable 0: half Interrupt disable
	0	R/W	1	en_total_int1	Channel 1 total interrupt enable: 1: total Interrupt enable 0: total Interrupt disable
	0	R/W	0	dma_chen1	Channel 1 write 1 enable
DMA_CFG2: Configuration of channel 2					
3C			31:16		Reserved
	0	RW	15:14	Src_width2	Channel 2 source transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved
	0	RW	13:12	Dest_width2	Channel 2 destination transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved
			11		Reserved
	0	RW	10	circ_mode2	Channel 2 Circular Buffer Mode 1: Circular buffer mode enabled 0: Circular buffer mode disabled
	0	R/W	9:8	pri_ch2	Channel 2 priority level: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (Default)

Index	Default	R/W	Bit	Name	Description
	0	R/W	7:6	src_adr2_ctl	Channel 2 source address control: 3: Reserved 2: Fixed source address 1: Reserved 0: Increment source address (Default)
	0	R/W	5:4	dst_adr2_ctl	Channel 2 destination address control: 3: Reserved 2: Fixed destination address 1: Reserved 0: Increment destination address (Default)
	0	R/W	3	en_err_int2	Channel 2 error interrupt enable 1: Error Interrupt enable 0: Error Interrupt disable
	0	R/W	2	en_half_int2	Channel 2 half interrupt enable: 1: half Interrupt enable 0: half Interrupt disable
	0	R/W	1	en_total_int2	Channel 2 total interrupt enable: 1: total Interrupt enable 0: total Interrupt disable
	0	R/W	0	dma_chen2	Channel 2 write 1 enable
DMA_CFG3: Configuration of channel 3					
4C			31:16		Reserved
	0	RW	15:14	Src_width3	Channel 3 source transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved
	0	RW	13:12	Dest_width3	Channel 3 destination transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved
			11		Reserved
	0	RW	10	circ_mode3	Channel 3 Circular Buffer Mode 1: Circular buffer mode enabled 0: Circular buffer mode disabled
	0	R/W	9:8	pri_ch3	Channel 3 priority level: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (Default)
	0	R/W	7:6	src_adr3_ctl	Channel 3 source address control: 3: Reserved 2: Fixed source address 1: Reserved 0: Increment source address (Default)
	0	R/W	5:4	dst_adr3_ctl	Channel 3 destination address control: 3: Reserved 2: Fixed destination address 1: Reserved 0: Increment destination address (Default)
	0	R/W	3	en_err_int3	Channel 3 error interrupt enable 1: Error Interrupt enable 0: Error Interrupt disable

Index	Default	R/W	Bit	Name	Description
	0	R/W	2	en_half_int3	Channel 3 half interrupt enable: 1: half Interrupt enable 0: half Interrupt disable
	0	R/W	1	en_total_int3	Channel 3 total interrupt enable: 1: total Interrupt enable 0: total Interrupt disable
	0	R/W	0	dma_chen3	Channel 3 write 1 enable
DMA_CFG4: Configuration of channel 4					
5C			31:16		Reserved
	0	RW	15:14	Src_width4	Channel 4 source transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved
	0	RW	13:12	Dest_width4	Channel 4 destination transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved
			11		Reserved
	0	RW	10	circ_mode4	Channel 4 Circular Buffer Mode 1: Circular buffer mode enabled 0: Circular buffer mode disabled
	0	R/W	9:8	pri_ch4	Channel 4 priority level: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (Default)
	0	R/W	7:6	src_adr4_ctl	Channel 4 source address control: 3: Reserved 2: Fixed source address 1: Reserved 0: Increment source address (Default)
	0	R/W	5:4	dst_adr4_ctl	Channel 4 destination address control: 3: Reserved 2: Fixed destination address 1: Reserved 0: Increment destination address (Default)
	0	R/W	3	en_err_int4	Channel 4 error interrupt enable 1: Error Interrupt enable 0: Error Interrupt disable
	0	R/W	2	en_half_int4	Channel 4 half interrupt enable: 1: half Interrupt enable 0: half Interrupt disable
	0	R/W	1	en_total_int4	Channel 4 total interrupt enable: 1: total Interrupt enable 0: total Interrupt disable
	0	R/W	0	dma_chen4	Channel 4 write 1 enable
DMA_CFG5: Configuration of channel 5					
6C			31:16		Reserved
	0	RW	15:14	Src_width5	Channel 5 source transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved

Index	Default	R/W	Bit	Name	Description
	0	RW	13:12	Dest_width5	Channel 5 destination transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved
			11		Reserved
	0	RW	10	circ_mode5	Channel 5 Circular Buffer Mode 1: Circular buffer mode enabled 0: Circular buffer mode disabled
	0	R/W	9:8	pri_ch5	Channel 5 priority level: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (Default)
	0	R/W	7:6	src_adr5_ctl	Channel 5 source address control: 3: Reserved 2: Fixed source address 1: Reserved 0: Increment source address (Default)
	0	R/W	5:4	dst_adr5_ctl	Channel 5 destination address control: 3: Reserved 2: Fixed destination address 1: Reserved 0: Increment destination address (Default)
	0	R/W	3	en_err_int5	Channel 5 error interrupt enable 1: Error Interrupt enable 0: Error Interrupt disable
	0	R/W	2	en_half_int5	Channel 5 half interrupt enable: 1: half Interrupt enable 0: half Interrupt disable
	0	R/W	1	en_total_int5	Channel 5 total interrupt enable: 1: total Interrupt enable 0: total Interrupt disable
	0	R/W	0	dma_chen5	Channel 5 write 1 enable
DMA_CFG6: Configuration of channel 6					
7C			31:16		Reserved
	0	RW	15:14	Src_width6	Channel 6 source transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved
	0	RW	13:12	Dest_width6	Channel 6 destination transfer data width 00: 8bits 01: 16bits 10: 32bits 11: Reserved
			11		Reserved
	0	RW	10	circ_mode6	Channel 6 Circular Buffer Mode 1: Circular buffer mode enabled 0: Circular buffer mode disabled
	0	R/W	9:8	pri_ch6	Channel 6 priority level: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (Default)

Index	Default	R/W	Bit	Name	Description
	0	R/W	7:6	src_adr6_ctl	Channel 6 source address control: 3: Reserved 2: Fixed source address 1: Reserved 0: Increment source address (Default)
	0	R/W	5:4	dst_adr6_ctl	Channel 6 destination address control: 3: Reserved 2: Fixed destination address 1: Reserved 0: Increment destination address (Default)
	0	R/W	3	en_err_int6	Channel 6 error interrupt enable 1: Error Interrupt enable 0: Error Interrupt disable
	0	R/W	2	en_half_int6	Channel 6 half interrupt enable: 1: half Interrupt enable 0: half Interrupt disable
	0	R/W	1	en_total_int6	Channel 6 total interrupt enable: 1: total Interrupt enable 0: total Interrupt disable
	0	R/W	0	dma_chen6	Channel 6 write 1 enable

8 GPIO

8.1 Main Features

Each general-purpose I/O port (PA, PB, PC, PD) has three configuration registers (gpio_px_mode, gpio_px_ot and gpio_px_pupd), two data registers (gpio_px_out and gpio_px_in) and set/reset/toggle registers (gpio_px_br, gpio_px_bs and gpio_px_bt). In addition, all GPIOs have alternate function selection registers (gpio_px_af).

- AHB I/F, support word access with single burst
- Output states: push-pull or open-drain + pull-up/down
- Output data from output data register (gpio_px_out) or peripheral (alternate function output)
- Input states: floating, pull-up/down, analog
- Input data to input data register (gpio_px_in) or peripheral (alternate function input)
- Analog function
- Alternate function selection registers
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

8.2 Block Diagram

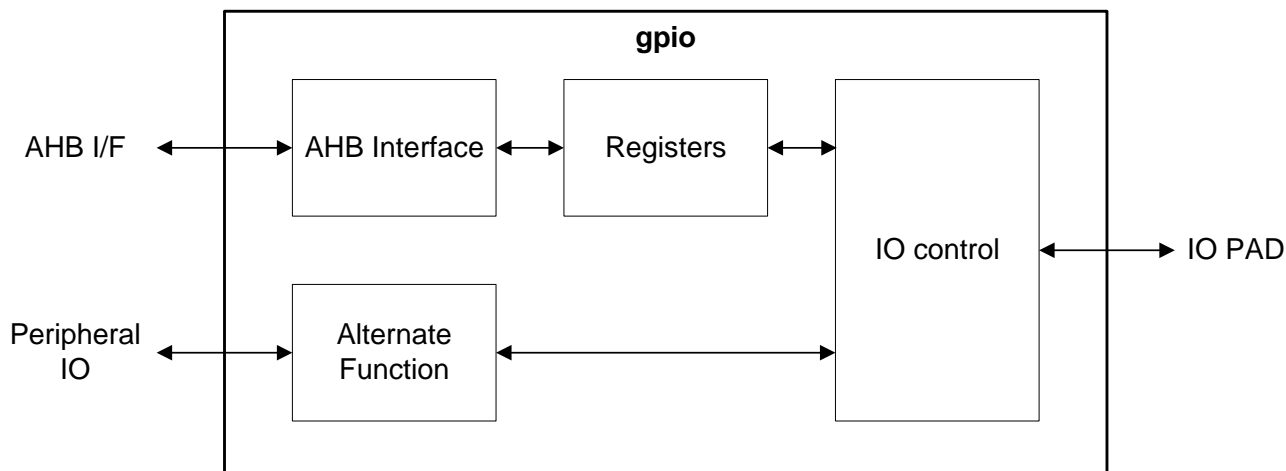


Figure 27 GPIO Block Diagram

8.3 Register Definition

Table 19 GPIO Control Register

Index	Bit	R/W	Default	Name	Description
GPIOA_MODE: Pad mode register					
000h	31:0	R/W	ebff_0fffh	gpio_pa_mode	PA mode register for port 0~15 These bits are written by software to configure the I/O mode. [2y+1:2y]: mode configuration for port x.

Index	Bit	R/W	Default	Name	Description
					2'b00: Input mode 2'b01: General purpose output mode 2'b10: Alternate function mode 2'b11: Analog mode Note: by default, PA[13], PA[14] are alternate function. PA[6], PA[7] are input mode.
Reserved					
004h	31:0	-	-	-	Reserved
GPIOA_PUPD: Pad pull-up/pull-down register					
008h	31:0	R/W	2400_5000h	gpio_pa_pupd	PA pull-up/pull-down resistor These bits are written by software to configure the I/O pull-up or pull-down [2y+1:2y]: pull-up/pull-down for port x. 2'b00: No pull-up, pull-down 2'b01: Pull-up 2'b10: Pull-down 2'b11: Reserved Note: by default, PA[13] is pull-up; PA[14] is pull-down. PA[6] and PA[7] are pull-up.
Reserved					
00ch	31:0	-	-	-	Reserved
GPIOA_DO: Pad output data					
010h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pa_out	PA output data
GPIOA_DI: Pad input data					
014h	31:16	-	-	-	Reserved
	15:0	R	-	gpio_pa_in	PA input data
GPIOA_OTYPE: Pad output type					
018h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pa_ot	PA output type 0: Push-pull 1: Open-drain Note: In open-drain type, if gpio_pa_pupd is enabled as pull-up, logic 1 is driven on pin by internal pull-up resistor; otherwise, logic 1 should be driven by external pull-up resistor.
GPIOA_DS: Pad driving strength					
01ch	15:0	W	-	gpio_pa_ds	PA driving strength 0: low 1: high
GPIOA_BT_RST: Pad bit reset					
020h	31:16	-	-	-	Reserved
	15:0	W	-	gpio_pa_br	PA bit reset 0: No effect 1: Reset correspond bit of gpio_pa_out as "0"
GPIOA_BT_SET: Pad bit set					
024h	31:16	-	-	-	Reserved
	15:0	W	-	gpio_pa_bs	PA bit set 0: No effect 1: Set correspond bit of gpio_pa_out as "1"
GPIOA_BT_TGLE: Pad bit toggle					

Index	Bit	R/W	Default	Name	Description
028h	31:16	-	-	-	Reserved
	15:0	W	-	gpio_pa_bt	PA bit toggle 0: No effect 1: Toggle correspond bit of gpio_pa_out "0->1" or "1->0"
GPIOA_AF0: Pad alt function					
040h	31:0	R/W	0	gpio_pa_af0	PA alternate function for port 0~7 These bits are written by software to configure alternate function I/Os [4y+3:4y]: mode configuration for port x. 0: Alternate function 0 (AF0) 1: Alternate function 1 (AF1) 2: Alternate function 2 (AF2) 3: Alternate function 3 (AF3) 4: Alternate function 4 (AF4) 5: Alternate function 5 (AF5) 6~15: Reserved
GPIOA_AF1: Pad alt function					
044h	31:0	R/W	0	gpio_pa_af1	PA alternate function for port 8~15 These bits are written by software to configure alternate function I/Os [4y+3:4y]: mode configuration for port x. 0: Alternate function 0 1: Alternate function 1 2: alternate function 2 3: Alternate function 3 4: Alternate function 4 5: Alternate function 5 6~15: Reserved
Reserved					
048h	31:0	-	-	-	Reserved
Reserved					
04ch	31:0	-	-	-	Reserved
GPIOA_IE: Pad interrupt enable					
060h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pa_ie	PA interrupt function 0: Disable 1: Enable
GPIOA_ISS: Pad interrupt sense selection					
064h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pa_iss	PA interrupt sense select 0: Edge sensitive 1: Level sensitive
GPIOA_BET: Pad interrupt both edges trigger					
068h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pa_bet	PA interrupt both edges trigger 0: Interrupt condition is controlled by gpio_pa_trg 1: Enable both rising and falling edges trigger Note: this register will be ignored when level sensitive.
GPIOA_TRG: Pad interrupt trigger event select					

Index	Bit	R/W	Default	Name	Description
06ch	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pa_trg	PA trigger event select 0: Falling edge or low level trigger 1: Rising edge or high level trigger
GPIOA_IF: Pad interrupt raw flag					
070h	31:16	-	-	-	Reserved
	15:0	R/W1c	0	gpio_pa_if	PA interrupt raw flag This register indicates the status for GPIO raw interrupts. A GPIO interrupt is sent to the interrupt controller if the corresponding bit in gpio_pa_ie register is enabled. 0: No interrupt 1: Interrupt occurred
GPIOB_MODE: Pad mode register					
100h	31:0	R/W	ffff_ffff	gpio_pb_mode	PB mode register for port 0~15 These bits are written by software to configure the I/O mode. [2y+1:2y]: mode configuration for port x. 2'b00: Input mode 2'b01: General purpose output mode 2'b10: Alternate function mode 2'b11: Analog mode
Reserved					
104h	31:0	-	-	-	Reserved
GPIOB_PUPD: Pad pull-up/pull-down register					
108h	31:0	R/W	0	gpio_pb_pupd	PB pull-up/pull-down resistor These bits are written by software to configure the I/O pull-up or pull-down [2y+1:2y]: pull-up/pull-down for port x. 2'b00: No pull-up, pull-down 2'b01: Pull-up 2'b10: Pull-down 2'b11: Reserved
Reserved					
10ch	31:0	-	-	-	Reserved
GPIOB_DO: Pad output data					
110h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pb_out	PB output data
GPIOB_DI: Pad input data					
114h	31:16	-	-	-	Reserved
	15:0	R	-	gpio_pb_in	PB input data
GPIOB_OTYPE: Pad output type					
118h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pb_ot	PB output type 0: Push-pull 1: Open-drain Note: In open-drain type, if gpio_pb_pupd is enabled as pull-up, logic 1 is driven on pin by internal pull-up resistor; otherwise, logic 1 should be driven by external pull-up resistor.
GPIOB_DS: Pad driving strength					

Index	Bit	R/W	Default	Name	Description
11ch	15:0	W	-	gpio_pb_ds	PB driving strength 0: low 1: high
GPIOB_BT_RST: Pad bit reset					
120h	31:16	-	-	-	Reserved
	15:0	W	-	gpio_pb_br	PB bit reset 0: No effect 1: Reset correspond bit of gpio_pb_out as "0"
GPIOB_BT_SET: Pad bit set					
124h	31:16	-	-	-	Reserved
	15:0	W	-	gpio_pb_bs	PB bit set 0: No effect 1: Set correspond bit of gpio_pb_out as "1"
GPIOB_BT_TGLE: Pad bit toggle					
128h	31:16	-	-	-	Reserved
	15:0	W	-	gpio_pb_bt	PB bit toggle 0: No effect 1: Toggle correspond bit of gpio_pb_out "0->1" or "1->0"
GPIOB_AF0: Pad alt function					
140h	31:0	R/W	0	gpio_pb_af0	PB alternate function for port 0~7 These bits are written by software to configure alternate function I/Os [4y+3:4y]: mode configuration for port x. 0: Alternate function 0 (AF0) 1: Alternate function 1 (AF1) 2: Alternate function 2 (AF2) 3: Alternate function 3 (AF3) 4: Alternate function 4 (AF4) 5: Alternate function 5 (AF5) 6~15: Reserved
GPIOB_AF1: Pad alt function					
144h	31:0	R/W	0	gpio_pb_af1	PB alternate function for port 8~15 These bits are written by software to configure alternate function I/Os [4y+3:4y]: mode configuration for port x. 0: Alternate function 0 1: Alternate function 1 2: alternate function 2 3: Alternate function 3 4: Alternate function 4 5: Alternate function 5 6~15: Reserved
Reserved					
148h	31:0	-	-	-	Reserved
Reserved					
14ch	31:0	-	-	-	Reserved
GPIOB_IE: Pad interrupt enable					
160h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pb_ie	PB interrupt function 0: Disable 1: Enable

Index	Bit	R/W	Default	Name	Description
GPIOB_ISS: Pad interrupt sense selection					
164h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pb_iss	PB interrupt sense select 0: edge sensitive 1: level sensitive
GPIOB_BET: Pad interrupt both edges trigger					
168h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pb_bet	PB interrupt both edges trigger 0: Interrupt condition is controlled by gpio_pb_trg 1: Enable both rising and falling edges trigger Note: this register will be ignored when level sensitive.
GPIOB_TRG: Pad interrupt trigger event select					
16ch	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pb_trg	PB trigger event select 0: Falling edge or low level trigger 1: Rising edge or high level trigger
GPIOB_IF: Pad interrupt raw flag					
170h	31:16	-	-	-	Reserved
	15:0	R/W1c	0	gpio_pb_if	PB interrupt raw flag This register indicates the status for GPIO raw interrupts. A GPIO interrupt is sent to the interrupt controller if the corresponding bit in gpio_pb_ie register is enabled. 0: No interrupt 1: Interrupt occurred
GPIOC_MODE: Pad mode register					
200h	31:0	R/W	ffff_ffffh	gpio_pc_mode	PC mode register for port 0~15 These bits are written by software to configure the I/O mode. [2y+1:2y]: mode configuration for port x. 2'b00: Input mode 2'b01: General purpose output mode 2'b10: Alternate function mode 2'b11: Analog mode
Reserved					
204h	31:0	-	-	-	Reserved
GPIOC_PUPD: Pad pull-up/pull-down register					
208h	31:0	R/W	0	gpio_pc_pupd	PC pull-up/pull-down resistor These bits are written by software to configure the I/O pull-up or pull-down [2y+1:2y]: pull-up/pull-down for port x. 2'b00: No pull-up, pull-down 2'b01: Pull-up 2'b10: Pull-down 2'b11: Reserved
Reserved					
20ch	31:0	-	-	-	Reserved
GPIOC_DO: Pad output data					
210h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pc_out	PC output data

Index	Bit	R/W	Default	Name	Description
GPIOC_DI: Pad input data					
214h	31:16	-	-	-	Reserved
	15:0	R	-	gpio_pc_in	PC input data
GPIOC_OTYPE: Pad output type					
218h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pc_ot	PC output type 0: Push-pull 1: Open-drain Note: In open-drain type, if gpio_pc_pupd is enabled as pull-up, logic 1 is driven on pin by internal pull-up resistor; otherwise, logic 1 should be driven by external pull-up resistor.
GPIOC_DS: Pad driving strength					
21ch	15:0	W	-	gpio_pc_ds	PC driving strength 0: low 1: high
GPIOC_BT_RST: Pad bit reset					
220h	31:16	-	-	-	Reserved
	15:0	W	-	gpio_pc_br	PC bit reset 0: No effect 1: Reset correspond bit of gpio_pc_out as "0"
GPIOC_BT_SET: Pad bit set					
224h	31:16	-	-	-	Reserved
	15:0	W	-	gpio_pc_bs	PC bit set 0: No effect 1: Set correspond bit of gpio_pc_out as "1"
GPIOC_BT_TGLE: Pad bit toggle					
228h	31:16	-	-	-	Reserved
	15:0	W	-	gpio_pc_bt	PC bit toggle 0: No effect 1: Toggle correspond bit of gpio_pc_out "0->1" or "1->0"
GPIOC_AF0: Pad alt function					
240h	31:0	R/W	0	gpio_pc_af0	PC alternate function for port 0~7 These bits are written by software to configure alternate function I/Os [4y+3:4y]: mode configuration for port x. 0: Alternate function 0 (AF0) 1: Alternate function 1 (AF1) 2: Alternate function 2 (AF2) 3: Alternate function 3 (AF3) 4: Alternate function 4 (AF4) 5: Alternate function 5 (AF5) 6~15: Reserved
GPIOC_AF1: Pad alt function					
244h	31:0	R/W	0	gpio_pc_af1	PC alternate function for port 8~15 These bits are written by software to configure alternate function I/Os [4y+3:4y]: mode configuration for port x. 0: Alternate function 0 1: Alternate function 1 2: alternate function 2

Index	Bit	R/W	Default	Name	Description
					3: Alternate function 3 4: Alternate function 4 5: Alternate function 5 6~15: Reserved
Reserved					
248h	31:0	-	-	-	Reserved
Reserved					
24ch	31:0	-	-	-	Reserved
GPIOC_IE: Pad interrupt enable					
260h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pc_ie	PC interrupt function 0: Disable 1: Enable
GPIOC_ISS: Pad interrupt sense selection					
264h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pc_iss	PC interrupt sense select 0: Edge sensitive 1: Level sensitive
GPIOC_BET: Pad interrupt both edges trigger					
268h	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pc_bet	PC interrupt both edges trigger 0: Interrupt condition is controlled by gpio_pc_trg 1: Enable both rising and falling edges trigger Note: this register will be ignored when level sensitive.
GPIOC_TRG: Pad interrupt trigger event select					
26ch	31:16	-	-	-	Reserved
	15:0	R/W	0	gpio_pc_trg	PC trigger event select 0: Falling edge or low level trigger 1: Rising edge or high level trigger
GPIOC_IF: Pad interrupt raw flag					
270h	31:16	-	-	-	Reserved
	15:0	R/W1c	0	gpio_pc_if	PC interrupt raw flag This register indicates the status for GPIO raw interrupts. A GPIO interrupt is sent to the interrupt controller if the corresponding bit in gpio_pc_ie register is enabled. 0: No interrupt 1: Interrupt occurred
GPIOD_MODE: Pad mode register					
300h	31:10	-	-	-	Reserved
	9:0	R/W	3ffh	gpio_pd_mode	PD mode register for port 0~4 These bits are written by software to configure the I/O mode. [2y+1:2y]: mode configuration for port x. 2'b00: Input mode 2'b01: General purpose output mode 2'b10: Alternate function mode 2'b11: Analog mode
Reserved					

Index	Bit	R/W	Default	Name	Description
304h	31:0	-	-	-	Reserved
GPIOD_PUPD: Pad pull-up/pull-down register					
308h	31:10	-	-	-	Reserved
	9:0	R/W	0	gpio_pd_pupd	PD pull-up/pull-down resistor These bits are written by software to configure the I/O pull-up or pull-down [2y+1:2y]: pull-up/pull-down for port x. 2'b00: No pull-up, pull-down 2'b01: Pull-up 2'b10: Pull-down (PD1 no pull-down function) 2'b11: Reserved
Reserved					
30ch	31:0	-	-	-	Reserved
GPIOD_DO: Pad output data					
310h	31:5	-	-	-	Reserved
	4:0	R/W	0	gpio_pd_out	PD output data
GPIOD_DI: Pad input data					
314h	31:5	-	-	-	Reserved
	4:0	R	-	gpio_pd_in	PD input data
GPIOD_OTYPE: Pad output type					
318h	31:5	-	-	-	Reserved
	4:0	R/W	0	gpio_pd_ot	PD output type 0: Push-pull 1: Open-drain Note: In open-drain type, if gpio_pd_pupd is enabled as pull-up, logic 1 is driven on pin by internal pull-up resistor; otherwise, logic 1 should be driven by external pull-up resistor.
GPIOD_DS: Pad driving strength					
31ch	15:0	W	-	gpio_pd_ds	PD driving strength 0: low 1: high
GPIOD_BT_RST: Pad bit reset					
320h	31:5	-	-	-	Reserved
	4:0	W	-	gpio_pd_br	PD bit reset 0: No effect 1: Reset correspond bit of gpio_pd_out as "0"
GPIOD_BT_SET: Pad bit set					
324h	31:5	-	-	-	Reserved
	4:0	W	-	gpio_pd_bs	PD bit set 0: No effect 1: Set correspond bit of gpio_pd_out as "1"
GPIOD_BT_TGLE: Pad bit toggle					
328h	31:5	-	-	-	Reserved
	4:0	W	-	gpio_pd_bt	PD bit toggle 0: No effect 1: Toggle correspond bit of gpio_pd_out "0->1" or "1->0"
GPIOD_AF0: Pad alt function					
340h	31:20				
	19:0	R/W	0	gpio_pd_af0	PD alternate function for port 0~4

Index	Bit	R/W	Default	Name	Description
					These bits are written by software to configure alternate function I/Os [4y+3:4y]: mode configuration for port x. 0: Alternate function 0 (AF0) 1: Alternate function 1 (AF1) 2: Alternate function 2 (AF2) 3: Alternate function 3 (AF3) 4: Alternate function 4 (AF4) 5: Alternate function 5 (AF5) 6~15: Reserved
Reserved					
344h	31:0	-	-	-	Reserved
Reserved					
348h	31:0	-	-	-	Reserved
Reserved					
34ch	31:0	-	-	-	Reserved
GPIOD_IE: Pad interrupt enable					
360h	31:5	-	-	-	Reserved
	4:0	R/W	0	gpio_pd_ie	PD interrupt function 0: Disable 1: Enable
GPIOD_ISS: Pad interrupt sense selection					
364h	31:5	-	-	-	Reserved
	4:0	R/W	0	gpio_pd_iss	PD interrupt sense select 0: Edge sensitive 1: Level sensitive
GPIOD_BET: Pad interrupt both edges trigger					
368h	31:5	-	-	-	Reserved
	4:0	R/W	0	gpio_pd_bet	PD interrupt both edges trigger 0: Interrupt condition is controlled by gpio_pd_trg 1: Enable both rising and falling edges trigger Note: this register will be ignored when level sensitive.
GPIOD_TRG: Pad interrupt trigger event select					
36ch	31:5	-	-	-	Reserved
	4:0	R/W	0	gpio_pd_trg	PD trigger event select 0: Falling edge or low level trigger 1: Rising edge or high level trigger
GPIOD_IF: Pad interrupt raw flag					
370h	31:5	-	-	-	Reserved
	4:0	R/W1c	0	gpio_pd_if	PD interrupt raw flag This register indicates the status for GPIO raw interrupts. A GPIO interrupt is sent to the interrupt controller if the corresponding bit in gpio_pd_ie register is enabled. 0: No interrupt 1: Interrupt occurred

R/W1C: read & write one clear

8.4 Functional Description

Subject to the specific hardware characteristics of each I/O port, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function with pull-up or pull-down capability

Each I/O port bit is freely programmable; however the I/O port registers have to be accessed as 32-bit words. The purpose of the `gpio_px_br`, `gpio_px_bs` and `gpio_px_bt` registers is to allow atomic read/modify accesses to any of the `gpio_px_out` registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 278 show the basic structures of a standard I/O port bit, respectively gives the possible port bit configurations.

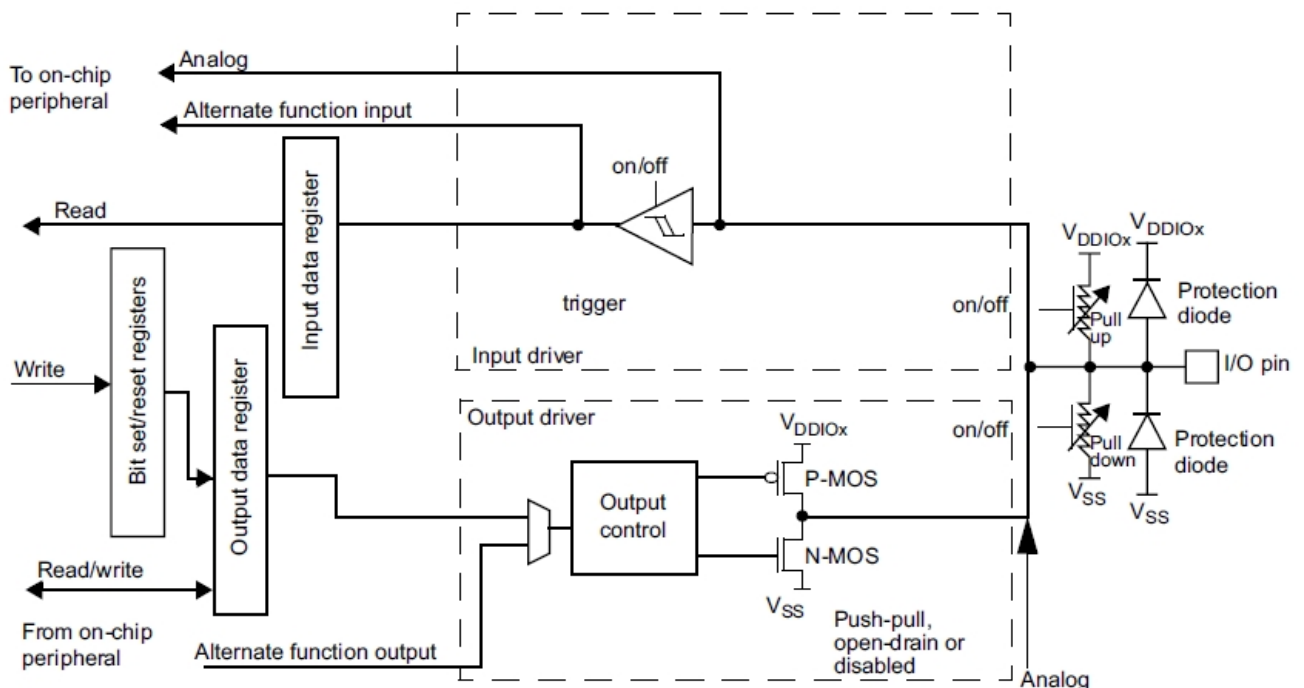


Figure 28 Basic structure of an I/O port bit

8.4.1 General-purpose I/O

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA14: SWCLK in pull-down
- PA13: SWDIO in pull-up

When the pin is configured as output, the value written to the output data register (gpio_px_out) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z). The input data register (gpio_px_in) captures the data present on the I/O pin at every AHB clock cycle.

8.4.2 General-purpose I/O

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to sixteen alternate function inputs (AF0 to AF15) that can be configured through the gpio_px_af0 and gpio_px_af1 registers:

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through gpio_px_mode register.
- The specific alternate function assignments for each pin are detailed in the Chapter 2.3 “Alternate Function I/O Priority”.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user has to proceed as follows:

- Debug function: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host
- GPIO: configure the desired I/O as output, input or analog in the gpio_px_mode register.
- Peripheral alternate function:
 - Connect the I/O to the desired AFx in one of the gpio_px_af0 or gpio_px_af1 register.
 - Select the type, pull-up/pull-down and output speed via the gpio_px_ot and gpio_px_pupd registers, respectively.
 - Configure the desired I/O as an alternate function in the gpio_px_mode register.

8.4.3 I/O port control registers

Each of the GPIO ports has three memory-mapped control registers (gpio_px_mode, gpio_px_ot, gpio_px_pupd) to configure I/Os. The gpio_px_mode register is used to select the I/O mode (input, output, AF, analog). The gpio_px_ot registers are used to select the output type (push-pull or open-drain). The gpio_px_pupd register is used to select the pull-up/pull-down whatever the I/O direction.

8.4.4 I/O port data registers

Each GPIO has two memory-mapped data registers: input and output data registers (gpio_px_in and gpio_px_out). The gpio_px_out stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (gpio_px_in), a read-only register.

8.4.5 I/O data bitwise handling

The bit reset/set/toggle register (gpio_px_br, gpio_px_bs, gpio_px_bt) are registers which allows the application to reset/set/toggle each individual bit in the output data register (gpio_px_out).

To each bit in gpio_px_out[i], correspondent bit in gpio_px_br[i], gpio_px_bs[i] and gpio_px_bt[i]. When written to 1, bit gpio_px_br[i] **resets** the corresponding gpio_px_out[i] bit. When written to 1, bit gpio_px_bs[i] **sets** the gpio_px_out[i] corresponding bit. When written to 1, bit gpio_px_bt[i] **toggle** the gpio_px_out[i] corresponding bit. By the way, writing any bit to 0 in gpio_px_br/gpio_px_bs/gpio_px_bt does not have any effect on the corresponding bit in gpio_px_out.

Using the gpio_px_br/gpio_px_bs/gpio_px_bt registers to change the values of individual bits in gpio_px_out is a “one-shot” effect that does not lock the gpio_px_out bits. The gpio_px_out bits can always be accessed directly. The gpio_px_out register provides a way of performing atomic bitwise handling.

8.4.6 I/O alternate function input/output

Alternate function registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the gpio_px_af0 and gpio_pa_af1 alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the Chapter 2.3 “Alternate Function I/O Priority”.

8.4.7 Input configuration

When the I/O port is programmed as input:

- The output buffer is enabled
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the gpio_px_pupd register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register provides the I/O state

Figure 29 shows the input configuration of the I/O port bit.

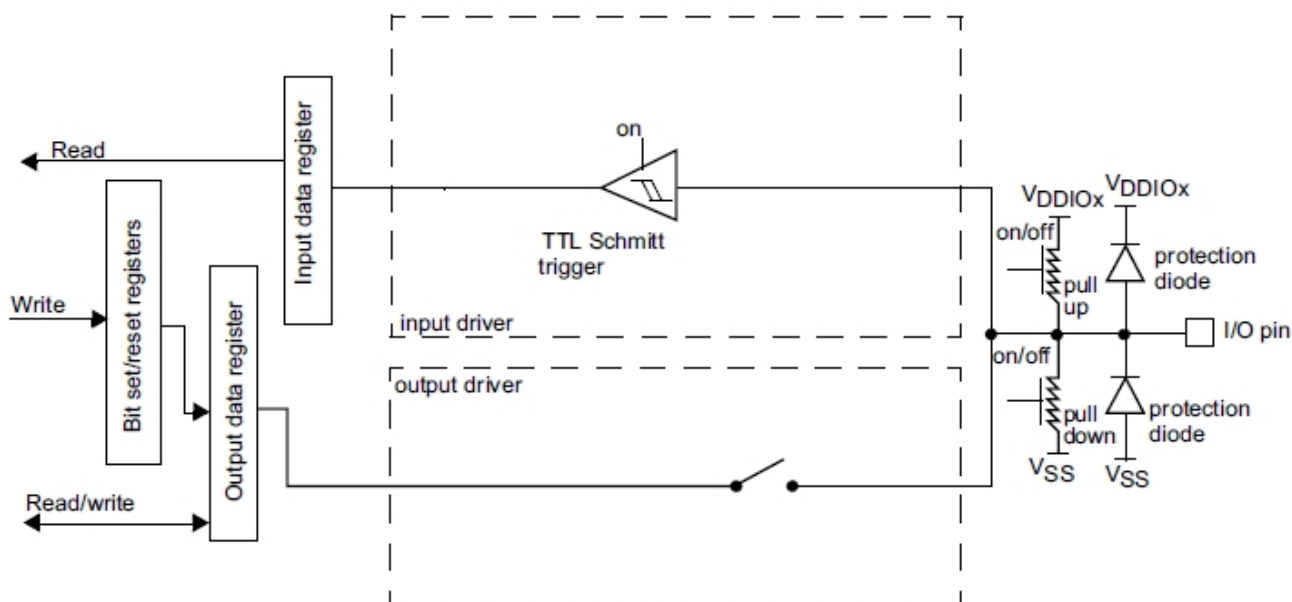


Figure 29 Input configuration

8.4.8 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 whereas 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 whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the gpio_px_pupd register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

Figure 30 shows the output configuration of the I/O port bit.

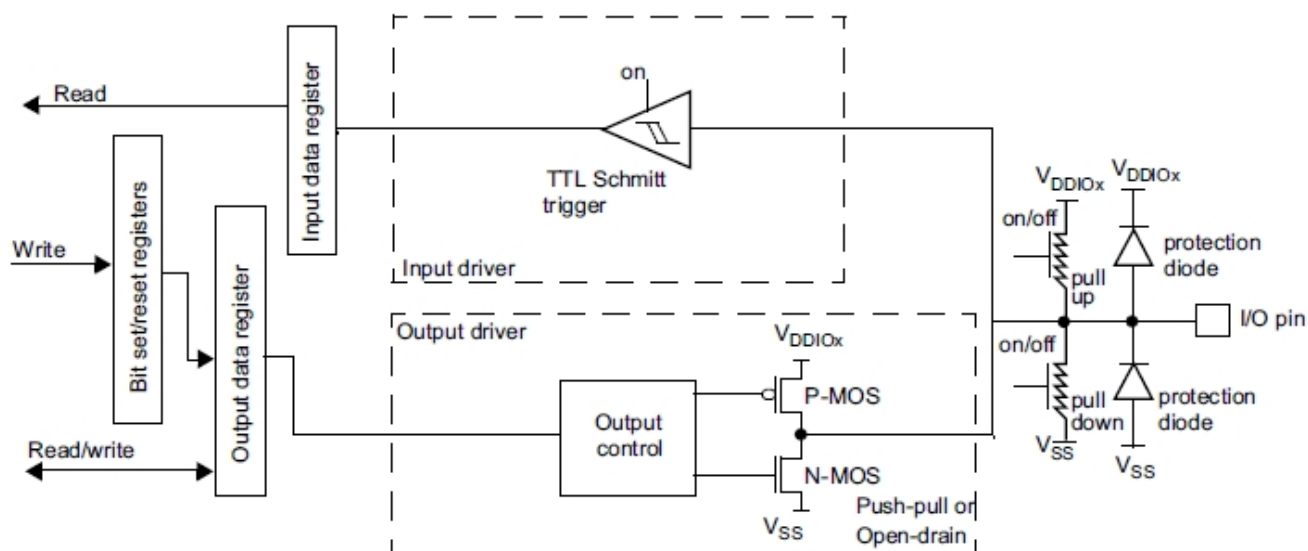


Figure 30 Output configuration

8.4.9 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are activated or not depending on the value in the gpio_px_pupd register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state

Figure 31 shows the Alternate function configuration of the I/O port bit.

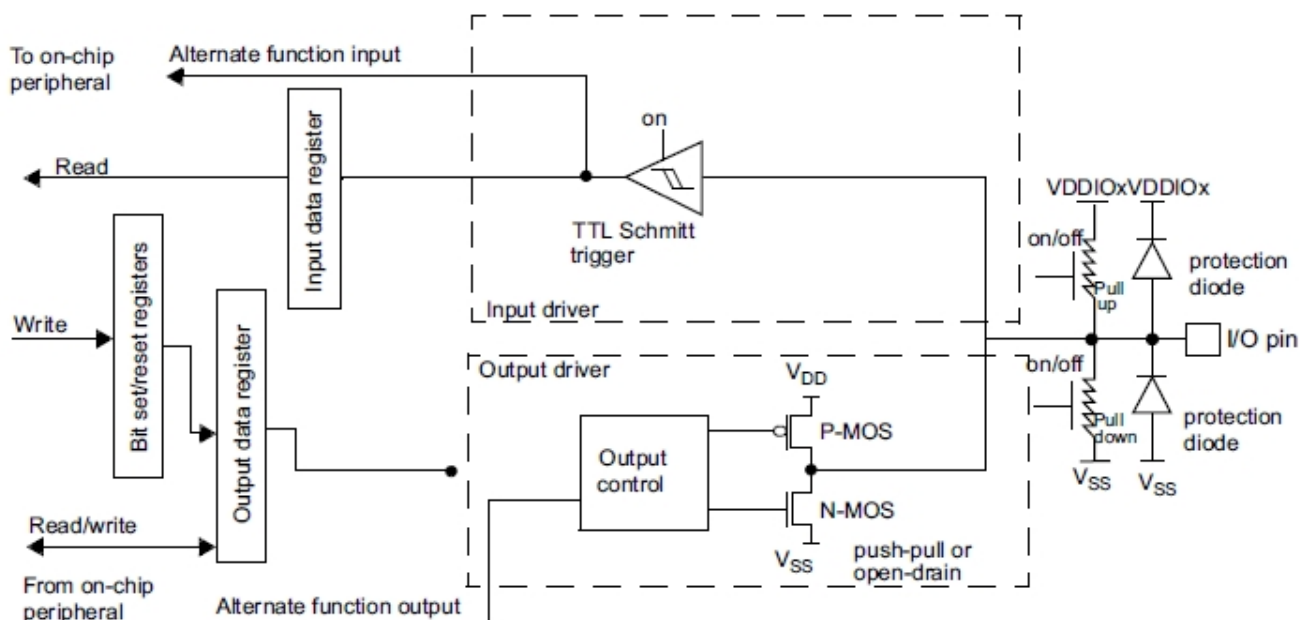


Figure 31 Alternate function configuration

8.4.10 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 by hardware
- Read access to the input data register gets the value "0"

Figure 32 shows the high-impedance, analog-input configuration of the I/O port bit.

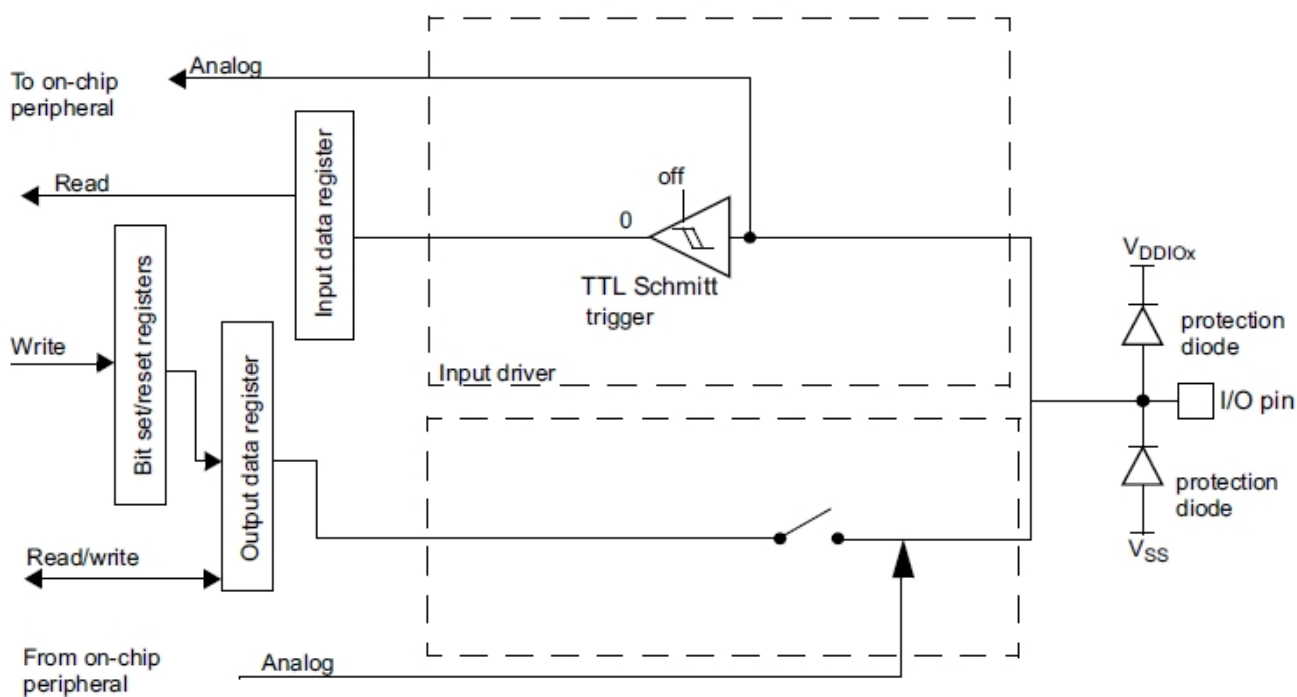


Figure 32 Analog configuration

8.4.11 Using the HXTAL or LXTAL oscillator pins as GPIOs

- ✓ When the HXTAL or LXTAL oscillator is switched OFF, the related oscillator pins can be used as normal GPIOs.
- ✓ When the HXTAL or LXTAL oscillator is switched ON, the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

9 USB

9.1 Main Features

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB bus. USB suspend/resume are supported which allows to stop the device clocks for low-power consumption.

- Compatible with USB 2.0 Full Speed Operation
- Compatible with USB audio device class spec V1.0
- 1 Control Endpoint (Endpoint 0), IN/OUT each with 64B (8/16/32/64B programmable) FIFO
- 7 Generic Endpoints (IN/OUT, INT/Bulk programmable); Endpoints 1~6 and EP9.
- 2 Isochronous Endpoints (Endpoint 7, 8) with DMA channel between SRAM and USB FIFO
- Total FIFO for Endpoints 0~9: 1024B+576B
 - Endpoints 7& 8: share 1024B (supports DMA)
 - Endpoint 0: In 64B, out 64B
 - Endpoints 1~ 6, 9: 64B/each
- Support USB Suspend, Resume and Remote-Wakeup
- USB function could be disabled for non-USB application

9.2 Block Diagram

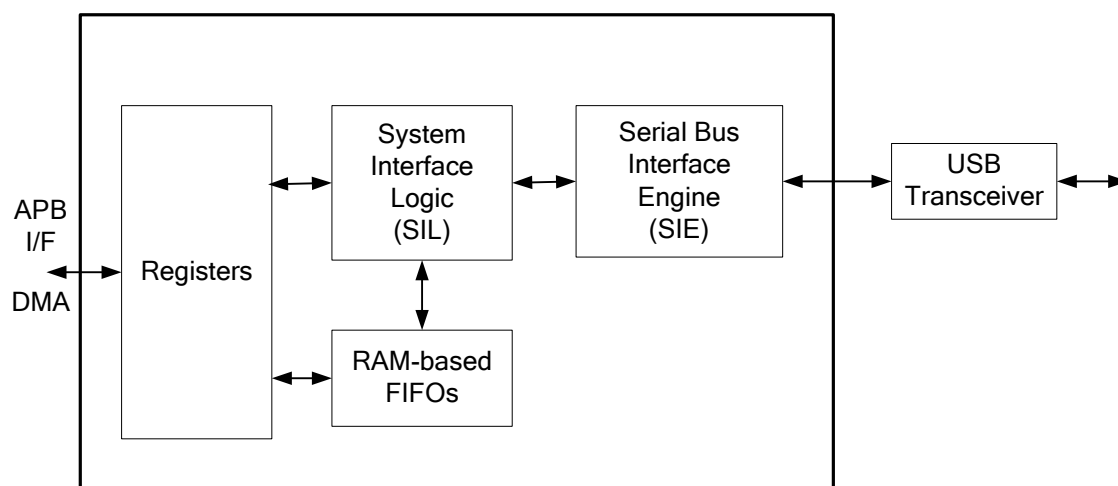


Figure 33 USB Block Diagram

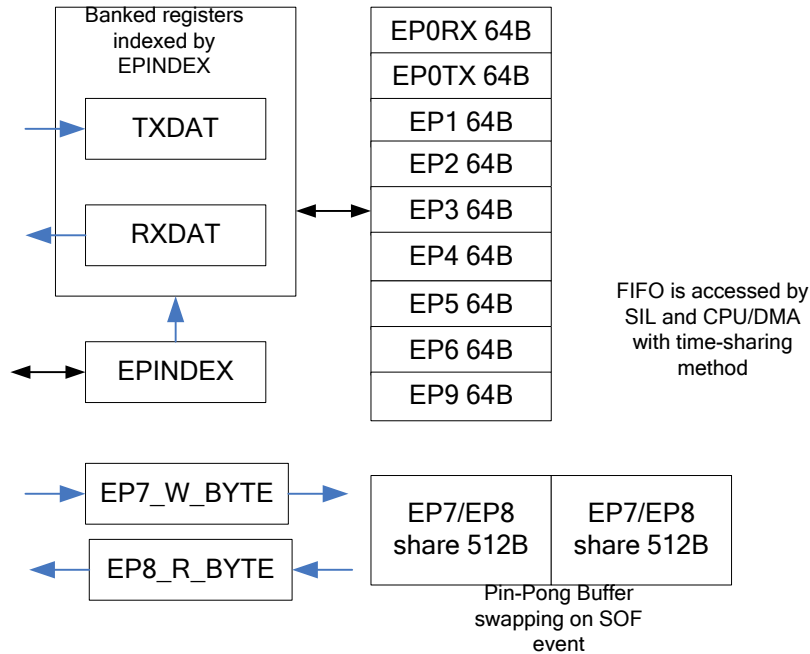


Figure 34 USB Endpoints FIFO

9.3 Register Definition

Table 20 USB Control Register

Index	Bit	R/W	Default	Name	Description
USB_FADDR: USB Function Address					
00h	31:7	-	-	-	Reserved
	6:0	R/W	0	fa	7-bit programmable functional address
USB_USBFIF: USB Function Interrupt					
04h	31:11	-	-	-	Reserved
	10	R	0	usbx9int	Function Tx/Rx Done Flag for endpoint 9
	9:8	-	-	-	Reserved
	7	R	0	usbx6int	Function Tx/Rx Done Flag for endpoint 6
	6	R	0	usbx5int	Function Tx/Rx Done Flag for endpoint 5
	5	R	0	usbx4int	Function Tx/Rx Done Flag for endpoint 4
	4	R	0	usbx3int	Function Tx/Rx Done Flag for endpoint 3
	3	R	0	usbx2int	Function Tx/Rx Done Flag for endpoint 2
	2	R	0	usbx1int	Function Tx/Rx Done Flag for endpoint 1
	1	R	0	usbrx0int	Function Receive Done Flag for endpoint 0
	0	R	0	usbtx0int	Function Transmit Done Flag for endpoint 0
USB_USBFIE: USB Function Interrupt Enable					
08h	31:11	-	-	-	Reserved
	10	R/W	0	usbx9int_ie	Function Tx/Rx Done Interrupt Enable for EP9
	9:8	-	-	-	Reserved
	7	R/W	0	usbx6int_ie	Function Tx/Rx Done Interrupt Enable for EP6

Index	Bit	R/W	Default	Name	Description
	6	R/W	0	usbx5int_ie	Function Tx/Rx Done Interrupt Enable for EP5
	5	R/W	0	usbx4int_ie	Function Tx/Rx Done Interrupt Enable for EP4
	4	R/W	0	usbx3int_ie	Function Tx/Rx Done Interrupt Enable for EP3
	3	R/W	0	usbx2int_ie	Function Tx/Rx Done Interrupt Enable for EP2
	2	R/W	0	usbx1int_ie	Function Tx/Rx Done Interrupt Enable for EP1
	1	R/W	0	usbrx0int_ie	Function Receive Done Interrupt Enable for EP0
	0	R/W	0	usbtx0int_ie	Function Transmit Done Interrupt Enable for EP0
USB_SIEI: SIE Interface					
0ch	31:6	-	-	-	Reserved
	5	R/W	0	usben	Enable USB transceiver function
	4	R/W	0	dm	DM Pull-up Enable
	3	R/W	0	psofen	Enable Pseudo-SOF (PSOF) generator
	2	-	-	-	Reserved
	1	R/W	0	dp	DP Pull-up Enable
	0	W	0	wakeup	This bit is used by the USB function to initiate a remote wakeup. Set by firmware to drive resume signaling on the USB lines to the host or upstream hub. Cleared by hardware when resume signaling is done
USB_EPINDEX: Endpoint Index					
14h	31:4	-	-	-	Reserved
	3:0	R/W	0	epinx	The value in this register selects the associated bank of endpoint-indexed registers including USB_TXDAT, USB_TXCON, USB_TXFLG, USB_TXCNT, USB_TXSTAT, USB_RXDAT, USB_RXCON, USB_RXFLG, USB_RXCNT, USB_RXSTAT, USB_EPCON, USB_EPNUM. 0 = Endpoint 0. 1 = Endpoint 1. 2 = Endpoint 2. 3 = Endpoint 3. 4 = Endpoint 4. 5 = Endpoint 5. 6 = Endpoint 6. 7 = Endpoint 7. (only has USB_EPNUM reg) 8 = Endpoint 8. (only has USB_EPNUM reg) 9 = Endpoint 9. others = Reserved
USB_EPCON: Endpoint Control (Endpoint-indexed)					
18h	31:8	-	-	-	Reserved
	7	R/W	0	rxstl	Stall Receive Endpoint: Set this bit to stall the receive endpoint. Clear this bit only when the host has intervened through commands sent down endpoint 0. When this bit is set and RXSETUP is clear, the receive endpoint will respond with a STALL handshake to a valid OUT token. When this bit is set and RXSETUP is set, the receive endpoint will NAK. This bit does not affect the reception of SETUP token by a control

Index	Bit	R/W	Default	Name	Description
					endpoint.
	6	R/W		txstl	Stall Transmit Endpoint: Set this bit to stall the transmit endpoint. This bit should be cleared only when the host has intervened through commands sent down endpoint 0. When this bit is set and RXSETUP is clear, the receive endpoint will respond with a STALL handshake to a valid IN token. When this bit is set and RXSETUP is set, the receive endpoint will NAK.
	5	R/W	0/1	ctlep	Control Endpoint: Set this bit to configure the endpoint as a control endpoint. Only control endpoint is capable of receiving SETUP tokens. (only EP0 is control endpoint)
	4	-	-	-	Reserved
	3	R/W	0	rxie	Receive Input Enable: Set this bit to enable data from the USB to be written into the receive FIFO. If cleared, the endpoint will not write the received data into the receive FIFO and at the end of reception, but will return a NAK handshake on a valid OUT token if the RXSTL bit is not set. This bit does not affect a valid SETUP token. A valid SETUP token and packet overrides this bit if it is cleared, and place the receive data in the FIFO.
	2	R/W	0/1	rxepen	Receive Endpoint Enable: Set this bit to enable the receive endpoint. When disabled, the endpoint does not respond to valid OUT or SETUP token. This bit is hardware read-only and has the highest priority among RXIE and RXSTL. Note that endpoint 0 is enabled for reception upon reset.
	1	R/W	0	txoe	Transmit Output Enable: This bit is used to enable the data in TXDAT to be transmitted. If cleared, the endpoint returns a NAK handshake to a valid IN token if the TXSTL bit is not set.
	0	R/W	0/1	txepen	Transmit Endpoint Enable: This bit is used to enable the transmit endpoint. When disabled, the endpoint does not response to a valid IN token. This bit is hardware read-only. Note that endpoint 0 is enabled for transmission upon reset.
USB_TXDAT: Transmit FIFO Data (Endpoint-indexed)					
20h	31:8	-	-	-	Reserved
	7:0	W	-	txdat	Transmit Data Byte (write-only): To write data to the transmit FIFO, write to this register. The write pointer is incremented automatically after a write.
USB_TXCON: Transmit FIFO Control (Endpoint-indexed)					
24h	31:8	-	-	-	Reserved

Index	Bit	R/W	Default	Name	Description
	7	W	0	txclr	Transmit Clear: Setting this bit flushes the transmit FIFO, resets all the read/write pointers, sets the EMPTY bit in TXFLG, and clears all other bits in TXFLG. After the flush, hardware clears this bit.
	6:0	-	-	-	Reserved
USB_TXFLG: Transmit FIFO Flag (Endpoint-indexed)					
28h	31:4	-	-	-	Reserved
	3	R	1	txemp	Transmit FIFO Empty Flag (read-only): Hardware sets this bit when the data set has been read out of the transmit FIFO by SIL. Hardware clears this bit when the empty condition no longer exists. This bit always tracks the current transmit FIFO status. This flag is also set when a zero-length data packet is transmitted.
	2	R	0	txfull	Transmit FIFO Full Flag (read-only): This flag indicates the data set is present in the transmit FIFO. This bit is set after write to TXCNT to reflect the condition of the data set. Hardware clears this bit when the data set has been successfully transmitted.
	1	R/W0 c	0	txurf	Transmit FIFO Under-run Flag (read-, clear-only)*: Hardware sets this flag when an addition byte is read from an empty transmit FIFO. This is a sticky bit that must be cleared through firmware by writing a „0“ to this bit. When the transmit FIFO under-runs, the read pointer will not advance – it remains locked in the empty position. Note: only EP0 has this flag.
	0	R/W0 c	0	txovf	Transmit FIFO Overrun Flag (read-, clear-only)*: This bit is set when an additional byte is written to a FIFO with TXFULL = 1. This is a sticky bit that must be cleared through firmware by writing a „0“ to this bit. When the transmit FIFO overruns, the write pointer will not advance – it remains locked in the full position. Note: only EP0 has this flag.
USB_TXCNT: Transmit FIFO Byte Count (Endpoint-indexed)					
2ch	31:7	-	-	-	Reserved
	6:0	W	0	txcnt	Transmit Byte Count (write-only): The number of bytes in the data set written to the transmit FIFO. When this register is written, TXFULL is set. Write the byte count to this register after writing data set to USB_TXDAT.
USB_TXSTAT: Endpoint Transmit Status (Endpoint-indexed)					
30h	31:8	-	-	-	Reserved
	7	R/W0 c	0	txseq	Transmit Current Sequence Bit (read, clear-only):

Index	Bit	R/W	Default	Name	Description
					This bit will be transmitted in the next PID and toggled on a valid ACK handshake. This bit is toggled by hardware on a valid SETUP token. The SIE will handle all sequence bit tracking. This bit should only be used when initializing a new configuration or interface.
	6:3	-	-	-	Reserved
	2	R	0	txvoid	Transmit Void (read-only): A void condition has occurred in response to a valid IN token. Transmit void is closely associated with the NAK/STALL handshake returned by the function after a valid IN token, due to the conditions that cause the transmit FIFO to be unable or not ready to transmit. Use this bit to check any NAK/STALL handshake returned by the function. This bit does not affect the USBTxxINT, TXERR or TXACK bit. This bit is updated by hardware at the end of a non-isochronous transaction in response to a valid IN token.
	1	R	0	txerr	Transmit Error (read-only): An error condition has occurred with the transmission. Complete or partial data has been transmitted. The error can be one of the following: 1. Data transmitted successfully but no handshake received. 2. Transmit FIFO goes into under-run condition while transmitting. The corresponding transmit done bit is set when active. This bit is updated by hardware along with the TXACK bit at the end of data transmission (this bit is mutually exclusive with TXACK).
	0	R	0	txack	Transmit Acknowledge (read-only): Data transmission completed and acknowledged successfully. The corresponding transmit done bit is set when active. This bit is updated by hardware along with the TXERR bit at the end of data transmission (this bit is mutually exclusive with TXERR).
USB_EPNUM: Endpoint Number (Endpoint-indexed)					
34h	31:4	-	-	-	Reserved
	3:0	R/W		epnm	Upon USB reset, default values are 1~9 for Endpoint 1~9. User can change the value to change the mapped endpoint number which is presented to USB host.
USB_DBCON: Double Buffer Control (Endpoint-indexed, for EP1~6 and 9)					
38h	31:8	-	-	-	Reserved
	7	R/W	0	dben	Double buffer enable (for EP1~6 and 9)
	6:5	-	-	-	Reserved
	4:0	R/W		dbbank	Double buffer bank number. Each bank

Index	Bit	R/W	Default	Name	Description
					occupies 64 bytes. Use this register to assign the address of the second buffer.
USB_NOTEBOOK: General purpose 8-bit register					
3ch	31:8	-	-	-	Reserved
	7:0	R/W	0	notebook	General purpose 8-bit register
USB_USBFI_CLR: USB Function Interrupt Clear					
40h	31:11	-	-	-	Reserved
	10:0	W			User should write "1" to register USB_USBFI_CLR to clear corresponding USBFI interrupt register bit. This register is write-only and auto-clear.
USB_USBFI2: USB Function Interrupt Register 2					
44h	31:8	-	-	-	Reserved
	7	R/W0 c	0	usbint	USB reset interrupt
	6	R/W0 c	0	resume	USB SIE has detected a RESUME signaling on the USB lines. This interrupt is used to terminate the power-down mode.
	5	R/W0 c	0	suspend	USB SIE has detected a SUSPEND signaling on the USB lines. The corresponding ISR should put the whole chip into power-down mode.
	4	-	-	-	Reserved
	3	R/W0 c	0	sofint	SOF/PSOF interrupt
	2:0	-	-	-	Reserved
USB_USBFI2: USB Function Interrupt Enable Register 2					
48h	31:8	-	-	-	Reserved
	7	R/W	0	nakint_ie	NAK Interrupt Enable: Set this bit to enable USB interrupt for hub and function even if NAK or STALL handshake is returned.
	6	R/W	0	resume_ie	RESUME Interrupt Enable.
	5	R/W	0	suspend_ie	SUSPEND Interrupt Enable.
	4	R/W	0	stallint_ie	STALL Interrupt Enable: Set this bit to enable USB interrupt for hub and function if STALL handshake is returned.
	3	R/W	0	tgerr_ie	RX-TOGGLE-ERROR Interrupt Enable: Set this bit to enable USB interrupt for hub and function if data toggle mismatch is found (RX_ERR/RXACK is cleared)
	2	R/W	0	sof_ie	SOF/PSOF interrupt Enable
	1	R/W	0	usbrt_ie	USB reset interrupt Enable
	0	R/W	0	resume_ie2	RESUME Interrupt Enable method 2. Wakeup MCU when DP=0.
USB_DBSTAT: Double Buffer Status (Endpoint-indexed, for EP1~6 and 9)					
54h	31:8	-	-	-	Reserved
	7	R	0	dfull_1	The second buffer full status
	6	R	0	dfull_0	The first buffer full status
	5	R	0	sil_bi	The buffer will be used by SIL. 0: first buffer, 1: second buffer

Index	Bit	R/W	Default	Name	Description
	4	R	0	sil_cpu	The buffer will be used by CPU. 0: first buffer, 1: second buffer
	3	-	-	-	Reserved
	2	R	0	rtfull_cpu	The buffer full status for CPU. (== dfull[sil_cpu])
	1:0	-	-	-	Reserved
USB_RXDAT: Receive FIFO Data (Endpoint-indexed)					
60h	31:8	-	-	-	Reserved
	7:0	R	-	rxdat	Receive Data Byte (read-only): To write data to the receive FIFO, the SIL writes to this register. To read data from the receive FIFO, the CPU reads from this register. The write pointer and read pointer are incremented automatically after a write and read, respectively.
USB_RXCON: Receive FIFO Control (Endpoint-indexed)					
64h	31:8	-	-	-	Reserved
	7	W	0	rxclr	Clear the Receive FIFO: Set this bit to flush the entire receive FIFO. All flags in RXFLG revert to their reset states (RXEMP is set; all other flags clear). Hardware clears this bit when the flush operation is complete.
	6:5	-	-	-	Reserved
	4	W	0	rxffrc	FIFO Read Complete: Set this bit to release the receive FIFO when a data set read is complete. Setting this bit clears the RXFULL bit (in the USB_RXFLG register) corresponding to the data set that was just read. Hardware clears this bit after the RXFULL bit is cleared. All data from this data set must have been read. Note that FIFO Read Complete only works if STOVW and EDOVW are cleared.
	3:0	-	-	-	Reserved
USB_RXFLG: Receive FIFO Flag (Endpoint-indexed)					
68h	31:8	-	-	-	Reserved
	7	R/W0 c	0	sendnak	NAK flag (read-, clear-only): This flag indicates a NAK is returned to host while NAKINT_IE is set.
	6	R/W0 c	0	sendstall	STALL flag (read-, clear-only): This flag indicates a STALL is returned to host while STALLINT_IE is set.
	5	R/W0 c	0	tgerr	RX-TGERR flag (read-, clear-only): This flag indicates data toggle error is found while TGERR_IE is set.
	4	-	-	-	Reserved
	3	R	1	rxemp	Receive FIFO Empty Flag (read-only): Hardware sets this bit when the data set has been read out of the receive FIFO. Hardware clears this bit when the empty condition no longer exists. This is not a sticky bit and always tracks the current status. This flag is also set when a zero-length packet is received.

Index	Bit	R/W	Default	Name	Description
	2	R	0	rxfull	Receive FIFO Full Flag (read-only): This flag indicates the data set is present in the receive FIFO. Hardware sets this bit when the data set has been successfully received. This bit is cleared after write to RXCNT to reflect the condition of the data set. Likewise, this bit is cleared after setting of the RXFFRC bit.
	1	R/W0 c	0	rxurf	Receive FIFO Under-run Flag (read-, clear-only)*: Hardware sets this bit when an additional byte is read from an empty receive FIFO. This bit is cleared through firmware by writing a „0“ to this bit. When the receive FIFO under-runs, the read pointer will not advance -- it remains locked in the empty position. When set, all transmissions are NAKed. Note: only EP0 has this flag.
	0	R/W0 c	0	rxovf	Receive FIFO Overrun Flag (read-, clear-only)*: This bit is set when the SIL writes an additional byte to a receive FIFO with RXFULL = 1. This is a sticky bit that must be cleared through firmware by writing a „0“ to this bit, although it can be cleared by hardware if a SETUP packet is received after an RXOVF error had already occurred. When the receive FIFO overruns, the write pointer will not advance -- it remains locked in the full position. When set, all transmissions are NAKed. Note: only EP0 has this flag.
USB_RXCNT: Receive FIFO Byte Count (Endpoint-indexed)					
6ch	31:7	-	-	-	Reserved
	6:0	R	0	rxcnt	Byte Count (read-only): The number of bytes in data set written to the receive FIFO. After the SIL writes a data set to the RXFIFO, it writes the byte count to this register. The CPU reads the byte count from this register to determine how many bytes to read from the RXFIFO.
USB_RXSTAT: Endpoint Receive Status (Endpoint-indexed)					
70h	31:8	-	-	-	Reserved
	7	R/W0 c	0	rxseq	Receive Endpoint Sequence Bit (read-, clear-only): This bit will be toggled on completion of an ACK handshake in response to an OUT token. This bit will be set (or created) by hardware after reception of SETUP token. The SIE will handle all sequence bit tracking. This bit should only be used when initializing a new configuration or interface. If you don't want to change sequence bit, set this bit to "1" when you write this register.
	6	R/W0 c	0	rxsetup	Receive Setup Token (read-, clear-only): This bit is set by hardware when a valid SETUP token has been received. When SIL set this bit,

Index	Bit	R/W	Default	Name	Description
					it causes received IN or OUT token to be NAKed until the bit is cleared to allow a control transaction. IN or OUT token is NAKed even if the endpoint is stalled (RXSTL or TXSTL) to allow a control transaction to clear a stalled endpoint. Clear this bit upon detection of a SETUP token after the firmware is ready to complete the setup stage of control transaction.
	5	R	0	stovw	Start Overwrite Flag (read-only): Set by hardware upon receipt of SETUP token for any control endpoint to indicate that the receive FIFO is being overwritten with new SETUP data. When set, the FIFO state (RXFULL and read pointer) resets and is locked for this endpoint until EDOVW is set. This prevents a prior, ongoing firmware read from corrupting the read pointer as the receive FIFO is being cleared and new data is being written into it. This bit is cleared by hardware at the end of handshake phase transmission of the setup stage. This bit is used only for control endpoint.
	4	R/W0 c	0	edovw	End Overwrite Flag (read-, clear-only): This flag is set by hardware during the handshake phase of a SETUP stage. It is set after every SETUP packet is received and must be cleared prior to reading the contents of the FIFO. When set, the FIFO state (RXFULL and read pointer) remains locked for this endpoint until this bit is cleared. This prevents a prior, ongoing firmware read from corrupting the read pointer after the new data has been written into the receive FIFO. This bit is only used for control endpoint. Note: Make sure the EDOVW bit is cleared prior to reading the contents of the receive FIFO.
	3	-	-	-	Reserved
	2	R	0	rxvoid	Receive Void Condition (read-only): This bit is set when no valid data is received in response to a SETUP or OUT token due to one of the following conditions: 1. The receive FIFO is still locked. 2. The USB_EPCON register's RXSTL bit is set. This bit is set and cleared by hardware. This bit is updated by hardware at the end of the transaction in response to a valid OUT token.
	1	R	0	rxerr	Receive Error (read-only): Set when an error condition has occurred with the reception. Complete or partial data has been written into the receive FIFO. No handshake is returned. The error can be one of the following conditions:

Index	Bit	R/W	Default	Name	Description
					1. Data failed CRC check. 2. Bit stuffing error. 3. A receive FIFO goes into overrun or underrun condition while receiving. This bit is updated by hardware at the end of a valid SETUP or OUT token transaction. The corresponding receive done bit is set when active. This bit is updated with the RXACK bit at the end of data reception and is mutually exclusive with RXACK.
	0	R	0	rxack	Receive Acknowledged (read-only): This bit is set when data is received completely into a receive FIFO and an ACK handshake is sent. This read-only bit is updated by hardware at the end of valid SETUP or OUT token transaction. The corresponding receive done bit set when active. This bit is updated with the RXERR bit at the end of data reception and is mutually exclusive with RXERR.
USB_EP7_CON: EP7 Control					
80h	31:2	-	-	-	Reserved
	1	R/W	0	ep7_dma_en	1: enable endpoint 7 DMA transfer
	0	R/W	0	ep7_en	0: disable endpoint 7 function 1: enable endpoint 7 function
USB_EP7_STAT: EP7 Status					
84h	31:13	-	-	-	Reserved
	12	R	0	ep7_tx_full	EP7 TX FIFO is full
	11:10	-	-	-	Reserved
	9:0	R	0	ep7_tx_len	EP7 TX FIFO data length
USB_EP7_BASE: EP7 Base					
88h	31:9	-	-	-	Reserved
	8:0	R/W	0	ep7_base	EP7 TX FIFO base pointer
USB_EP7_LIMIT: EP7 Limit					
8ch	31:10	-	-	-	Reserved
	9:0	R/W	100h	ep7_limit	EP7 TX FIFO buffer size
USB_EP7_W_BYTE: EP7 Write Data Byte					
90h	31:8	-	-	-	Reserved
	7:0	W	-	ep7_w_byte	EP7 TX FIFO write data
USB_EP8_CON: EP8 Control					
a0h	31:2	-	-	-	Reserved
	1	R/W	0	ep8_dma_en	1: enable endpoint 8 DMA transfer
	0	R/W	0	ep8_en	0: disable endpoint 8 function 1: enable endpoint 8 function
USB_EP8_STAT: EP8 Status					
a4h	31:12	-	-	-	Reserved
	11	R	0	ep8_rx_empty	EP8 RX FIFO is empty
	10	R	-	ep8_rx_err	EP8 RX packet error
	9:0	R	0	ep8_rx_len	EP8 RX FIFO data length
USB_EP8_BASE: EP8 Base					
a8h	31:9	-	-	-	Reserved
	8:0	R/W	100h	ep8_base	EP8 RX FIFO base pointer

Index	Bit	R/W	Default	Name	Description
USB_EP8_LIMIT: EP8 Limit					
ach	31:10	-	-	-	Reserved
	9:0	R/W	100h	ep8_limit	EP8 RX FIFO buffer size
USB_EP8_R_BYTE: EP8 Read Data Byte					
b0h	31:8	-	-	-	Reserved
	7:0	R	-	ep8_r_byte	EP8 RX FIFO read data
USB_FRAME: Frame Status					
c0h	31:13	-	-	-	Reserved
	12	R	0	frame_miss	Indicates expected SOF packet is missed (valid only when PSOF function is enabled)
	11	R	0	frame_error	Indicates last SOF packet has error
	10:0	R	0	frame	USB frame number

9.4 Function Description

9.4.1 Main blocks

The USB function interface manages communications between the Host and the USB function. The MCU interface consists of the USB full speed transceiver, the serial bus engine (SIE), the system interface logic (SIL), and the transmit/receive FIFOs. The USB transceiver in MCU provides a physical interface to USB lines. The SIE handles communication protocol of USB. The SIL handles data transfers and provides the interface among the SIE, the CPU and the function FIFOs.

The main blocks in the USB module are:

1. Full Speed USB Transceiver: This is an on-chip transceiver having one differential driver to transmit the USB data onto the USB bus and single ended receivers on the D+ and D- lines as well as a differential receiver to receive the USB data signal on the USB bus.

2. Serial Bus Interface Engine (SIE): The SIE does all the front-end functions of USB protocol such as clock/data separation, sync-field identification, NRZI-NRZ conversion, token packet decoding, bit stripping, bit stuffing, NRZ-NRZI conversion, CRC5 checking and CRC16 generation and checking. Besides, it manages detecting of reset, suspend and resume signals on the upstream port of the MCU to wakeup the system from the suspend state. It also provides serial-to-parallel conversion for the serial packet from the full speed USB transceiver to 8 bit parallel data to the system interface logic and for 8 bit parallel data from the system interface logic to serial packet to the full speed USB transceiver.

3. System Interface Logic (SIL): The SIL operates in conjunction with the CPU to provide the capabilities of controlling the operation of the FIFOs. The SIL also monitors the status of the data transactions, transfers event control to the CPU through interrupt requests at the appropriate moment, initiate resume signaling to USB bus while the MCU is in power-down mode. Operation of the SIL is controlled through the use of external function registers.

4. Device Function: The USB device function interface has ten endpoints that can support four types of USB data transfer: control, interrupt, bulk and isochronous transfer. Transmit FIFOs are written by CPU, and then read by SIL for transmission. Receive FIFO is written by the SIL following reception, and then read by the CPU. Endpoint 0 supports control transfer for configuration / command / status type communication flows between client software and function. Endpoint 1 ~ 6 and 9 supports interrupt/bulk transfer. Endpoint 7 supports isochronous IN transfer, and Endpoint 8 supports isochronous OUT transfer.

9.4.2 Function endpoints

The USB module supports 8 device function endpoints. Endpoint 0 contains two FIFOs for transmit and receive, respectively. Endpoint 1 to 6 and 9 can be programmed to transmit or receive. Endpoint 0 handles control data transfer. Endpoint 1 to 6 and 9 can be interrupt transfer or bulk transfer. The USB_EPINDEX register selects the endpoint for any given data transaction. Endpoint 7 and 8 are dedicated isochronous endpoints which have their control and data registers which are not indexed by USB_EPINDEX.

9.4.3 Transmit FIFOs

9.4.3.1 Transmit FIFOs Features

The USB transmit FIFOs are data buffers with the following features (see Figure 35)

- USB support for one data set of not greater than 8/16/32/64 bytes (programmable by counter setting).
- a byte count register to store the number of bytes in the data set
- protection against overwriting data in a full FIFO
- capable to retransmit the current data set

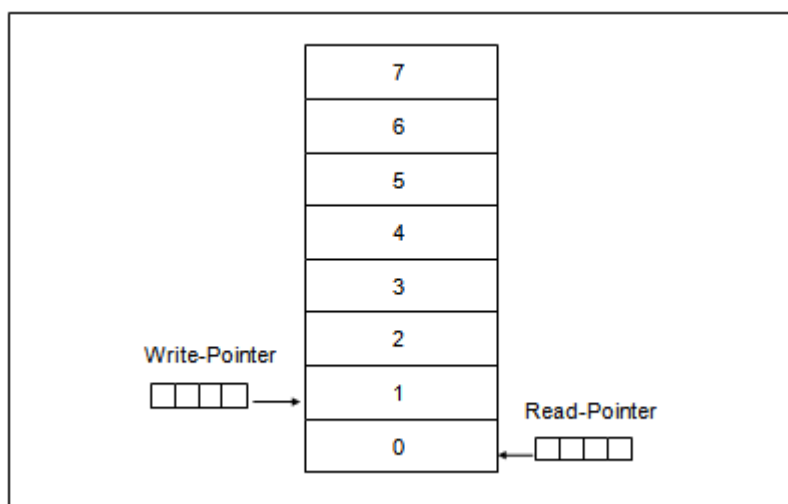


Figure 35 Transmit FIFO Buffer Outline (EX: for 8 bytes FIFO)

The CPU writes to the FIFO location specified by the write-pointer also used as the byte-counter to indicate how many bytes have been written and not yet read by the SIL. The write-pointer automatically increments by one after a write, and it decrements by one after a read. The read-pointer points the next FIFO location to be read by the USB SIL. The read-pointer automatically increments by one after a read. The transmit FIFO is inhibited to be read by the SIL when it is empty or before a data set has been successfully written into it.

9.4.3.2 Transmit Data Set Management

TXFULL = 1 in the USB_TXFLG register, indicates data set has been written into the FIFO and is ready for transmission. Following reset, TXFULL = 0 and TXEMP = 1, signifying an empty FIFO. Only the first eight bytes of the data set which size is greater than eight are written into the FIFO. In this case, TXFULL is not set until a write to TXCNT. In the case of TXFULL = 1 farther writing to TXDAT or TXCNT are ignored. Please note that the content of TXCNT determines the number of bytes transmitted over the USB lines. Discrepancy between the byte number written to TXCNT and number of bytes actually written to the FIFO will cause an unexpected result. Read the FIFO is prohibited when the FIFO is empty or TXFULL = 0.

Two events cause the TXFULL to be updated:

- A new data set is written to the FIFO: The CPU writes bytes to the FIFO via TXDAT and writes the number of bytes to TXCNT. TXFULL is only set after the write to TXCNT. Set TXCNT=0 indicates a zero length transmission. In this case, TXFULL is set and TXEMP remains unchanged to indicate the FIFO is still empty. This process is illustrated in Table 22.
- A data set in the FIFO is successfully transmitted: The SIL reads the data set from the FIFO for transmission. When a good transmission is acknowledged, the TXFULL is cleared and TXEMP is set.

Table 21 Writing to the Byte Count Register

TXFULL	TXEMP	Zero Length Transmission	Write bytes to USB_TXDATx	Data Set Written	TXFULL	TXEMP
0	0	No		Yes	1	0
0	1	No		Yes	1	0
0	1	Yes	Write byte count to	No	1	1
1	-	-	USB_TXCNTx	Write Ignored	1	-

When a good transmission is completed, both read pointer and write pointer is advanced to the start point of the FIFO to set up for transmitting the next data set. When a bad transmission is encountered, the read pointer is reversed to the start point of the FIFO to enable the SIL to re-read the last data set for retransmission. The pointer reversal and advance are accomplished automatically by hardware. Table 22 summarizes how actions following a transmission depend on TXERR and TXACK.

Table 22 Truth Table for Transmit FIFO Management

TXERR	TXACK	Action at End of Transfer Cycle
0	0	No operation
0	1	Read Pointer and Write Pointer both are set to the start point of FIFO
1	0	Read Pointer is set to the start point of FIFO

9.4.4 Receive FIFOs

9.4.4.1 Receive FIFOs Features

The receive FIFO is a data buffer with the following features (see Figure 36):

- support for one data set of not greater than 8/16/32/64 bytes
- a byte count register accesses the number of bytes in the data set
- flag to signal a full FIFO and an empty FIFO
- capability to re-receive the last data set

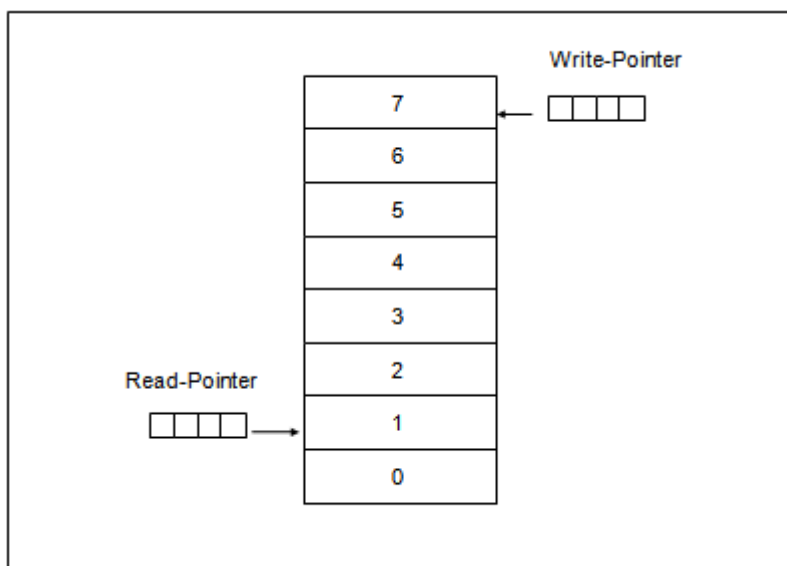


Figure 36 Receive FIFO Outline (EX: for 8 bytes FIFO)

The SIL writes to the FIFO location specified by the write pointer also used as the byte-counter to indicate how many bytes have been written and not yet read by the CPU. The write-pointer will automatically increment by one after a write and decrement by one after a read. The read-pointer points the next FIFO location to be read by the CPU. The read-pointer will automatically increment by one after a read. The

receive FIFO is inhibited to be read by the CPU when it is empty or before a data set has been successfully written into it. **When a SETUP token is detected by the SIL, the SIL flushes the FIFO even if the FIFO is being read by the CPU.**

9.4.4.2 Receive Data Set Management

RXFULL = 1 in the USB_RXFLG register, indicates the data set has been written into the FIFO and is ready for reception. Following reset, RXFULL = 0 and RXEMP = 1, signifying an empty FIFO. Only the first eight bytes of the data set which size is greater than eight are written into FIFO. RXFULL is however not set until reception is done and successfully acknowledged. RXFULL is cleared by setting the FFRC bit of RXCON in firmware to indicate the data set has successfully read by CPU. In the case of RXFULL = 1 farther writes to FIFO are ignored. Please note that the content of RXCNT should be read by CPU to determine the numbers of bytes need to be read from FIFO by CPU. Further reading from an empty FIFO is ignored.

Table 23 Status of the Receive FIFO Data Set

RXFULL	RXEMP	Status
0	0	Data set is being written to FIFO
0	1	Empty
1	0	Data set already written to FIFO
1	1	Zero length packet received

When a good reception is completed and the data set has been successfully read by CPU, firmware must set the FFRC bit of RXCON to advance the write pointer and read pointer to the start point of the FIFO to set up for receiving the next data set. When a bad reception is completed, the write pointer can be reversed to the position of the start point of the FIFO to enable the SIL to re-write the last data set for re-reception. The pointer advance and reversal are accomplished automatically by hardware. Table 24 summarizes how actions following a reception depend on RXERR and RXACK.

Table 24 Truth Table for Receive FIFO Management

RXERR	RXACK	Action at End of Transfer Cycle
0	0	No operation
0	1	Read Pointer and Write Pointer are set to the start point of FIFO when firmware sets the FFRC bit of RXCON
1	0	Write Pointer is set to the start point of FIFO

9.4.5 Setup Token Receive FIFO Handling

ETUP tokens received by the endpoint zero must be acknowledged, even if the receive FIFO is not empty. When a SETUP token is detected by the SIL, the SIL flushes the FIFO and sets the STOVW bit of RXSTAT for reset and locking the read pointer. These prevent RXURF bit of RXFLG and the read pointer from being set if the receive FIFO flush occurs in the middle of an CPU data read cycle. The STOVW bit is cleared and the EDOVW bit is set when a SETUP packet has been successfully acknowledged. The read pointer will remain locked until both the STOVW and EDOVW bits are cleared. For SETUP packets only, firmware must clear EDOVW before reading data from the FIFO. If this is not done, data read from the FIFO will be invalid. After processing a SETUP packet, firmware should always check the STOVW and EDOVW flags before setting the RXFFRC bit. When a SETUP packet either has been or is being received, setting of RXFFRC has no effect if either STOVW or EDOVW is set.

9.4.6 Suspend and Resume

In order to reduce the power consumption, MCU enters the suspend state when it has observed no bus traffic for 3ms. When in suspend, the CPU and its peripherals are in power down mode, an interrupt is enabled to support remote wakeup. The entire chip consumes less than 500 μ A in suspended states.

The MCU exits suspend mode when there is bus activity. A USB device may also request the host exits from suspend or selective suspend by using electrical signaling to indicate remote wakeup. The ability of a device to signal remote wakeup is optional and controlled by the USB host. Device states are described in Figure 37.

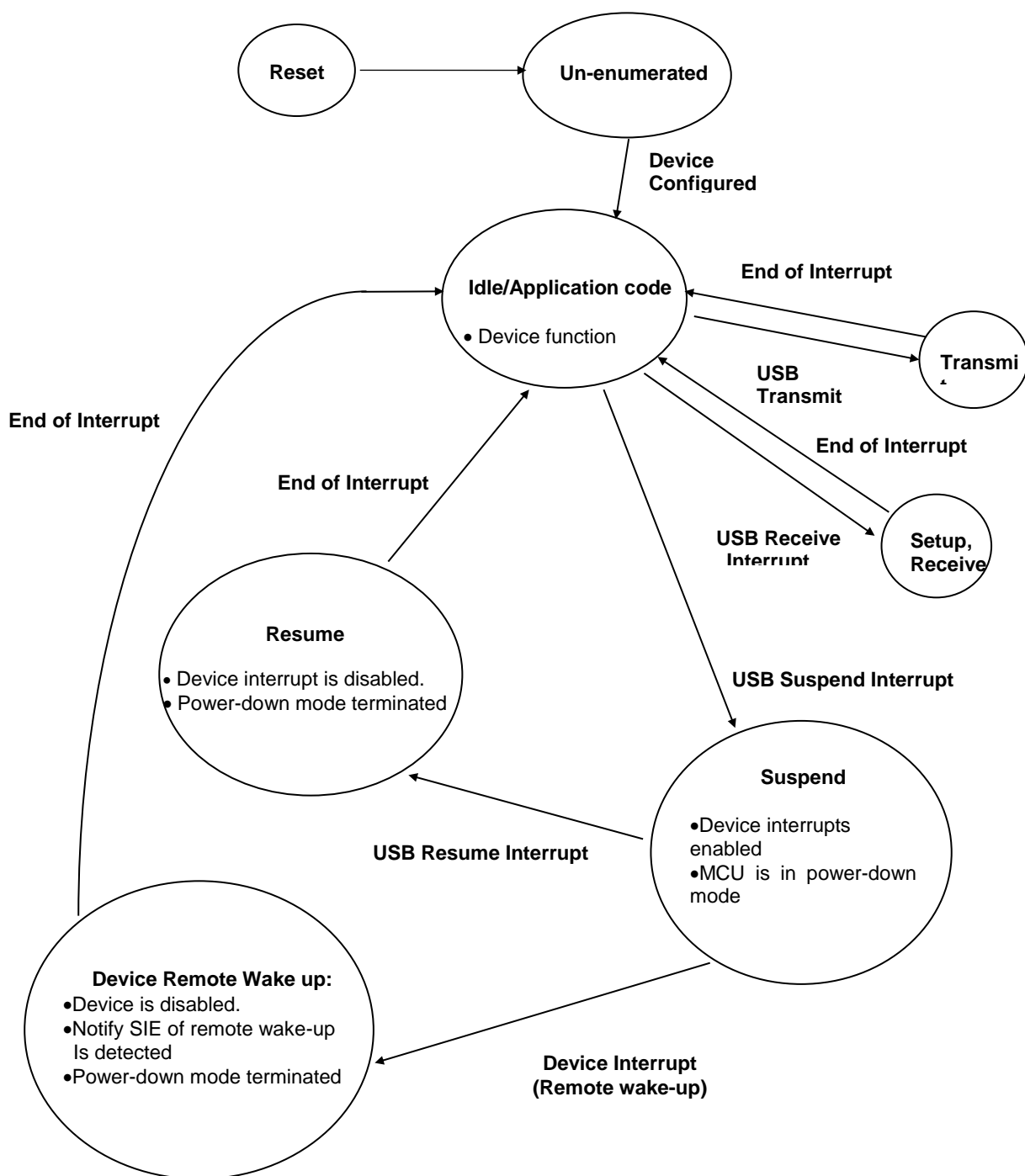


Figure 37 Suspend and Resume State Diagram

9.4.7 FIFO memory address mapping

In the initial state, each endpoint has its own FIFO memory space. If some endpoints are not used, their memory space can be reused as the second buffer for the other double-buffer-capable endpoints by properly program the **DBBANK** field of **USB_DBCON** register.

Table 25 FIFO Memory Address Mapping

bank	address	default EP0~6 and EP9 buffer space	Default EP7 & EP8 Isochronous buffer space
0	000h ~ 03fh	EP0_TX	
1	040h ~ 07fh	EP0_RX	
2	080h ~ 0bfb	EP1	
3	0c0h ~ 0ffh	EP2	
4	100h ~ 13fh	EP3	
5	140h ~ 17fh	EP4	
6	180h ~ 1bfb	EP5	
7	1c0h ~ 1ffh	EP6	
8	200h ~ 23fh		EP7
9	240h ~ 27fh		EP7
10	280h ~ 2bfb		EP7
11	2c0h ~ 2ffh		EP7
12	300h ~ 33fh		EP8
13	340h ~ 37fh		EP8
14	380h ~ 3bfb		EP8
15	3c0h ~ 3ffh		EP8
16	400h ~ 43fh		EP7
17	440h ~ 47fh		EP7
18	480h ~ 4bfb		EP7
19	4c0h ~ 4ffh		EP7
20	500h ~ 53fh		EP8
21	540h ~ 57fh		EP8
22	580h ~ 5bfb		EP8
23	5c0h ~ 5ffh		EP8
24	600h ~ 63fh	EP9	

$\text{start_addr}(\text{bank}) = 64 * \text{bank}$

$\text{start_addr_1}^{\text{st}}(\text{EP7}) = 200\text{h} + \text{EP7_BASE}$, $\text{start_addr_1}^{\text{st}}(\text{EP8}) = 200\text{h} + \text{EP8_BASE}$

$\text{start_addr_2}^{\text{nd}}(\text{EP7}) = 400\text{h} + \text{EP7_BASE}$, $\text{start_addr_2}^{\text{nd}}(\text{EP8}) = 400\text{h} + \text{EP8_BASE}$

10 Clock Recovery System

10.1 Main Features

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimable RC oscillator HSI48. The CRS provides a powerful means for oscillator output frequency evaluation, based on comparison with a selectable synchronization signal. It is capable of doing automatic adjustment of oscillator trimming based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In such case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, which is sent by a USB host at precise 1-ms intervals.

The synchronization signal can also be derived from the LSE oscillator output or from an external pin, or it can be generated by user software.

Main features

- Selectable synchronization source with programmable prescaler and polarity:
 - External pin
 - LSE oscillator output
 - USB SOF packet reception
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster start-up convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
 - Expected synchronization (ESYNC)
 - Synchronization OK (SYNCOK)
 - Synchronization warning (SYNCWARN)
 - Synchronization or trimming error (ERR)

10.2 Block Diagram

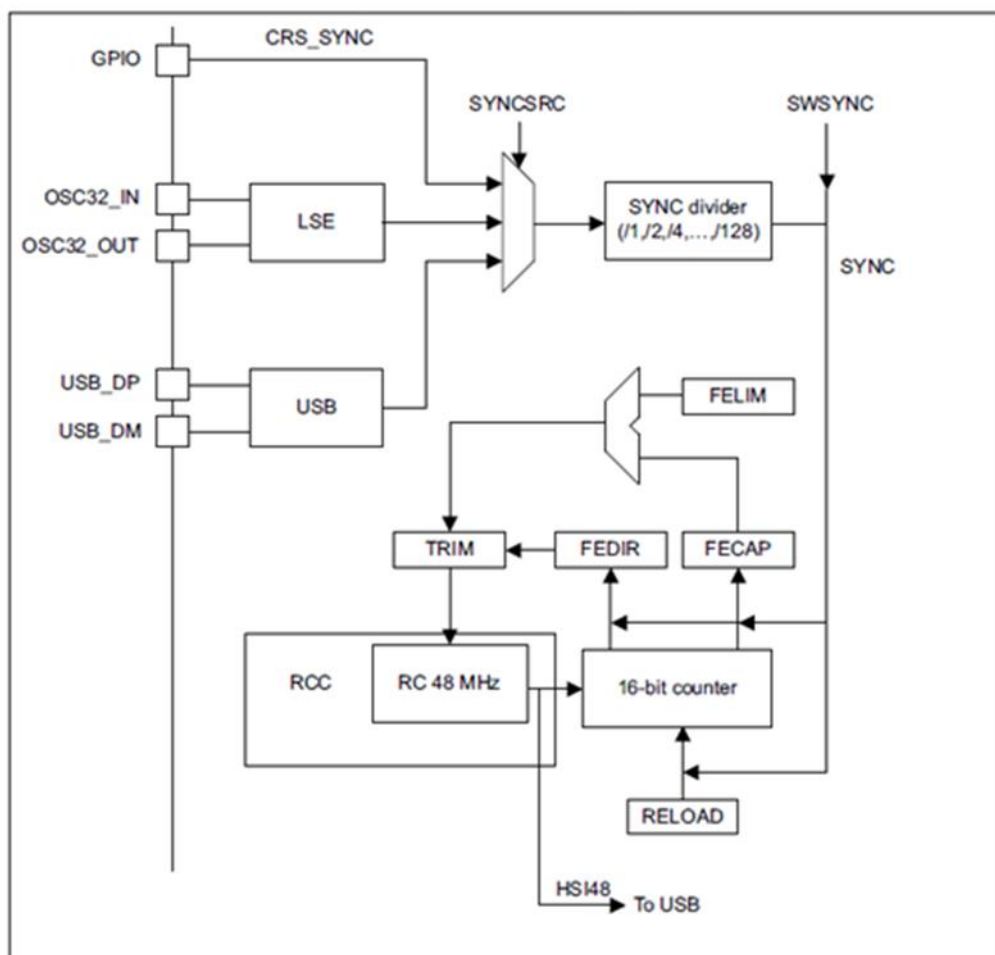


Figure 38 CRS block diagram

10.3 Register Definition

Table 26 CRS Control Register

Index	Bit	R/W	Default	Name	Description
CRS_CR: CRS control register					
00h	32:14	-	-	-	Reserved
	13:8	R/W	20h	TRIM	HSI48 oscillator smooth trimming This register is read-only when AUTOTRIMEN=1
	7	R/W	0	SWSYNC	Generate software SYNC event
	6	R/W	0	AUTOTRIMEN	Automatic trimming enable
	5	R/W	0	CEN	Frequency error counter enable
	4	-	-	-	Reserved

Index	Bit	R/W	Default	Name	Description
	3	R/W	0	ESYNCIE	Expected SYNC interrupt enable
	2	R/W	0	ERRIE	Synchronization or trimming error interrupt enable
	1	R/W	0	SYNCWARNIE	SYNC warning interrupt enable
	0	R/W	0	SYNCOKIE	SYNC event OK interrupt enable
CRS_CFGT: CRS configuration register					
04h*	31	R/W	0	SYNCPOL	SYNC polarity selection 0: SYNC active on rising edge (default) 1: SYNC active on falling edge
	30	-	-	-	Reserved
	29:28	R/W	2	SYNCSRC	SYNC signal source selection 0: GPIO selected as SYNC signal source 1: LSE selected as SYNC signal source 2: USB SOF selected as SYNC signal source (default) 3: Reserved
	27	-	-	-	Reserved
	26:24	R/W	0	SYNCDIV	SYNC divider 0: SYNC not divided (default) 1: SYNC divided by 2 2: SYNC divided by 4 3: SYNC divided by 8 4: SYNC divided by 16 5: SYNC divided by 32 6: SYNC divided by 64 7: SYNC divided by 128
	23:16	R/W	22h	FELIM	Frequency error limit
	15:0	R/W	bb7fh	RELOAD	Counter reload value
CRS_ISR: CRS interrupt/status register					
08h	31:16	R	0	FECAP	Frequency error capture
	15	R	0	FEDIR	Frequency error direction
	14:11	-	-	-	Reserved
	10	R	0	TRIMOVF	Trimming overflow or underflow
	9	R	0	SYNCMISS	SYNC missed
	8	R	0	SYNCERR	SYNC error
	7:4	-	-	-	Reserved
	3	R	0	ESYNCF	Expected SYNC flag
	2	R	0	ERRF	Error flag
	1	R	0	SYNCWARNF	SYNC warning flag
	0	R	0	SYNCOKF	SYNC event OK flag
CRS_ICR: CRS clear flag register					
0ch	31:4	-	-	-	Reserved
	3	W	0	ESYNCC	Expected SYNC clear flag
	2	W	0	ERRC	Error clear flag
	1	W	0	SYNCWARNC	SYNC warning clear flag
	0	W	0	SYNCOKC	SYNC event OK clear flag

* Notes:

1. TRIM register is read-only when AUTOTRIMEN=1
2. CRS_CFG register is read-only when CEN=1

10.4 Functional Description

10.4.1 Synchronization input

The CRS synchronization (SYNC) source, selectable through the CRS_CFGR register, can be the signal from the external CRS_SYNC pin, the LSE clock or the USB SOF signal. For a better robustness of the SYNC input, a simple digital filter (sampled by the HSI48 clock) is implemented to filter out any glitches. This source signal also has a configurable polarity and can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

10.4.2 Frequency error measurement

The frequency error counter is a 16-bit down/up counter which is reloaded with the RELOAD value on each SYNC event. It starts counting down till it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit where it eventually stops (if no SYNC event is received) and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the down-counting phase (before reaching the zero value), it means that the actual frequency is lower than the target (and so, that the TRIM value should be incremented), while when it is detected during the up-counting phase it means that the actual frequency is higher (and that the TRIM value should be decremented).

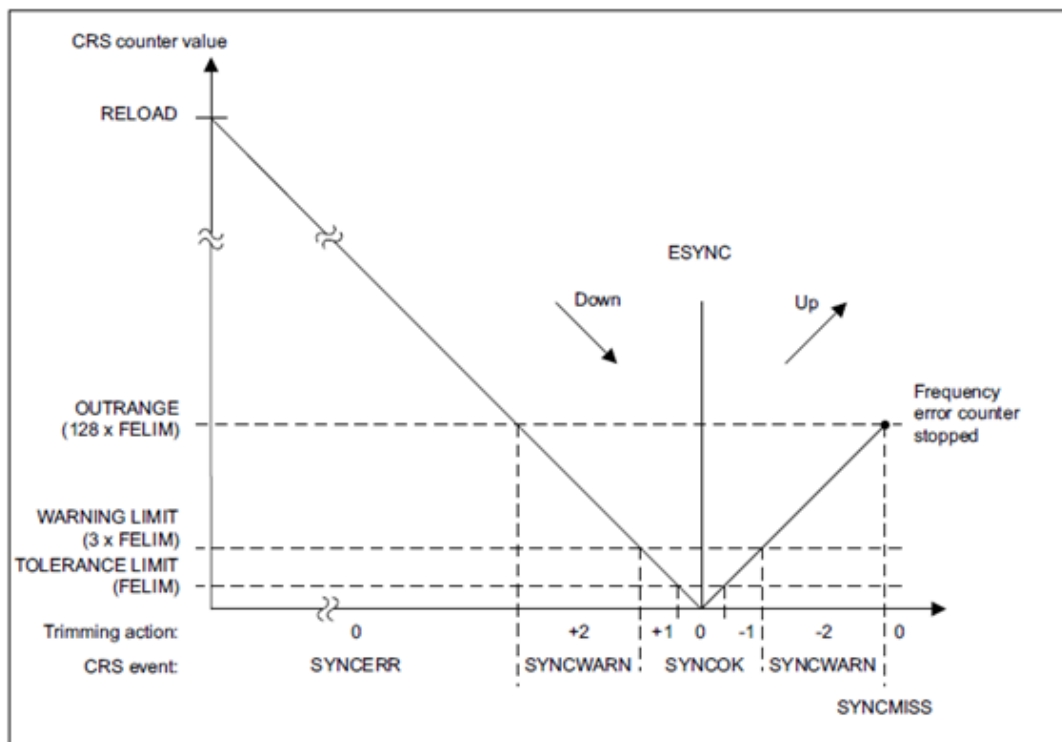


Figure 39 CRS Counter Behavior

10.4.3 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFGR register
- WARNING LIMIT, defined as 3 * FELIM value
- OUTRANGE (error limit), defined as 128 * FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one and that then, no trimming action is necessary.
 - SYNCOK status indicated
 - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
 - SYNCOK status indicated
 - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, it means that a stronger trimming action is necessary, and there is a risk that the optimal TRIM value will not be reached for the next period.
 - SYNCWARN status indicated
 - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, it means that the frequency is out of the trimming range. This can also happen when the SYNC input is not clean or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
 - SYNCERR or SYNCMISS status indicated
 - TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is so close to its limits that the automatic trimming would force it to overflow or underflow, then the TRIM value is set just to the limit and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

10.4.4 CRS initialization and configuration

RELOAD value

The RELOAD value should be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. It is then decreased by one in order to reach the expected synchronization on the zero value. The formula is the following:

$$\text{RELOAD} = (\text{fTARGET} / \text{fSYNC}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of HSI48 oscillator clock ticks. The following formula can be used:

$$\text{FELIM} = (\text{fTARGET} / \text{fSYNC}) * \text{STEP}[\%] / 100\% / 2$$

The result should be always rounded up to the nearest integer value in order to obtain the best trimming response. If frequent trimming actions are not wanted in the application, the trimming hysteresis can be increased by increasing slightly the FELIM value. The reset value of the FELIM field corresponds to $(\text{fTARGET} / \text{fSYNC}) = 48000$ and to a typical trimming step size of 0.14%.

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields which can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than $128 * \text{FELIM}$ value (OUTRANGE limit).

10.4.5 CRS low-power modes

Table 27 Effect of Low-Power Modes on CRS

Mode	Description
Sleep	No effect. CRS interrupts cause the device to exit the Sleep mode.
Stop	CRS registers are frozen.
Standby	The CRS stops operating until the Stop or Standby mode is exited and the HSI48 oscillator restarted.

10.4.6 CRS interrupts

Table 28 CRS Interrupt Control Bits

Interrupt event	Event flag	Enable control bit	Clear flag bit
Expected synchronization	ESYNCF	ESYNCE	ESYNCC
Synchronization OK	SYNCOF	SYNCOIE	SYNCOIC
Synchronization warning	SYNCF	SYNCFIE	SYNCFIC
Synchronization or trimming error (TRIMOVF, SYNCF, SYNCERR)	ERRF	ERRIE	ERRC

11 I2C

11.1 Main Features

I2C (inter-integrated circuit) bus Interface serves as an interface between the microcontroller and the serial I2C bus. It provides multi-master capability, and controls all I2C bus-specific sequencing, protocol, arbitration and timing. It supports standard and fast speed modes.

- Multi-master capability: the same interface can act as Master or Slave
- I2C Master features:
 - Clock generation
 - Start and Stop generation
- I2C Slave features:
 - Programmable I2C Address detection
 - Dual Addressing Capability to acknowledge 2 slave addresses
 - Stop bit detection
- Generation and detection of 7-bit/10-bit addressing and General Call
- Supports different communication speed:
 - Standard Speed Mode (up to 100 kHz)
 - Fast Mode (up to 400 kHz)
 - Fast Mode Plus (up to 1MHz)
- Status flags:
 - Transmitter/Receiver mode flag
 - End-of-Byte transmission flag
 - I2C busy flag
 - Master receiver return ACK(NACK) flag.
- Error flags:
 - Arbitration lost condition for master mode
 - Time out if SCL remained LOW for 25 ms (Timeout)
 - Detection of misplaced start or stop condition
 - Overrun/Underrun if clock stretching is disabled
- 1 Interrupt vector with various sources:
 - 1 Interrupt for address/data communication
 - 1 Interrupt for error/alert condition
- Optional Clock Stretching
- 1-byte buffer with DMA capability
- Configurable CRC (Cycle Redundancy Check) generation
 - CRC value can be transmitted as last byte in Tx mode
 - CRC error checking for last received byte

11.2 Block Diagram

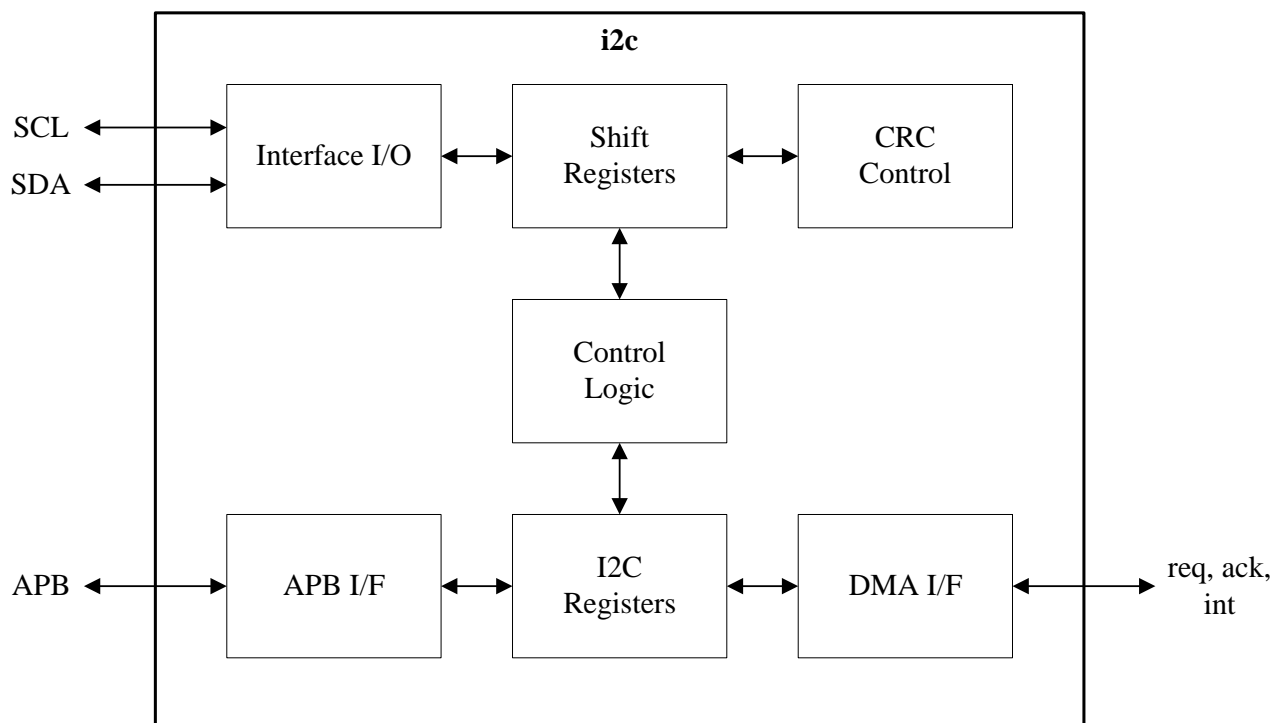


Figure 40 I2C Block Diagram

11.3 Register Definition

Table 29 I2C Control Register

Index	Bit	R/W	Default	Name	Description
I2C_CR1: I2C control register 1					
00h	31	R/W	0	I2cen	I2C peripheral enable 0: disabled. 1: enabled.
	30:23	-	-	-	Reserved
	22	R/W	0	mnackopt	Master mode received NACK option 0: auto stop. 1: continue to TX.
	21	R/W	0	snackopt	Slave mode transmitted data-NACK option 0: stop RX. 1: continue to RX.
	20	R/W	0	nosarb	Slave mode arbitration disable 0: with arbitration. 1: no arbitration.
	19:17	-	-	-	Reserved
	16	R/W	0		0: choose CMOS I/O level for I2C application
	12	R/W	0	gcen	General call address (00000000) enable 0: disabled.

Index	Bit	R/W	Default	Name	Description
					1: enabled.
	11	R/W	0	nostretch	Clock stretching disable (in slave mode) 0: Clock stretching enabled. 1: Clock stretching disabled.
	10	R/W	0	sbc	Slave bye control 0: disabled. 1: enabled.
	9	R/W	0	rxdmaen	DMA reception requests enable 0: disabled. 1: enabled.
	8	R/W	0	txdmaen	DMA transmission requests enable 0: disabled. 1: enabled.
	7	R/W	0	errie	Error interrupts enable 0: disabled. 1: enabled. Note: Any of these errors will generate an interrupt: arlo[0], berr[0], ovr[0], timeout[0], pecerr[0], alert[0]
	6	R/W	0	tcie	Transfer Complete interrupt enable 0: disabled. 1: enabled. Note: Any of these events will generate an interrupt: tc[0], tcr[0]
	5	R/W	0	stopie	STOP detection interrupt enable 0: disabled. 1: enabled.
	4	R/W	0	nackie	Not acknowledge received interrupt enable 0: disabled. 1: enabled.
	3	R/W	0	addrie	Address match interrupt enable 0: disabled. 1: enabled.
	2	R/W	0	rxie	RX interrupt (rxne[0]) enable 0: disabled. 1: enabled.
	1	R/W	0	txie	TX interrupt (txis[0]) enable 0: disabled. 1: enabled.
	0	-	-	-	Reserved
I2C_OAR1: I2C addr mode and oa1					
04h	31:12	-	-	-	Reserved
	11	R/W	0	addr_mode	0: 7-bit address mode 1: 10-bit address mode
	10:1	R/W	0	oa1	oa1[9:0] for 10-bit address oa1[6:0] for 7-bit address
	0	R/W	0	oa1_en	oa1 enable/disable
I2C_OAR2: I2C oa2					
08h	31:11	-	-	-	Reserved
	10:8	R/W	0	oa2msk	oa2 masks 000: no mask, all oa2[6:0] are compared 001: only oa2[6:1] are compared 010: only oa2[6:2] are compared

Index	Bit	R/W	Default	Name	Description
					011: only oa2[6:3] are compared 100: only oa2[6:4] are compared 101: only oa2[6:5] are compared 110: only oa2[6] is compared 111: all (except reserved) 7-bit received address are acknowledged.
	7:1	R/W	0	oa2	7-bit address
	0	R/W	0	oa2_en	oa2 enable/disable
I2C_TXDR: I2C 8-bit transmit data					
0ch	31:8	-	-	-	Reserved
	7:0	R/W	0	tx_data	8-bit transmit data
I2C_RXDR: I2C 8-bit receive data					
10h	31:8	-	-	-	Reserved
	7:0	R	0	rx_data	8-bit receive data
I2C_SADDR: I2C saddr					
14h	31:12	-	-	-	Reserved
	11	R/W	0	add10	0: 7-bit address mode 1: 10-bit address mode
	10:1	R/W	0	saad	saad[9:0] for 10-bit address saad[6:0] for 7-bit address
	0	R/W	0	rd_wrn	0: for write 1: for read
I2C_BCNT: I2C counter setting					
18h	31:10	-	-	-	Reserved
	9	R/W	0	auto_end	Automatic end mode (master mode) 0: software end mode: tc[0] flag is set when nbytes data are transferred, stretching SCL low. 1: automatic end mode: a STOP condition is automatically sent when nbytes[7:0] data are transferred. Note: this bit has no effect in slave mode or when the reload[0] bit is set.
	8	R/W	0	reload	NBYTES reload mode 0: the transfer is completed after the nbytes[7:0] data transfer (STOP or RESTART will follow). 1: the transfer is not completed after the nbytes[7:0] data transfer (nbytes[7:0] will be reloaded). tcr[0] flag was set when nbytes data are transferred, stretching SCL low.
	7:0	R/W	0	nbytes	Number of bytes The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with sbc[0]=0.
I2C_CR2: I2C control register 2					
1ch	31:4	-	-	-	Reserved
	3	R/W	0	swrst	Software Reset the I2C module
	2	R/W	0	nack	NACK generation (slave mode) This bit is set by software, clear by hardware when the NACK is sent, or when a STOP condition or an Address Matched is received. 0: an ACK is sent after current received byte. 1: an NACK is sent after current received byte.

Index	Bit	R/W	Default	Name	Description
					Note: Writing '0' to this bit has no effect.
	1	R/W	0	stop	Stop generation (master mode) This bit is set by software, clear by hardware when a Stop condition is detected. 0: No Stop generation. 1: Stop generation after current byte transfer.
	0	R/W	0	start	Start generation This bit is set by software, clear by hardware after the Start is sent. 0: No Start generation. 1: Start/Restart generation.
I2C_ISR: I2C status and control					
20h	31:16	-	-	-	Reserved
	15	R/W1c	0	busy	Bus Busy This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a Stop condition is detected.
	14	-	-	-	Reserved
	12	R/W1c	0	timeout	Timeout or t_{LOW} detection flag This flag is set by hardware when a timeout or extended clock timeout occurred.
	11	-	-	-	Reserved
	10	R/W1c	0	ovr	Overrun/Underrun (slave mode) 1: Overrun or underrun 0: No overrun/underrun
	9	R/W1c	0	arlo	Arbitration lost 1: Arbitration Lost detected 0: No Arbitration Lost detected
	8	R/W1c	0	berr	Bus error 1: Misplaced start or stop condition 0: No misplaced start or stop condition
	7	R/W	0	tcr	Transfer Complete Reload This flag is set by hardware when reload[0] =1 and nbytes[7:0] data have been transferred. It is cleared by software when nbytes is written to a non-zero value.
	6	R/W	0	tc	Transfer Complete (Master mode) This flag is set by hardware when reload[0]=0, auto_end[0]=0 and nbytes[7:0] data have been transferred. It is cleared by software when START bit or STOP bit is set.
	5	R/W1c	0	stopf	Stop condition flag This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer: - either as a master, provided that the STOP condition is generated by the peripheral. - or as a slave, provided that the peripheral has been addressed previously during this transfer.
	4	R/W1c	0	nackf	Not Acknowledge received flag This flag is set by hardware when a NACK is

Index	Bit	R/W	Default	Name	Description
					received after a byte transmission.
	3	R/W1c	0	addr	Address matched (slave mode) This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses.
	2	R/W	0	rxne	Receive data register not empty This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It's cleared when I2C_RXDR register is read.
	1	R/W	0	txis	Transmit interrupt status This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register. This bit can be written to '1' by software when nostretch[0]=1 only, in order to generate a txis[0] event.
	0	R/W	0	txe	Transmit data register empty This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register. This bit can be written to '1' by software in order to flush the transmit data register I2C_TXDR.
I2C_SR2: I2C status register					
24h	31:17	-	-	-	Reserved
	16	R	0	alertp	SMBA input status (master mode) 1: activated (the pin is in LOW level)
	15	R/W	0	busy	Bus busy 0: Not busy 1: Busy
	14	-	-	-	Reserved
	13	R/W	0	sdap	Filtered SDA pin input status
	12	R/W	0	sclp	Filtered SCL pin input status
	11	R/W	0	notactive	The peripheral is not in active state 0: Active 1: Not active
	10	R/W	0	slvactived	Slave address matched, auto cleared after STOP condition 0: Not matched 1: Matched
	9	R/W	0	master	Master is active 0: Not active 1: Active
	8	R/W	0	mrw	Master R/W (Receive/Transmit) status 0: Transmit 1: Receive
	7:1	R/W	0	addcode	Address match code (slave mode) These bit are updated with the received address when an address match event occurred (addr[0]=1). In the case of a 10-bit address, addcode[6:0]

Index	Bit	R/W	Default	Name	Description
					provides the 10-bit header followed by the 2 MSBs of the address.
	0	R/W	0	dir	Transfer direction (slave mode) 0: Write transfer, slave enters receiver mode. 1: Read transfer, slave enters transmitter mode.
I2C_PECR: I2C packet error checking					
28h	31:8	-	-	-	Reserved
	7:0	R	0	Pec	Packet error checking register This field contains the internal PEC.
I2C_TIMINGR1: I2C timing register 1					
2ch (i2c_timingr1)	31:20	-	-	-	Reserved
	19:16	R/W	0	filter	Digital noise filter for SCL/SDA inputs. The filter will filter spike with a length of up to filter[3:0]*t _{CLK}
	15:8	R/W	0	scldel	Data setup time t _{SCLDEL} = (scldel[7:0]+1)*t _{CLK}
	7:0	R/W	0	sdadel	Data hold time t _{SCLDEL} = (sdadel[7:0]+1)*t _{CLK}
I2C_TIMINGR2: I2C timing register 2					
30h (i2c_timingr2)	31:24	-	-	-	Reserved
	23:12	R/W	0	sclh	SCL high period (master mode) t _{SCLH} = (sclh[11:0]+1)*t _{CLK}
	11:0	R/W	0	scll	SCL low period (master mode) t _{SCLL} = (scll[11:0]+1)*t _{CLK}
I2C_TIMEOCTR1: I2C time-out register 1					
34h (i2c_timeoutr1)	31:16	-	-	-	Reserved
	15	R/W	0	timouten	Clock timeout enable 0: SCL timeout detection is disabled. 1: SCL timeout detection is enabled.
	14:13	-	-	-	Reserved
	12	R/W	0	tidle	Idle clock timeout detection 0: timeouta[11:0] is used to detect SCL low timeout 1: timeouta[11:0] is used to detect both SCL and SDA high timeout (bus idle condition)
	11:0	R/W	0	timeouta	Bus Timeout A This field is used to configure: - The SCL low timeout condition t _{TIMEOUT} when tidle[0] = 0 t _{TIMEOUT} = (timeouta[11:0] + 1)*2048*t _{CLK} - The bus idle condition (both SCL and SDA high) when tidle[0] = 1 t _{IDLE} = (timeouta[11:0] + 1)*4*t _{CLK}
I2C_TIMEOCTR2: I2C time-out register 2					
38h (i2c_timeoutr2)	31:16	-	-	-	Reserved
	15	R/W	0	texten	Extended clock timeout enable 0: disabled. 1: enabled.
	14:12	-	-	-	Reserved
	11:0	R/W	0	timeoutb	Bus Timeout B t _{LOW:EXT} = (timeoutb[11:0] + 1)*2048*t _{CLK}

Note: R/W1C: read & write one clear

11.4 Functional Description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), fast Mode (up to 400kHz), or Fast Mode Plus (up to 1MHz) I2C bus.

11.4.1 Mode Selection

The interface can operate in one of the following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

11.4.1.1 Communication Flow

In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection may be enabled or disabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the start condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

Acknowledge may be enabled or disabled in master mode by software. The master RX return NACK in last transfer. The I2C interface addresses (dual addressing 7-bit/ 10-bit and/or general call address) can be selected by software.

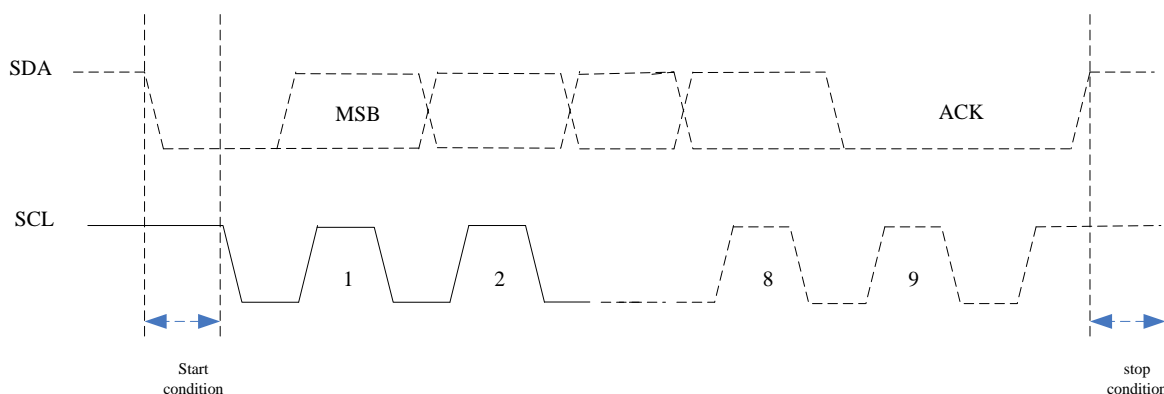


Figure 41 I2C Bus Protocol

11.4.2 I2C initialization

11.4.2.1 Noise filter

Before enabling the I2C peripheral, you must configure the digital noise filter, if needed. When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than the programming length set by the filter in I2C_TIMINGR1 register.

11.4.2.2 Timing

The timings must be configured in order to guarantee a correct data hold and setup time used in master and slave modes. This is done by programming the scldel[7:0] and sdacl[7:0] bits in the I2C_TIMINGR1 register. In addition, in master mode, the SCL clock high and low levels must be configured by programming the sclh[11:0] and scll[11:0] bits in the I2C_TIMINGR1 register.

11.4.2.3 Software reset

A software reset can be performed by setting the swrst[0] bit in the I2C_CR2 register. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

11.4.3 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

11.4.3.1 Reception

The SDA input fills the shift register. When the complete data byte is received, the shift register is copied into I2C_RXDR register if it is empty (rxne[0]=0). If rxne[0]=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR register is read if the clock stretching is enabled (nostretch[0]=0). The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).

11.4.3.2 Transmission

If the I2C_TXDR register is not empty (txe[0]=0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If txe[0]=1, meaning that no data is written yet in I2C_TXDR register, SCL line is stretched low until I2C_TXDR register is written. The stretch is done after the 9th SCL pulse.

11.4.3.3 Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and RESTART generation in master mode
- ACK control in slave receiver mode

The byte counter is always used in master mode. By default it is disabled in slave mode, but it can be enabled by software by setting the sbc[0] (Slave Byte Control) bit in the I2C_CR1 register.

The number of bytes to be transferred is programmed in the nbytes[7:0] bit field in the I2C_BCNTNTR register. If the number of bytes to be transferred (nbytes[7:0]) is greater than 255, or if a receiver wants to

control the acknowledge value of a received data byte, the reload mode must be selected by setting the reload[0] bit in the I2C_BCNTNTR register. In this mode, tcr[0] flag in the I2C_ISCR register is set when the number of bytes programmed in nbytes[7:0] has been transferred, and an interrupt is generated if tcie[0] is set. SCL is stretched as long as tcr[0] flag is set. The tcr[0] is cleared by software when nbytes[7:0] is written to a non-zero value. When the nbytes[7:0] counter is reloaded with the last number of bytes, reload[0] bit must be cleared.

When reload[0]=0 in master mode, the counter can be used in 2 modes:

- Automatic end mode (auto_end[0] = '1' in the I2C_BCNTNTR register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the nbytes[7:0] bit field has been transferred.
- Software end mode (auto_end[0] = '0' in the I2C_BCNTNTR register). In this mode, software action is expected once the number of bytes programmed in the nbytes[7:0] bit field has been transferred; the tc[0] flag is set and an interrupt is generated if the tcie[0] bit is set. The SCL signal is stretched as long as the tc[0] flag is set. The tc[0] flag is cleared by software when the START or STOP bit is set in the I2C_CR2 register. This mode must be used when the master wants to send a RESTART condition.

11.4.4 I2C Slave Mode

As soon as a start condition is detected, the address is received from the SDA line and sent to the shift register. Then it is compared with the address of register I2C_OAR1 (bit1~bit10 for 10 bit mode or bit1~7 for 7 bit mode) and with the address of register I2C_OAR2 (bit1~bit7, if oa2_en[0] bit=1) or the General Call address (if register I2C_CR1 bit0 (gcn[0]) = 1). Up to 7 oa2 LSB can be masked by configuring the oa2msk[2:0] bits in the I2C_OAR2 register. As soon as oa2msk[2:0] is not equal to 0, the address comparator for oa2[6:0] excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If oa2msk[2:0] =7, all received 7-bit addresses are acknowledged (except reserved addresses). oa2[6:0] is always a 7-bit address.

Note: In 10-bit addressing mode, the comparison includes the header sequence (11110xx0), where xx denotes the two most significant bits of the address.

- Address not matched: the interface ignores it and waits for another START condition.
- Address matched: the interface generates in sequence:
 - An interrupt is generated if the addrie in the I2C_CR1 register is set.
 - If several addresses are enabled you must read the addcode[6:0] bits in the I2C_SR2 register in order to check which address matched. The dir[0] flag must also be checked in order to know the transfer direction.

In 10-bit mode, after receiving the address sequence the slave is always in Receiver mode. It will enter Transmitter mode on receiving a repeated START condition followed by the header sequence with matching address bits and the least significant bit set (11110xx1).

The dir[0] flag in i2c_sr2 indicates whether the slave is in Receiver or Transmitter mode.

Slave Byte Control Mode

In order to allow byte ACK control in slave reception mode, Slave Byte Control mode must be enabled by setting the sbc[0] bit in the I2C_CR1 register. Reload mode must be selected in order to allow byte ACK control in slave reception mode (reload[0]=1). To get control of each byte, nbytes[7:0] must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the tcr[0] bit in is set, stretching the SCL signal low between the 8th and 9th SCL pulses. You can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the nack[0] bit in the I2C_CR2 register. The SCL stretch is released by programming nbytes[7:0] to a non-zero value: the acknowledge or not acknowledge is sent and next byte can be received. The nbytes[7:0] can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during nbytes[7:0] data reception.

Note 1: The sbc[0] bit must be configured when the I2C is disabled, or when the slave is not addressed, or when addr[0]=1. The reload bit value can be changed when addr[0]=1, or when tcr[0]=1.

Note 2: Slave Byte Control mode is not compatible with nostretch mode. Setting sbc[0] when nostretch[0]=1 is not allowed.

Slave transmitter

A transmit interrupt status (txis[0]) is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the txie[0] bit is set in the I2C_CR1 register. The txis[0] bit is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the nackf[0] bit is set in the I2C_ISCR register and an interrupt is generated if the nackie[0] bit in the I2C_CR1 register is set. The slave automatically releases the SCL and SDA lines in

order to let the master perform a STOP or a RESTART condition. The txis[0] bit is not set when a NACK is received.

When a STOP is received and the stopie[0] bit is set in the I2C_CR1 register, the stopf[0] flag is set in the I2C_ISCR register and an interrupt is generated. In most applications, the sbc[0] bit is usually programmed to '0'. In this case, if txe[0] = 0 when the slave address is received (addr[0]=1), you can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register by setting the txe[0] bit in order to program a new data byte.

In Slave Byte Control mode (sbc[0]=1), the number of bytes to be transmitted must be programmed in nbytes[7:0] in the address match interrupt subroutine (addr[0]=1). In this case, the number of txis[0] events during the transfer corresponds to the value programmed in nbytes[7:0].

Slave receiver

The rxne[0] is set in I2C_ISCR register when the I2C_RXDR register is full, and generates an interrupt if rxie[0] in I2C_CR1 register is set. The rxne[0] is cleared when I2C_RXDR register is read. When a STOP is received and stopie is set in I2C_CR1 register, stopf[0] is set in I2C_ISCR register and an interrupt is generated.

11.4.5 I2C Master Mode

Before enabling the I2C, software must configure the SCCH and SCLL in the I2C_TIMINGR1 register to set the master clock frequency. In order to support multi-master environment and slave clock stretching, a clock synchronization mechanism is implemented:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

11.4.5.1 Communication Initialization

To initiate the communication, following parameters must be programmed for the addressed slave in the I2C_SADDR register:

- Addressing mode (7-bit or 10-bit): add10[0]
- Slave address to be sent: sadd[9:0]
- Transfer direction: rd_wrn[0]
- The number of bytes to be transferred: nbytes[7:0]. If the number of bytes is equal to or greater than 255 bytes, nbytes[7:0] must initially be filled with 0xFF.

You must set the start bit in I2C_CR2 register. Changing all the above bits is not allowed when start[0] bit is set. The master will automatically send the START condition followed by the slave address as soon as it detects that the bus is free (busy[0] = 0). In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

11.4.5.2 Master Transmitter

For a write transfer, the txis[0] flag is set after each byte transmission, after the 9th SCL pulse when an ACK is received. An interrupt will be generated if the txie[0] bit in the I2C_CR1 register is set. Written data to the I2C_TXDR register will clear the flag.

The number of txis[0] events during the transfer corresponds to the value programmed in nbytes[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the reload[0] bit in the I2C_BCNTNTR register. In this case, when nbytes[7:0] data have been transferred, the tcr[0] flag in register I2C_ISCR is set and the SCL line is stretched low until nbytes[7:0] is written to a non-zero value.

The txis[0] flag is not set when a NACK is received. A STOP condition is automatically sent after the NACK reception. The nackf[0] flag is set in the I2C_ISCR register, and an interrupt is generated if the nackie[0] bit is set.

11.4.5.3 Master Receiver

For a read transfer, the rxne[0] flag is set after each byte reception, after the 8th SCL pulse. An rxne[0] event generates an interrupt if the rxie[0] bit is set in the I2C_CR1 register. The flag is cleared when i2c_rxdn is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the reload bit in the I2C_BCNTNTR register. In this case, when nbytes[7:0] data have been transferred, the tcr[0] flag is set and the SCL line is stretched low until nbytes[7:0] is written to a non-zero value.

When reload[0]=0 and nbytes[7:0] data have been transferred:

- In automatic end mode (auto_end[0]=1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (auto_end[0]=0), a NACK is automatically sent after the last received byte, the tc[0] flag is set and the SCL line is stretched low in order to allow software actions

A RESTART condition can be requested by setting the start bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the start bit clears the tc[0] flag and the START condition, followed by slave address, are sent on the bus.

A STOP condition can be requested by setting the stop bit in the I2C_CR2 register. Setting the stop[0] bit clears the tc[0] flag and the STOP condition is sent on the bus.

11.4.6 DMA requests

11.4.6.1 Transmission using DMA

DMA mode can be enabled for transmission by setting the txdmaen[0] bit in the I2C_CR1 register. Data will be loaded from a Memory area configured using the DMA to the I2C_TXDR register whenever the txis[0] bit is set.

Only the data is transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the nbytes[7:0] counter.
- In slave mode:
 - With nostretch[0]=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing addr[0].
 - With nostretch[0]=1, the DMA must be initialized before the address match event.

11.4.6.2 Reception using DMA

DMA mode can be enabled for reception by setting the `rxdmaen[0]` bit in the `I2C_CR1` register. Data will be loaded from the `I2C_RXDR` register to a Memory area configured using the DMA whenever the `rxne[0]` bit is set. Only the data (including `pec[0]`) are transferred with DMA.

- In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the `nbytes[7:0]` counter.
- In slave mode with `nostretch[0]=0`, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the `addr[0]` flag.

12 UART

12.1 Main Features

- Full duplex, asynchronous communications
- NRZ standard format
- Configurable oversampling method by 16 or by 8 clocks
- A baud rate generator
- 8 bits or 9 bits data word length
- 1 or 2 stop bits
- Separate enable bits for Transmitter or Receiver
- Configurable communication using DMA
 - Buffering of received/transmitted bytes in reserved SRAM
- IrDA SIR ENDEC
 - Support for 3/16 bit duration for normal mode
- Single-Wire half-duplex communication
- Transfer detection flags
 - Receive buffer full
 - Transmit buffer empty
 - End of Transmission
- Parity control
 - Even, odd or no-parity generation and detection
- Four error detection flags
 - Overrun error
 - Noise error
 - Frame error
 - Parity error
- Interrupt sources with flags
 - Transmit data register empty
 - Transmit complete
 - Receive data register full
 - Overrun error
 - Frame error
 - Noise error
 - Parity error

12.2 Block Diagram

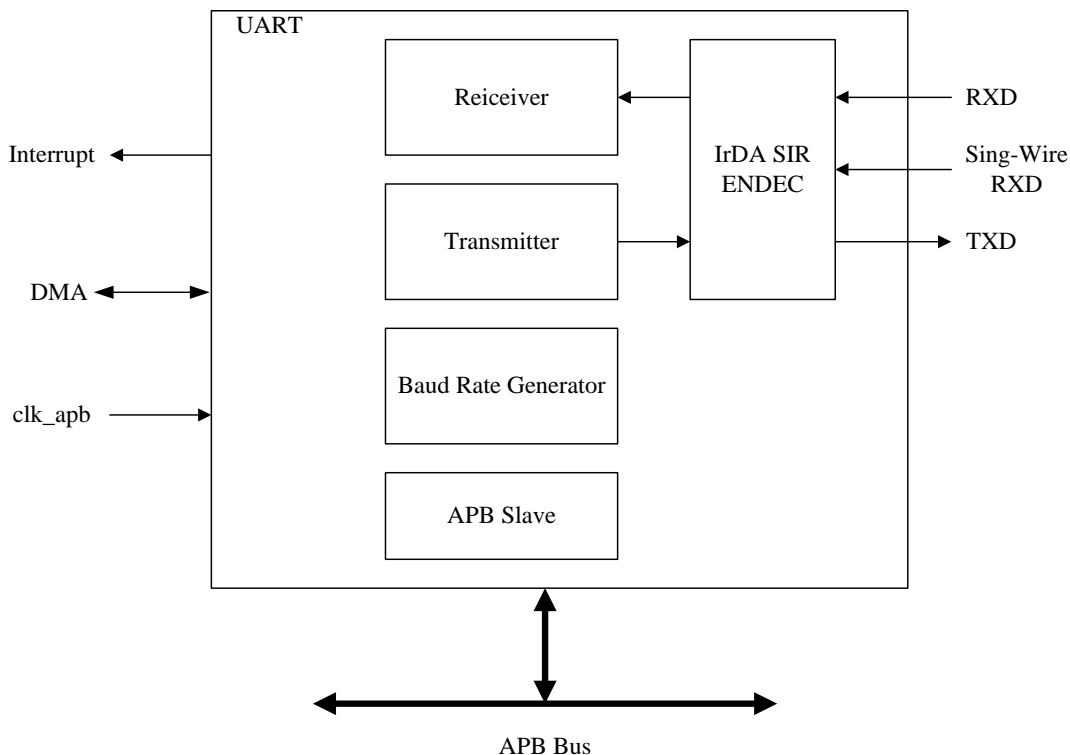


Figure 42 UART Block diagram

12.3 Register Definition

Table 30 UART Control Register

Index	Bit	R/W	Default	Name	Description
UART_CR1: Control register 1					
00h	31	R/W	0	UE	UART enable 0: UART prescaler and outputs disabled 1: UART enabled
	30:16	-	-	-	Reserved
	15	R/W	0	TE	Transmitter enable 0: Transmitter is disabled 1: Transmitter is enabled
	14	R/W	0	RE	Receiver enable 0: Receiver is disabled 1: Receiver is enabled and begins searching for a start bit
	13	R/W	0	OVER8	Over sampling mode 0: over sampling by 16 1: over sampling by 8 Note: Over sampling by 8 is not available in IrDA, when IREN =1 then OVER8 is forced to '0' by hardware.
	12	R/W	0	WORD_L	Word length

Index	Bit	R/W	Default	Name	Description
					0: 1 Start bit, 8 Data bits, n Stop bit 1: 1 Start bit, 9 Data bits, n Stop bit
	11	R/W	0	DMAT	DMA enable transmitter 1: DMA mode is enabled for transmission 0: DMA mode is disabled for transmission
	10	R/W	0	DMAR	DMA enable receiver 1: DMA mode is enabled for reception 0: DMA mode is disabled for reception
	9:3	-	-	-	Reserved
	2	R/W	0	PCE	Parity control enable 0: disable 1: enable
	1	R/W	0	PS	Parity selection 0: Even 1: Odd
	0	R/W	0	STOP	0: STOP = 1bit 1: STOP = 2bit
UART_CR2: Control register 2					
04h	31:7	-	-	-	Reserved
	6	R/W	0	TXEIE	TXE interrupt enable 0: Interrupt is inhibited 1: An UART interrupt is generated whenever TXE=1
	5	R/W	0	TCIE	Transmission complete interrupt enable 0: Interrupt is inhibited 1: An UART interrupt is generated whenever TC=1
	4	R/W	0	RXNEIE	RXNE interrupt enable 0: Interrupt is inhibited 1: An UART interrupt is generated whenever ORE=1 or RXNE=1
	3	R/W	0	EIE	Error Interrupt enable 0: Interrupt is inhibited 1: An interrupt is generated whenever FER=1 or ORE=1 or NE=1..
	2:1	-	-	-	Reserved
	0	R/W	0	PEIE	PE interrupt enable 0: Interrupt is inhibited 1: An UART interrupt is generated whenever PE=1
UART_STS: Status register					
08h	31:7	-	-	-	Reserved
	6	R/W	0	TXE	Transmit data register empty 0: Data is not transferred to the shift register 1: Data is transferred to the shift register)
	5	R/W	0	TC	UART: Transmission complete 0: Transmission is not complete 1: Transmission is complete Clear by S/W write to 1 or Clear by H/W when transmit data register is not empty.
	4	R/W	0	RXNE	UART: Read data register not empty 0: Data is not received

Index	Bit	R/W	Default	Name	Description
					1: Received data is ready to be read. Clear by S/W read Receive DATA Register or S/W write to 1
	3:0	-	-	-	Reserved
UART_TXDR: Transmit data register					
0Ch	31:9	-	-	-	Reserved
	8:0	W	0	TDR[8:0]	Transmit UART Data value
UART_RXDT: Receive data register					
10h	31:9	-	-	-	Reserved
	8:0	R	0	RDR[8:0]	Receive Data value
UART_BRR: Baudrate Register					
14h	31:16	-	-	-	Reserved
	15:4	R/W	0	Mantissa[11:0]	mantissa of Baud Rate Generator
	3:0	R/W	0	Fraction[3:0]	fraction of Baud Rate Generator
UART_CR3: Control register 3					
18h	31:15	-	-	-	Reserved
	14	R/W	0	IREN	IrDA mode enable 0: IrDA disable 1: IrDA enable
	13	R/W	0	HDSEL	Half-duplex selection 0: Half-duplex disable 1: Half-duplex enable
	12:0	-	-	-	Reserved
UART_FERE: Framing error detection					
24h	31:4	-	-	-	Reserved
	3	R/W	0	FERE	Framing Error Detection Enable 0: Framing error detection is disabled 1: Framing error detection is enable
	2:0	-	-	-	Reserved
UART_ERR: Error flag register					
30h	31:8	-	-	-	Reserved
	7	R/W	0	ORE	Overrun error 0: No overrun error 1: Overrun error is detected When this bit is set, the RDR register content will not be lost but the shift register will be overwritten. Clear by S/W write to 1
	6	R/W	0	NE	Noise error flag 0: No noise is detected 1: Noise is detected Clear by S/W write to 1
	5	R/W	0	PE	UART Parity bit error flag 0: No parity error 1: Parity error Clear by S/W write to 1
	4	-	-	-	Reserved
	3	R/W	0	FER	Framing Error Flag 0: Framing error is not detected 1: Framing error has been detected Clear by S/W write to 1.

Index	Bit	R/W	Default	Name	Description
					Error detection condition: The stop bit of each data byte is low in reception process.
	2:0	-	-	-	Reserved

12.4 Functional Description

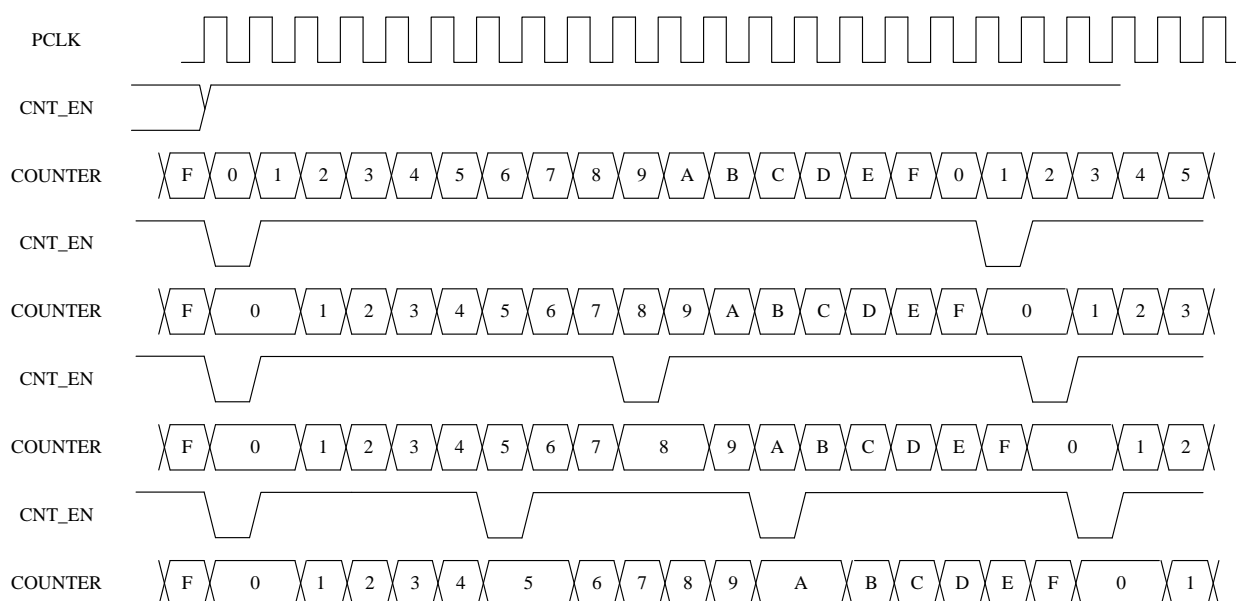
12.4.1 Fractional baud rate generation

$$\text{Baud rate} = \frac{\text{Sysclk}}{8 * (2 - \text{OVER8}) * (\text{Mantissa} + \frac{\text{Fraction}}{16})}$$

Table 31 UART Baud Rate Table

	Sysclk 16MHz (OVER8=0)			Sysclk 16MHz (OVER8=1)		
	Baud Rate Register	Actual	Error	Baud Rate Register	Actual	Error
2.4 K	416.625	2400	0.0%	833.3125	2400	0.0%
9.6 K	104.125	9604	0.04%	208.3125	9601	0.01%
19.2 K	52.0625	19208	0.04%	104.125	19208	0.04%
38.4 K	26.0625	38369	0.08%	52.0625	38415	0.04%
57.6 K	17.3125	57762	0.28%	34.75	57554	0.08%
115.2 K	8.625	115942	0.64%	17.375	115108	0.08%
230.4 K	4.3125	231884	0.64%	8.6875	230216	0.08%
921.6K	NA	NA	NA	2.1875	914286	0.8%
2 M	NA	NA	NA	1	2M	0.0%
3 M	NA	NA	NA	NA	NA	NA

Example: $\frac{Pclk}{16 * (1 + \frac{1}{16})}$, $\frac{Pclk}{16 * (1 + \frac{2}{16})}$, $\frac{Pclk}{16 * (1 + \frac{3}{16})}$



Example: $\frac{Pclk}{16 * (3 + \frac{3}{16})}$

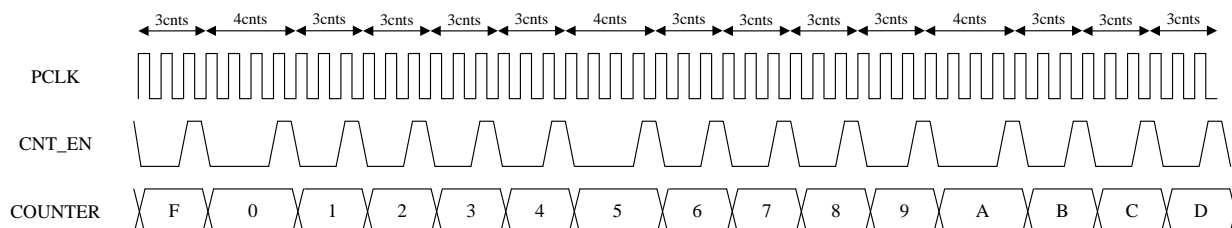


Figure 43 Example for Baud Rate Generator

12.4.2 Receiver

Overrun error

An overrun error occurs when a character is received when RXNE has not been reset. Data cannot be transferred from the shift register to the RDR register until the RXNE bit is cleared.

Noise error

Data sampling when oversampling by 16, sample value is bit 8, 9, 10.

Data sampling when oversampling by 8, sample value is bit 4, 5, 6.

The Noise error flag bit is set at the rising edge of the RXNE bit.

Table 32 UART Noise Error

Sample value	Noise error	Received bit
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit of each data byte is low in reception process.

12.4.3 Parity control

Table 33: Parity control table

WORD_L bit	PCE bit	UART Data Frame
0	0	Start + 8 bit data + Stop
0	1	Start + 7 bit data + parity bit + Stop
1	0	Start + 9 bit data + Stop
1	1	Start + 8 bit data + parity bit + Stop

12.4.4 Single-wire half-duplex communication

The single-wire half-duplex mode is selected by setting the HDSEL bit. In this mode, CLKEN and IREN bit must keep cleared.

When HDSEL is written to 1:

1. The TX and RX lines are internally connected.
2. The RX pin is no longer used.
3. The TX pin is floating input when no data is transmitted

13 SPI

13.1 Main Features

The Serial Peripheral Interface (SPI) module is a half/full-duplex, synchronous serial interface useful for communicating with other peripheral or microcontroller devices.

- Master or slave operation
- Programmable clock polarity and phase
- Programmable data order with msb-first or lsb-first shifting
- Dedicated transmission and reception flags with interrupt capability
- Full-duplex synchronous transfers on three lines
- Simplex synchronous transfers on two lines with or without a bidirectional data line
- 8 or 16 bits transfer frame format selection
- Multimaster mode capability with bus busy status flag
- Master or slave mode baud rate up to $\text{clk_apb} / 2$
- NSS management by hardware or software for both master and slave
- Hardware CRC feature with configurable polynomial for reliable communication
- Miscellaneous error flags with interrupt capability
- Transmission and reception buffer with DMA capability
- Capable for next access delay configuration
- Capable for master rx-only pseudo count

13.2 Block Diagram

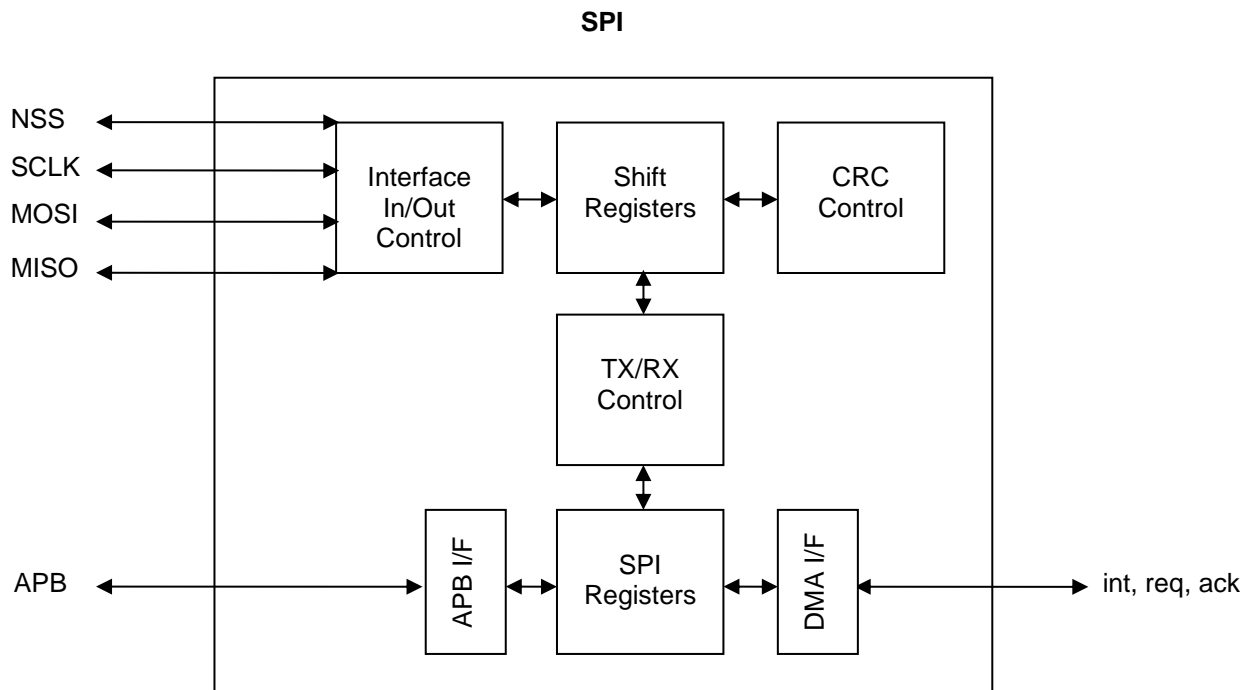


Figure 44 SPI module block diagram

13.3 Register Definition

Table 34 SPI Control Register

Index	Bit	R/W	Default	Name	Description
SPI_CR1: SPI control register 1					
00h	15	R/W	0	SPE	SPI enable, if MODFIF occurs, this bit will be cleared Note: SCLK should be kept idle state while SPE is turned off
	14	R/W	0	HSCC	high-speed clock compensation set HSCC if SCLK ≥ 12MHz
	13	R/W	0	BIDIMODE	bidirectional data mode enable (simplex)
	12	R/W	0	BIDIOE	bidirectional mode output enable 0: RX 1: TX
	11	R/W	0	CRCE	hardware CRC enable
	10	R/W	0	CRCNEXT	CRC transfer next write '1' after last TX data is written or after second last data is read Note: this bit is automatically cleared after CRC is generated/checked
	9:8	-	-	-	Reserved
	7	R/W	0	MSTR	master selection
	6	R/W	0	SSOE	slave select output enable (master mode), if

Index	Bit	R/W	Default	Name	Description
					SSOE is set, NSS input is ignored and MODFIF should not be set in master mode
	5	R/W	0	SSM	software slave mode 0: master mode: detected NSS signal is controlled by SPE slave mode: SPI detects the NSS signal coming from NSS port SPI is in hold state if detected NSS is inactive 1: SPI detects the NSS signal coming from SSI setting
	4	R/W	0	SSI	software slave input software NSS signal level, it's valid only SSM = '1'
	3	R/W	0	MODE16	frame length setting (8/16 bits) 0: 8 1: 16
	2	R/W	0	LSBFE	LSB-first enable
	1	R/W	0	CPOL	clock polarity 0: sclk = '0' when bus is idle 1: sclk = '1' when bus is idle
	0	R/W	0	CPHA	clock phase 0: first clock transition is the data capture edge 1: second clock transition is the data capture edge
SPI_CR2: SPI control register 2					
04h	15:12	-	-	-	Reserved
	11:8	R/W	0h	NAD	next access delay (word to word delay, unit: 0.5bit) includes NSS to SCLK to NSS delay (master mode only) note: the BUSY flag is inactive during NAD period
	7:0	R/W	01h	BRS	bit rate selection, valid range is from 00h to ffh $SCLK = CLK_APB / ((BRS+1)*2)$ note: $CLK_AHB / ((BRS+1)*2) \leq 24MHz$
SPI_EN: SPI counter and enable bit					
08h	15:8	R/W	00h	PSCNT	pseudo master rx-only counter (for sclk generation)
	7	R/W	0	TXNFIE	TX not full interrupt enable
	6	R/W	0	RXNEIE	RX not empty interrupt enable
	5	R/W	0	MODFIE	mode fault interrupt enable
	4	R/W	0	CRCFIE	CRC fault interrupt enable
	3	R/W	0	TXOVRIE	TX overrun interrupt enable
	2	R/W	0	RXOVRIE	RX overrun interrupt enable
	1	R/W	0	TXDMAEN	TX DMA enable
	0	R/W	0	RXDMAEN	RX DMA enable
SPI_STS: SPI status flags					
0ch	15:8	-	-	-	Reserved
	7	R	1	TXNFIF	TX not full flag, write BUF_DAT to clear
	6	R	0	RXNEIF	RX not empty flag, read BUF_DAT to clear
	5	R/W	-	MODFIF	mode fault flag, write '1' to clear

Index	Bit	R/W	Default	Name	Description
					fault condition: master mode: NSS = input-0 slave mode: NSS = high but frame is incomplete
	4	R/W	-	CRCFIF	CRC fault flag, write '1' to clear (note)
	3	R/W	-	TXOVRIF	TX overrun flag, write '1' to clear
	2	R/W	-	RXOVRIF	RX overrun flag, write '1' to clear
	1	R	-	BUSY	busy status under TX/RX
	0	-	-	-	Reserved
SPI_DAT: SPI data buffer					
10h	15:0	R/W	00h	BUF_DAT	buffer data for TX/RX
SPI_CRC_TX: SPI CRC TX register					
18h	15:0	R	ffffh	CRC_TX	CRC TX data
SPI_CRC_RX: SPI CRC RX register					
1ch	15:0	R	ffffh	CRC_RX	CRC RX data

Note: Illegal CRCFIF condition under unidirectional TX should be handled by F/W.

13.4 Functional Description

Status Flags

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

TX buffer not full flag (TXNF)

When it is set, this flag indicates that the TX buffer is not full and the next data to be transmitted can be loaded into the buffer. The TXNF flag is cleared when writing to the BUF_DAT 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 BUF_DAT is read.

BUSY flag

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

When BUSY is set, it indicates that the SPI is busy communicating.

The BUSY 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.

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

The BUSY flag is set when a transfer starts.

It is cleared:

- when a transfer is finished
- when the SPI is disabled
- when a master mode fault occurs (MODFIF=1)

When communication is not continuous, the BUSY flag is low between each communication.

When communication is continuous:

- in master mode, the BUSY flag is kept high during all the transfers
- in slave mode, the BUSY flag goes low for half SPI clock cycle between each transfer

Error Flags

Mode fault (MODFIF)

Mode fault occurs when the master device has its input NSS pin pulled low or deactivated NSS is detected during data transfer for slave device, this automatically sets the MODFIF bit. Mode fault affects the SPI peripheral in the following ways:

- The MODFIF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.

Write '1' to MODFIF bit to clear the mode fault status.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODFIF bit clearing sequence. The interrupt caused by MODFIF is not controlled by SPE setting.

As a security, hardware does not allow the setting of the SPE bit while the MODFIF bit is set.

In a slave device, if an incomplete frame is processing and NSS is deactivated from low to high, the MODFIF bit will be set. An interrupt routine can be used to recover cleanly from this state by performing a reset or returning to a default state.

CRC error

This flag is used to verify the validity of the value received when the CRCE bit in the control register. The CRCERR flag register is set if the CRC value received is incorrect. The CRCFIF responses CRC check result after CRCNEXT is cleared by hardware.

Underrun & overrun condition

TXUDRIF

An underrun condition occurs when the device has not fill data bytes before frame transmission starts. When an underrun condition occurs:

- the TXUDRIF bit is set and an interrupt is generated if the TXUDRIE bit is set.

In this case, the transmitter buffer has no contents to transmit from the device, in this condition, transmitted data is meaningless if TXUDRIF is occurred.

Clearing the TXUDRIF bit is done by write '1' to the TXUDRIF register.

RXOVRIF

An overrun condition occurs when the TX device has sent data bytes and the RX device has not cleared the RXNE bit resulting from the previous data byte transmitted. When an overrun condition occurs:

- the RXOVRIF bit is set and an interrupt is generated if the RXOVRIE bit is set.

In this case, the receiver buffer contents will not be updated with the newly received data from the master device. A read from the BUF_DAT register returns this byte. All other subsequently transmitted bytes are lost.

Clearing the RXOVRIF bit is done by write '1' to the RXOVRIF register.

CRC Calculation

Two separate CRC calculators (on transmission and reception data flows) are implemented in order to check the reliability of transmitted and received data. The SPI offers CRC8 or CRC16 calculation depending on the data format selected through the mode selection bit. The CRC is calculated serially using the polynomial programmed in the CRC_POLY register.

CRC calculation is enabled by setting the CRCE register, and a low to high transition of CRCE resets CRC logic. The calculation is processed on the sampling clock edge defined by the CPHA and CPOL settings. The calculated CRC value is checked automatically at the end of the data block as well as for transfer managed by CPU or by the DMA. When a mismatch is detected between the CRC calculated internally on the received data and the CRC sent by the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC calculation depends on the SPI configuration and the chosen transfer management. For 8-bit mode, only CRC_POLY[7:0] is used, both initial CRC value and XOR CRC output are fixed to 0.

Reference CRC polynomials:

8-bit: $C(x) = x^8 + x^2 + x^1 + x^0$, CRC_POLY = 16'h0007

16-bit: $C(x) = x^{16} + x^{12} + x^5 + x^0$, CRC_POLY = 16'h1021

SPI Interrupts

Interrupt event	Event flag	Enable Control bit
Transmit buffer not full flag	TXNFIF	TXNFIE
Receive buffer not empty flag	RXNEIF	RXNEIE
Mode fault flag	MODFIF	MODFIE
CRC error flag	CRCFIF	CRCFIE
TX underrun error flag	TXUDRIF	TXUDRIE
RX overrun error flag	RXOVRIF	RXOVRIE

SPI interrupt is controlled by SPE setting, interrupt enable and interrupt flag, except MODFIF that is an interrupt source independent of SPE setting.

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 underrun/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 is turned on. Separate requests must be issued to the TX and RX buffers:

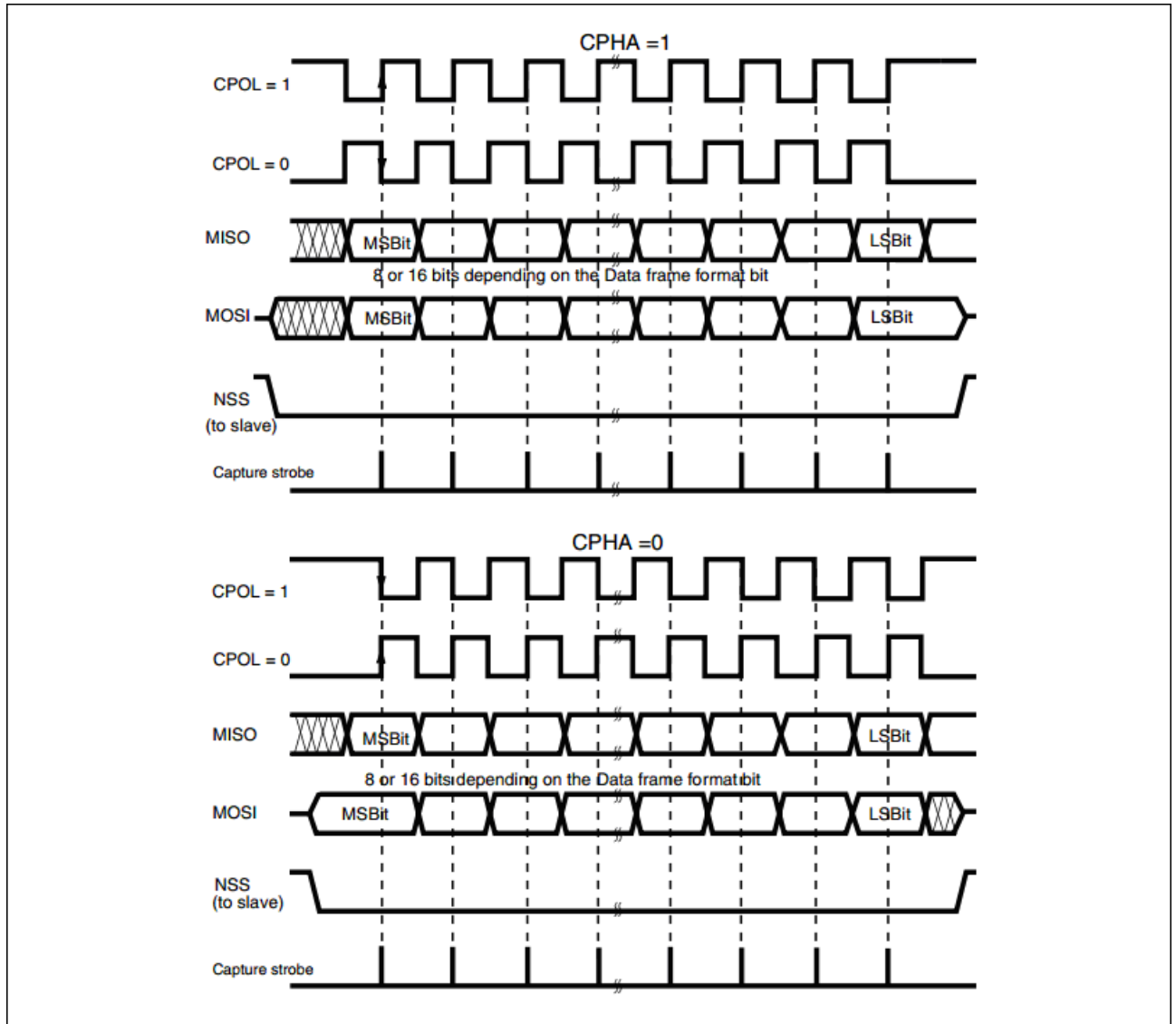
- In transmission, a DMA request is issued each time TXNF is set to 1. The DMA then writes to the BUF_DAT register (this clears the TXNF flag).
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the BUF_DAT register (this clears the RXNE flag).

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

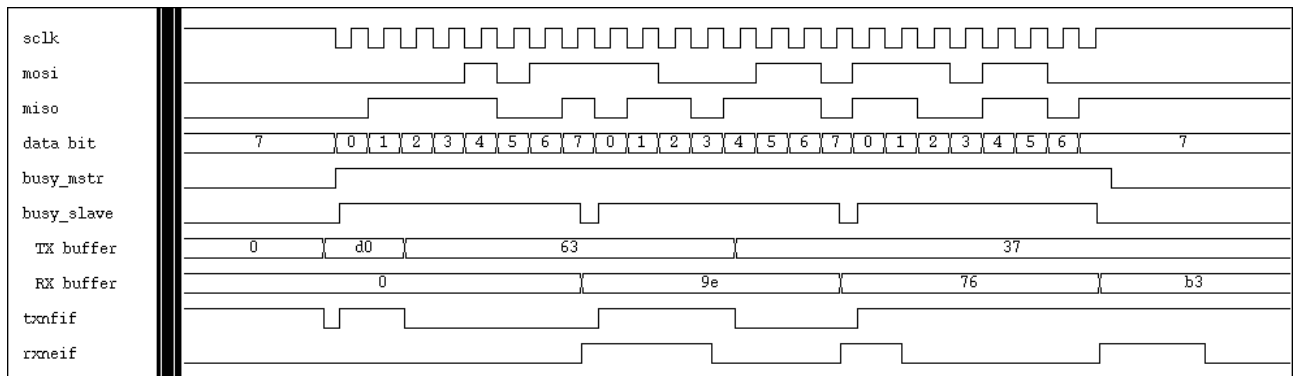
When the SPI is used only to receive data, it is possible to enable only the SPI RX DMA channel by setting PSCNT register.

In transmission mode, when the DMA has written all the data to be transmitted (flag TCIF is set in the DMA register), the BUSY 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 TXNF=1 and then until BUSY=0.

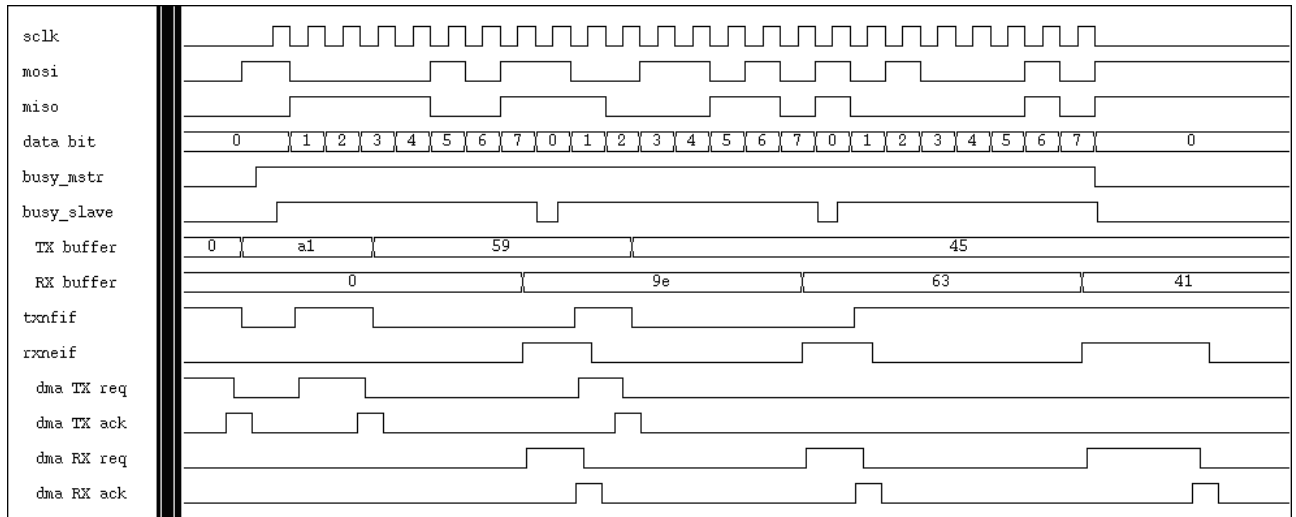
13.5 SPI Timing Diagram



Data clock timing diagram for $LSBFE = 0$



Signal behavior in master/slave mode (LSBFE = CPOL = CPHA = 1, MODE16 = 0)



Transmission/reception using DMA (LSBFE = 1, mode16 = CPOL = CPHA = 0)

Note: It needs at least 3 system clocks between falling NSS and first SPI clock transition, last SPI clock transition and rising NSS in slave mode

14 I2S

14.1 Main Features

- APB interface compatible
- I2S can operate as either master or slave
- Capable of handling 16, 24, and 32 bit word sizes
- I2S and MSB justified data format supported
- Support one output channel and one input channel (share BCLK and LRCK)
- Two 8 word FIFO data buffers are provided, one for transmit and the other for receive
- Two DMA requests, one for transmit and the other for receive
- Generates interrupt requests when buffer levels cross a programmable boundary
- The ratio of bit clock and left/right clock is selectable from 32/48/64

14.2 Block Diagram

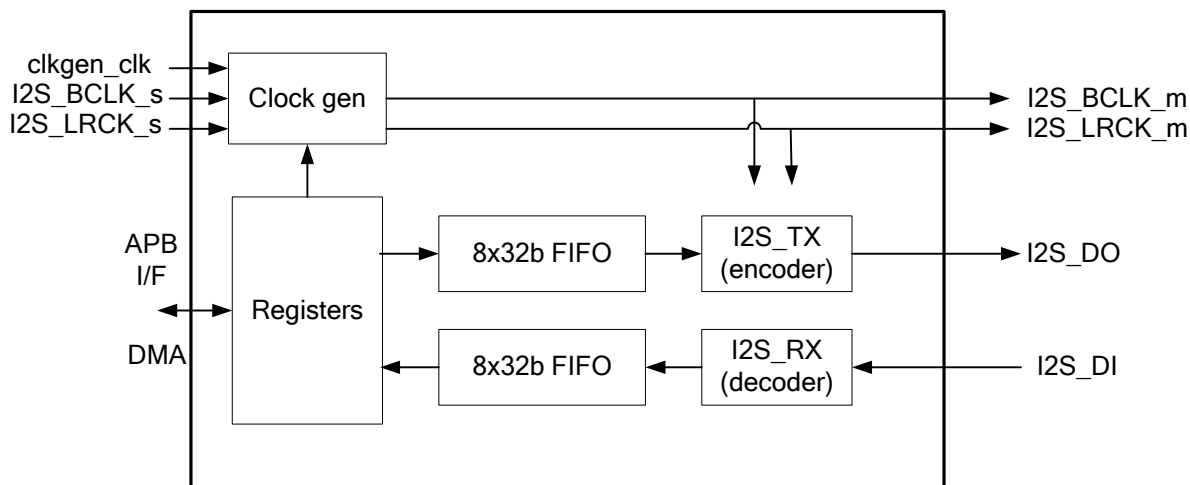


Figure 45 I2S block diagram

14.3 Register Definition

Table 35 I2S Control Register

Index	Bit	R/W	Default	Name	Description
I2S_CFG: I2S configuration register					
00h	31:20	-	-	-	Reserved
	19:18	R/W	0	BCLKGEN_SRC	Bit clock generator's clock source selection: 0 or 2: HSE 1: PLL

Index	Bit	R/W	Default	Name	Description
					3: Reserved 11: reserved
	17:11	R/W	05h	BCLK_LOW_DUTY	Bit clock low duty, clock base is system clock
	10:4	R/W	05h	BCLK_HIGH_DUTY	Bit clock high duty, clock base is system clock
	3:2	R/W	0	BCLK_RATE	0 or 1: 32 x fs, 2: 48 x fs, 3: 64 x fs
	1	R/W	0	LJ_MODE	1: left justified, 0: I2S mode
	0	R/W	1	MASTER_SLAVE	1: master mode, 0: slave mode
I2S_DCFG: I2S data configuration					
04h	31:20	-	-	-	Reserved
	19:18	R/W	0	RX0_THRESHOLD	Generate INT/DMA when there are n words data received in the FIFO (RX0 channel) 0: 1 word, 1: 2 words, 2: 3 words, 3: 4 words
	17:16	R/W	0	RX0_BLEN	Number of bits of audio data (RX0 channel) 0 or 1: 16-bit, 2: 24-bit, 3: 32-bit
	15:4	-	-	-	Reserved
	3:2	R/W	0	TX0_THRESHOLD	Generate INT/DMA when there are n words free space in the FIFO (TX0 channel) 0: 1 word, 1: 2 words, 2: 3 words, 3: 4 words
	1:0	R/W	0	TX0_BLEN	Number of bits of audio data (TX0 channel) 0 or 1: 16-bit, 2: 24-bit, 3: 32-bit
I2S_CR: I2S TX/RX control register					
08h	31:20	-	-	-	Reserved
	19	R/W	0	RX0_FLUSH	Write 1 to flush FIFO (RX0 channel)
	18	R/W	0	RX0_INT_EN	Enable Interrupt (RX0 channel)
	17	R/W	0	RX0_DMA_EN	Enable DMA (RX0 channel)
	16	R/W	0	RX0_EN	Enable I2S (RX0 channel)
	12:4	-	-	-	Reserved
	3	R/W	0	TX0_FLUSH	Write 1 to flush FIFO (TX0 channel)
	2	R/W	0	TX0_INT_EN	Enable Interrupt (TX0 channel)
	1	R/W	0	TX0_DMA_EN	Enable DMA (TX0 channel)
	0	R/W	0	TX0_EN	Enable I2S (TX0 channel)
I2S_TXSR: I2S TX status					
0Ch	31:8	-	-	-	Reserved
	7	R/W1C		TX0_ERROR	An error (OV/UV) occurred (TX0 channel). Write 1 to clear.
	6	R	1	TX0_EMPTY	FIFO is empty (TX0 channel)
	5	R	0	TX0_FULL	FIFO is full (TX0 channel)
	4	R	1	TX0_INT	Interrupt flag (TX0 channel)
	3:0	R	0	TX0_CNT	# of words currently in the FIFO (TX0 channel)
I2S_RXSR: I2S RX status					
10h	31:8	-	-	-	Reserved
	7	R/W1C	0	RX0_ERROR	An error (OV/UV) occurred (RX0 channel). Write 1 to clear.
	6	R	0	RX0_EMPTY	FIFO is empty (RX0 channel)
	5	R	0	RX0_FULL	FIFO is full (RX0 channel)
	4	R	1	RX0_INT	Interrupt flag (RX0 channel)
	3:0	R	0	RX0_CNT	# of words currently in the FIFO (RX0 channel)
I2S_TX_FIFO: I2S TX FIFO					
20h	31:0	W	-	TX0_FIFO	TX FIFO (TX0 channel) 16-bit data: [31:16] is R-CH, [15:0] is L-CH. 24-bit data: [23:0] is R-CH or L-CH (L-CH goes

Index	Bit	R/W	Default	Name	Description
					first). 32-bit data: [31:0] is R-CH or L-CH (L-CH goes first).
I2S_RX_FIFO: I2S RX FIFO					
30h	31:0	R	-	RX0_FIFO	RX FIFO (RX0 channel)

R/W1C: read & write one clear

Note: When sampling rate is changed (high low duty is changed) in master mode, or master slave mode configuration is changed, please disable I2S enable register and enable it after new setting is configured.

14.4 Functional Description

The I2S module has two clock signals, BCLK (bit clock) and LRCK (left/right clock), and two data line for data receive and transmit (DI and DO). One data line contains two channels. Because only one BCLK and LRCK, only I2S encoder or decoder can be enable at the same time unless the DAC and ADC CODEC has the same BCLK and LRCK. The BCLK in master mode is divided from system clock and the clock period and high/low duration meets the I2S spec ($\pm 10\%$). The high/low duty of BCLK is configured by control register and the value infers the sampling rate.

14.4.1 The Basics of I2S Bus

Both master and slave modes of I2S are supported by the I2S interfaces of the MCU. Master mode means BCLK and LRCK are provided by the MCU as shown in Fig. 1(a). On the contrary, slave mode means BCLK and LRCK are provided by the I2S codecs as depicted in Fig. 1(b).

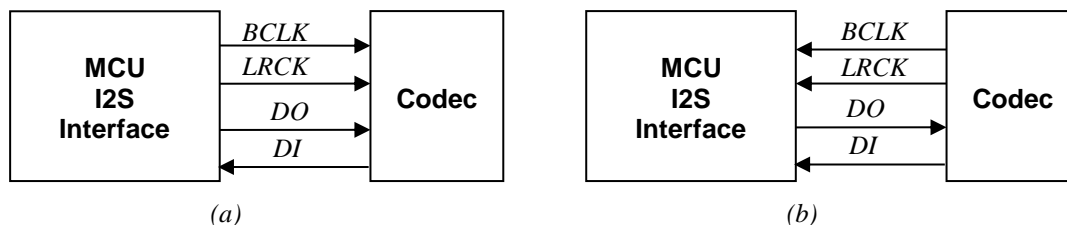


Figure 46 (a) I2S Master mode (b) I2S Slave mode

Figure 47 indicates the basic waveform of I2S. Note that LRCK is generated at the negative edges of BCLK with the ratios 1/64. Data lines are transited at the negative edges of BCLK, and are sampled at the positive edges of BCLK by codecs in case of playback or by MCU in case of recording.

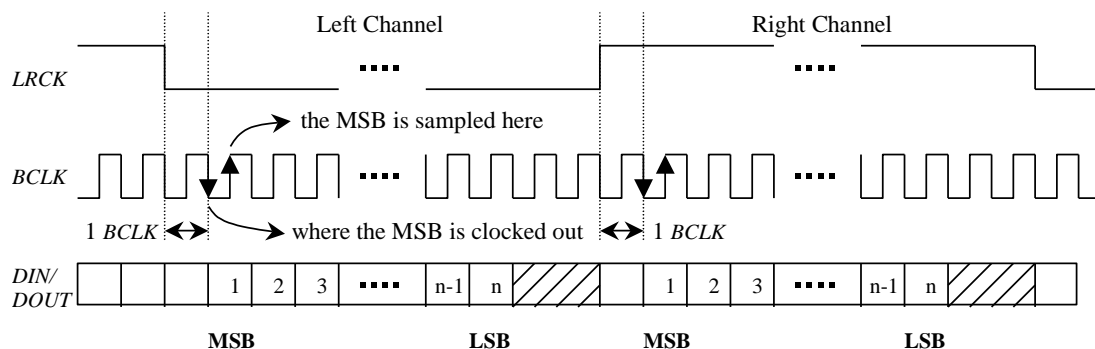


Figure 47 The basic timing diagram of the I2S interface

For the I2S DAC controller, the audio data is transformed from the parallel format to the serial format before transmitted. Then, the bit data is shifted out one by one with the MSB first via DOUT signal. In the same manner, the audio data is transformed from the coming serial format to the parallel format for an I2S ADC controller.

14.4.2 Left Justified Mode

In the left justified mode of the I2S DAC controller, the MSB data bit is clocked out by the MCU at the negative edge of BCLK which is aligned to the transition of LRCK. In the left justified mode of I2S ADC controllers, the MSB data bit is clocked out by codecs and sampled by the MCU at the first positive edge of BCLK which follows a LRCK transition. LRCK is high during left channel transmission and low during right channel transmission in the left justified mode. Figure 48 shows all of these.

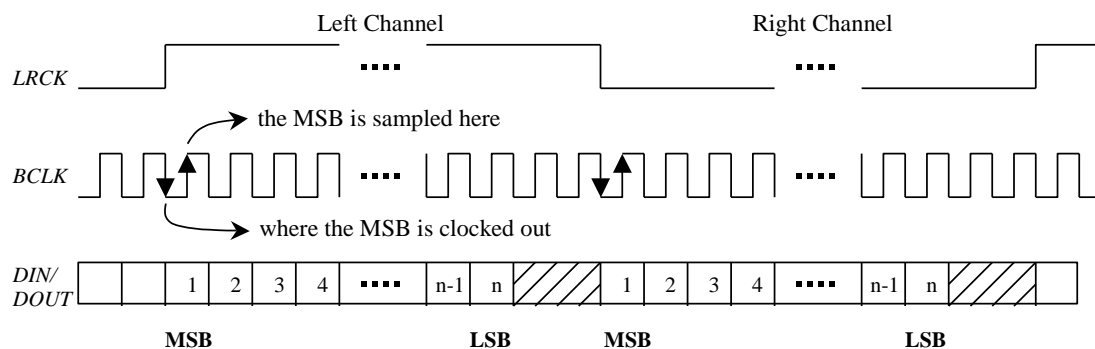


Figure 48 Left justified mode timing diagram of I2S interface

14.4.3 I2S Mode

In the I2S mode of the I2S DAC controller, the MSB data bit is clocked out by MCU at the first negative edge of BCLK which follows a LRCK transition. In the I2S mode of I2S ADC controllers, the MSB data bit is clocked out by codecs and sampled by the MCU at the second positive edge of BCLK which follows a LRCK

transition. LRCK is low during left channel transmission and high during right channel transmission in the I2S mode. Figure 49 indicates all of these.

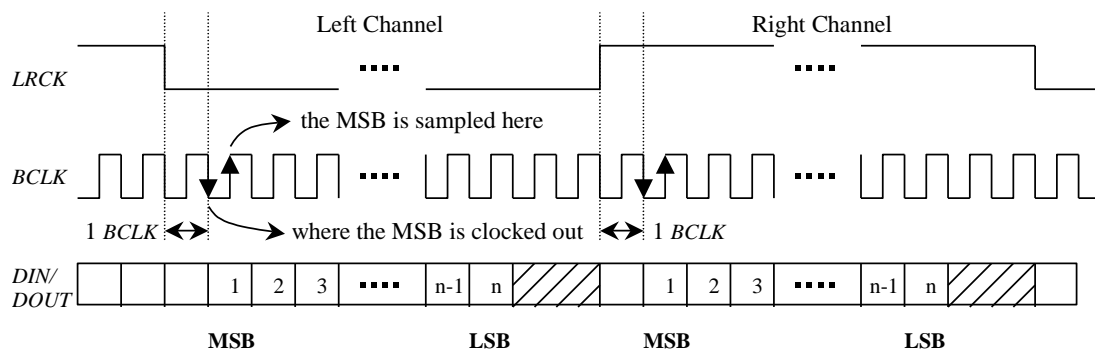


Figure 49 I2S mode timing diagram of I2S interface

14.4.4 I2S Clock Generation

The I2S bit clock is divided from system clock in master mode and the relationship of clock high/low duty and sampling rate is described as the table 36.

Table 36 I2S Sampling Rate

Main clock	Sampling rate	Clock high duty setting	Clock low duty setting	f_{BCLK} / f_{LRCK}	Bit clock rate	Ideal bit clock
18.432 MHz	48k	5	5	32	1536k	1536k
18.432 MHz	48k	3	3	48	2304k	2304k
18.432 MHz	48k	2	2	64	3072k	3072k
24.576 MHz	48k	7	7	32	1536k	1536k
24.576 MHz	48k	5	4	48	2234k	2304k
24.576 MHz	48k	3	3	64	3072k	3072k

Table 37 I2S System Clock = 24MHz (assume $f_{BCLK}=64fs$)

Sampling rate	Clock high duty	Clock low duty	Bit clock rate	Ideal bit clock
8k	22	22	521.74k	512k
11.025k	16	16	705.88k	705.6k
12k	15	15	750k	768k
16k	11	11	1000k	1024k
22.025k	8	7	1411.76k	1411.2k
24k	7	7	1500k	1536k
32k	5	5	2000k	2048k
44.1k	3	3	3000k	2822.4k
48k	3	3	3000k	3072k
88.2k	1	1	6000k	5644.8k
96k	1	1	6000k	6144k
192k	0	0	12000k	12288k

Table 38 I2S System Clock = 12MHz (assume $f_{\text{BCLK}} = 64\text{fs}$)

Sampling rate	Clock high duty	Clock low duty	Bit clock rate	Ideal bit clock
8k	11	10	521.74k	512k
11.025k	8	7	705.88k	705.6k
12k	7	7	750k	768k
16k	5	5	1000k	1024k
22.025k	4	3	1411.76k	1411.2k
24k	3	3	1500k	1536k
32k	2	2	2000k	2048k
44.1k	1	1	3000k	2822.4k
48k	1	1	3000k	3072k
88.2k	0	0	6000k	5644.8k
96k	0	0	6000k	6144k

15 Real-Time Clock

15.1 Main Features

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupt. The RTC includes also a programmable periodic interrupt. Registers contain the seconds, minutes, hours (24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically.

- Calendar with seconds, minutes, hours (24-hour format), day (day of week), date (day of month), month, and year.
- Programmable alarm with interrupt function. The alarm can be triggered by any combination of the calendar fields.
- Programmable periodic interrupt function
- Leap year correction
- Requires external 32.768 kHz clock crystal

15.2 Block Diagram

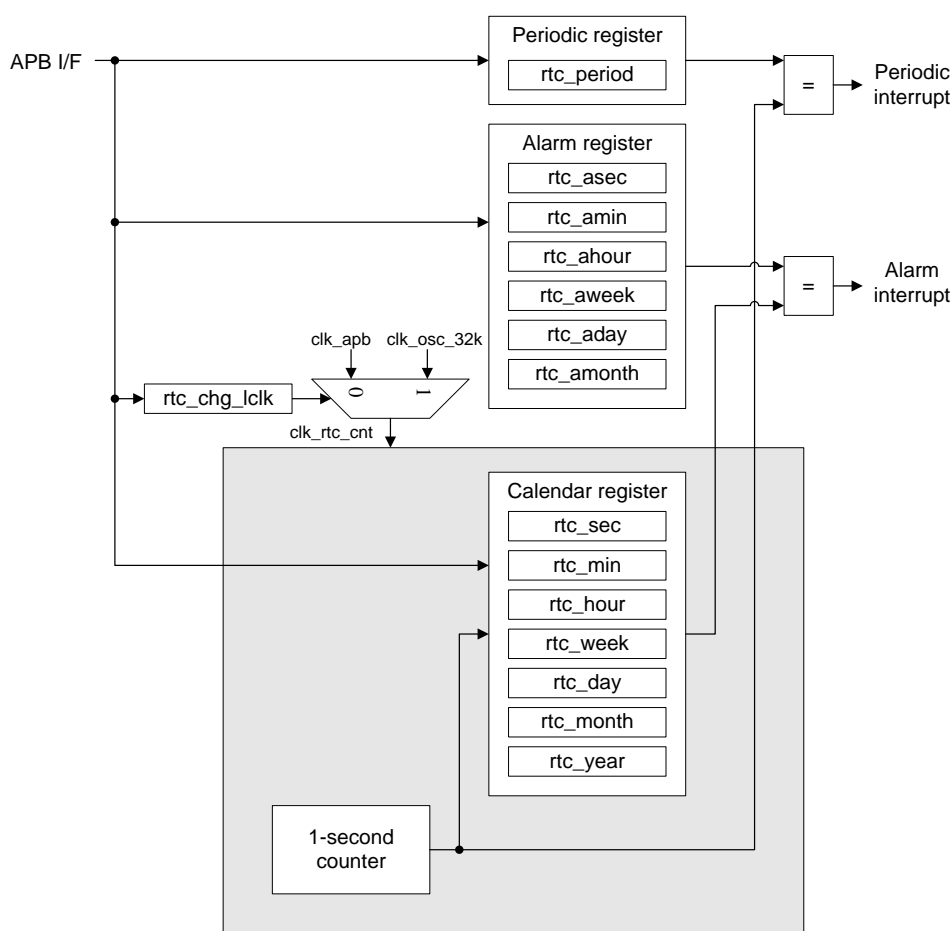


Figure 50 RTC Block Diagram

15.3 Register Definition

Table 39 RTC Control Register

Index	Bit	R/W	Default	Name	Description
RTC_CR: RTC control					
00h	31:16	-	-	-	Reserved
	15	R/W	0	rtc_en	RTC enable 0: disable RTC 1: enable RTC
	14:10	-	-	-	Reserved
	9	R/W	0	rtc_lse_en	Select RTC clock source. LSE: 32768Hz or LSI: 37 kHz 0: LSI 1: LSE
	8	R/W	0	rtc_chg_lclk	Enable RTC clock=32kHz low frequency clock 0: can read/write RTC register 1: can't read/write RTC register
	7:6	-	-	-	Reserved
	5:4	R/W	0	rtc_period	RTC period time 0: 1 s (1Hz) 1: 62.5 ms (16Hz) 2: 31.25 ms (32Hz) 3: 15.625 ms (64Hz)
	3:2	-	-	-	Reserved
	1	R/W	0	rtc_alarm_ie	RTC alarm interrupt enable 0: Disable alarm interrupt 1: Enable alarm interrupt
	0	R/W	0	rtc_period_ie	RTC periodic interrupt enable 0: Disable periodic interrupt 1: Enable periodic interrupt
RTC_ISR: RTC interrupt status register					
04h	30:2	-	-	-	Reserved
	1	R/W1c	0	rtc_alarm_if	RTC alarm interrupt flag 0: no event of alarm interrupt 1: event of alarm interrupt
	0	R/W1c	0	rtc_period_if	RTC periodic interrupt flag 0: no event of periodic interrupt 1: event of periodic interrupt
RTC_TIME: RTC time setting					
10h	31:22	-	-	-	Reserved
	21:16	R/W	0	rtc_hour	Hour coded in BCD, range is 0~23 rtc_hour[5:4]: hour tens rtc_hour[3:0]: hour units
	15	-	-	-	Reserved
	14:8	R/W	0	rtc_min	Minute coded in BCD, range is 0~59 rtc_min[6:4]: minute tens rtc_min[3:0]: minute units
	7	-	-	-	Reserved
	6:0	R/W	0	rtc_sec	Second coded in BCD, range is 0~59 rtc_sec[6:4]: second tens rtc_sec[3:0]: second units
RTC_DATE: RTC date setting					
14h	31:24	R/W	0	rtc_year	Year coded in BCD, range is 0~99

Index	Bit	R/W	Default	Name	Description
					rtc_year[7:4]: year tens rtc_year[3:0]: year units
	23:21	-	-	-	Reserved
	20:16	R/W	1	rtc_month	Month coded in BCD, range is 1~12 rtc_month[4]: month tens rtc_month[3:0]: month units
	15:14	-	-	-	Reserved
	13:8	R/W	1	rtc_day	Day coded in BCD, range is 1~31 rtc_day[5:4]: day tens rtc_day[3:0]: day units
	7:3	-	-	-	Reserved
	2:0	R/W	0	rtc_week	Week day coded in BCD, range is 0~6 0: Sunday 1: Monday 2: Tuesday 3: Wednesday 4: Thursday 5: Friday 6: Saturday
RTC_A_TIME: RTC alarm time setting					
18h	31:23	-	-	-	Reserved
	22	R/W	0	rtc_ahour_en	Enable alarm hour 0: disable, hour don't care in alarm comparison 1: enable, alarm set if the hour match
	21:16	R/W	0	rtc_ahour	Alarm hour coded in BCD, range is 0~23 rtc_ahour[5:4]: hour tens rtc_ahour[3:0]: hour units
	15	R/W	0	rtc_amin_en	Enable alarm minute 0: disable, minute don't care in alarm comparison 1: enable, alarm set if the minute match
	14:8	R/W	0	rtc_amin	Alarm minute coded in BCD, range is 0~59 rtc_amin[6:4]: minute tens rtc_amin[3:0]: minute units
	7:0	-	-	-	Reserved
RTC_A_DATE: RTC alarm date setting					
1ch	31:22	-	-	-	Reserved
	21	R/W	0	rtc_amonth_en	Enable alarm month 0: disable, month don't care in alarm comparison 1: enable, alarm set if the month match
	20:16	R/W	1	rtc_amonth	Alarm month coded in BCD, range is 1~12 rtc_amonth[4]: month tens rtc_amonth[3:0]: month units
	15	-	-	-	Reserved
	14	R/W	0	rtc_aday_en	Enable alarm day 0: disable, day don't care in alarm comparison 1: enable, alarm set if the day match
	13:8	R/W	1	rtc_aday	Alarm day coded in BCD, range is 1~31 rtc_aday[5:4]: day tens rtc_aday[3:0]: day units
	7:4	-	-	-	Reserved

Index	Bit	R/W	Default	Name	Description
	3	R/W	0	rtc_aweek_en	Enable alarm week day 0: disable, week day don't care in alarm comparison 1: enable, alarm set if the week day match
	2:0	R/W	0	rtc_aweek	Alarm week day coded in BCD, range is 0~6 0: Sunday 1: Monday 2: Tuesday 3: Wednesday 4: Thursday 5: Friday 6: Saturday
RTC_LSE_CFG: RTC LSE configuration					
20h	31:14	-	-	-	Reserved
	13:12	R/W	3h	rtc_lse_cur	Select the Bias Current Ratio to X'tal cir. Block for LSE X'tal pad 3h: Default
	11:10	-	-	-	Reserved
	9:8	R/W	3h	rtc_lse_bgo	Select the Bias Current from Bandgap block for LSE X'tal pad 3h: Default
	7	-	-	-	Reserved
	6:4	R/W	0	rtc_lse_envt	Select the threshold voltage driving level of X'tal inverter for LSE X'tal pad 000: Default
	3:2	-	-	-	Reserved
	1:0	R/W	0	rtc_lse_ed	Select Driving Current for LSE X'tal pad 00: Default

15.4 Functional Description

15.4.1 Programmable Alarm

The programmable alarm function is enabled through the `rtc_alarm_ie[0]` register. The `rtc_alarm_if[0]` is set to 1 if the calendar minutes, hours, date or day match the values programmed in the alarm registers (18h and 1ch registers). Each calendar field can be independently selected through the enable bits (`rtc_amin_en[0]`, `rtc_ahour_en[0]`, `rtc_aweek_en[0]`, ...) of the these registers. The alarm interrupt is enabled through the `rtc_alarm_ie[0]` bit in the 00h register

15.4.2 Periodic Interrupt

The periodic interrupt flag is generated by internal 15-bit 1-second counter. The periodic interrupt function is enabled through the `rtc_period_ie[0]` in the 00h register. The periodic interrupt timer can be programmed to generate 1s, 62.5ms, 31.25ms and 15.63 ms period wake-up time.

16 Independent Watchdog Timer

16.1 Main Features

The independent watchdog (IWDT) is clocked by its own dedicated low-speed clock 37 kHz and thus stays active even if the main clock fails. The IWDT is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints.

- Free-running down counter
- Clocked from an independent oscillator (37 kHz)
- Reset (if watchdog activated) when the down counter value of 000h is reached

16.2 Block Diagram

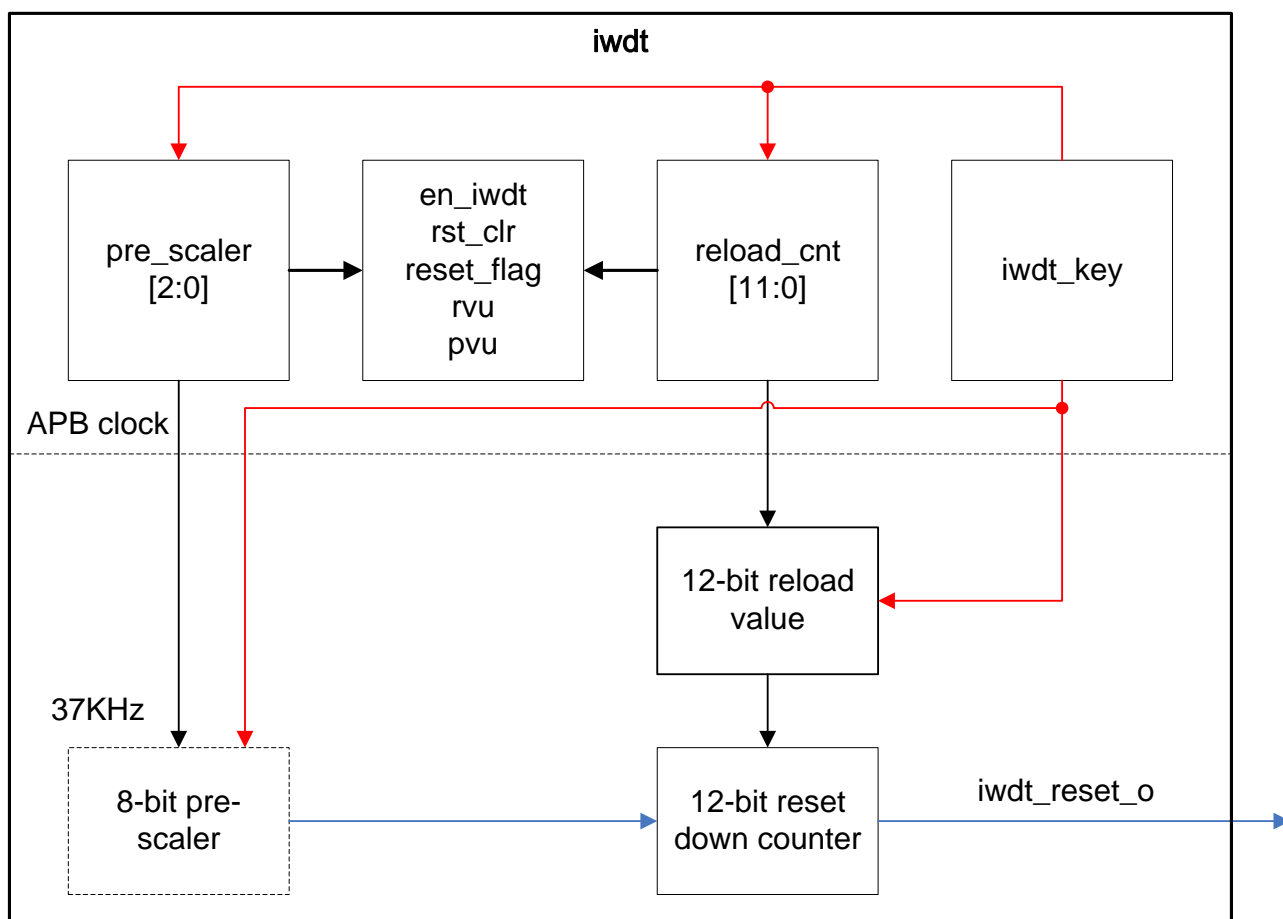


Figure 51 IWDT Block Diagram

16.3 Register Definition

Table 40 IWDT Control Register

Index	Bit	R/W	Default	Name	Description
IWDT_KEY: IWDT key register					
00h	31:16	-	-	-	Reserved
	15:0	W	-	iwdt_key	This write-only register must be written by software at regular intervals to reload reset counter, otherwise the watchdog generates a reset when the reset counter reaches 0. cccc: Enable IWDT (Reset Counter Start) aaaah: Reload reset counter and pre-scaler reset 5555h: Enable access pre-scaler and reset counter registers (Lock open) Note: time interval between this register accesses must be at least 60μs.
IWDT_STS: IWDT status register					
04h	31:8	-	-	-	Reserved
	7	R	0	en_iwdt	0: IWDT enable 1: IWDT disable
	6:5	-	-	-	Reserved
	4	R	0	reload_work	0: Ready for next reset counter reloading 1: Previous reload process is ongoing, invalid for next reset counter reloading
	3	-	-	-	Reserved
	2	-	-	-	Reserved
	1	R	0	rvu	Counter reload value update This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed
	0	R	0	pvu	Pre-scaler value update This bit is set by hardware to indicate that an update of the pre-scaler value is ongoing. It is reset by hardware when the pre-scaler update operation is completed
IWDT_PRE_SCALER: IWDT pre-scaler for down counter					
08h	31:3	-	-	-	Reserved
	2:0	R/W	0	pre_scaler	Clock pre-scaler for down counter This register is write-access protected by iwdt_key[15:0] = 5555h lock open. 0: divide 4 1: divide 8 2: divide 16 3: divide 32 4: divide 64 5: divide 128 6, 7: divide 256 Note: pvu[0] bit must be reset in order to be able to change the pre-scaler[2:0].
IWDT_RELOAD: IWDT down counter reload value					
0ch	31:12	-	-	-	Reserved

Index	Bit	R/W	Default	Name	Description
	11:0	R/W	FFFh	reload_cnt	Down counter reload value This register is write-access protected by iwdt_key[15:0] = 5555h lock open. They are written by software to define the value to be loaded in the watchdog counter each time the value aaaah is written in the iwdt_key register. The reset counter counts down from this value. The timeout period is a function of this value and the clock pre-scaler. The rvu[0] bit must be reset in order to be able to change the reload value.

16.4 Functional Description

When IWDT is started by writing the value cccch in the KEY register (iwdt_key[15:0]), the counter starts counting down from the reset value of fffh. When it reached the end of count value (000h) a reset signal is generated (IWDT Reset). Whenever the key value aaaah is written in the iwdt_key[15:0] register, the reload_cnt[11:0] value is reloaded in the counter and Watchdog reset is prevented.

Write access to the pre_scaler[2:0] and reload_cnt[11:0] registers is protected. To modify them, the code 5555h in the iwdt_key[15:0] must first be written. A write access to this register with a different value will break the sequence and register access will be protected again. This implies that it is the case of the reload operation (writing aaaah). Status registers (rvu[0] and pvu[0]) are available to indicate that an update of the pre-scaler[2:0] or the down-counter reload value is ongoing.

The minimum/maximum timeout period at 37kHz is show as below.

Table 41 IWDT Min./Max. Timeout Period at 37 kHz Clock Input

pre_scaler[2:0]	Minimum timeout (ms) reload_cnt[11:0]=000h	Maximum timeout (ms) reload_cnt[11:0]=fffh
0	0.108	442.810
1	0.216	885.622
2	0.432	1,771.243
3	0.865	3,542.456
4	1.730	7,084.973
5	3.459	14,169.946
6 or 7	6.919	28,339.892

17 Window Watchdog Timer

17.1 Main Features

The “Window Watchdog Timer” (WWDT) is used to detect the occurrence of a software fault which causes the application program to abandon its normal sequence. The WWDT generates an MCU reset, unless the program refreshes the down counter before its bit-6 cleared. An MCU reset is also generated if the down counter is refreshed before it has reached the window register value. This means that the down counter must be refreshed in a limited window.

- Programmable free-running down counter
- Conditional reset:
 - Reset (if WWDT enable) when the down counter value becomes less than 40h
 - Reset (if WWDT enable) if the down counter is reloaded outside the window
- Early wakeup interrupt (EWI): triggered (if WWDT enable) when the down counter is equal to 40h. This function can be used to reload the counter and prevent WWDT reset.

17.2 Block Diagram

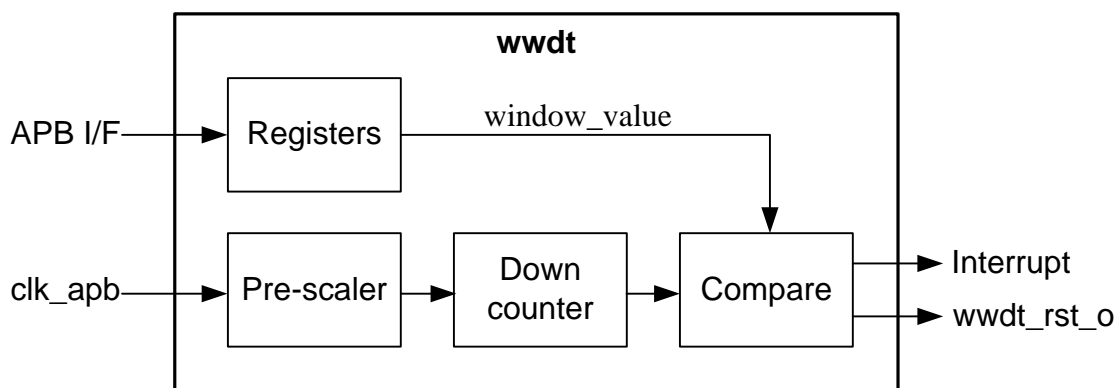


Figure 52 WWDT Block Diagram

17.3 Register Definition

Table 42 WWDT Control Register

Index	Bit	R/W	Default	Name	Description
WWDT_CR1: WWDT control register 1					
00h	31	R/W	0	wwdt_enable	Enable WWDT function
	30:7	-	-	-	Reserved
	6:0	R/W	7fh	down_cnt_time	Reset down counter (valid only wwdt_en = 1) Reset active when down_cnt_time = 3fh.
WWDT_CR2: WWDT control register 2					
04h	31:13	-	-	-	Reserved
	12	R/W	0	early_int_en	Early interrupt enable

Index	Bit	R/W	Default	Name	Description
					0: disable early wwdt interrupt 1: enable early wwdt interrupt
	11:8	R/W	0	time_base	Module clock (clk_apb) divided by 4096 and divided by 2**{time_base} Note: time_base should be set in 0~8.
	7	-	-	-	Reserved
	6:0	R/W	7fh	window_value	Window value to be compared to reset down counter
WWDT_INT_F: WWDT early interrupt flag					
08h	31:1	-	-	-	Reserved
	0	R/W1c	0	early_int_flag	Early interrupt flag (Reset down counter = 40h)

R/W1C: read & write one clear

17.4 Functional Description

If WWDT is enabled and when the down counter rolls over 40h to 3fh, it generates a MCU reset. If the software reloads the counter while the value is greater than the value stored in the window register, then a reset is generated. The application program must write in the down_cnt_time register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the down counter value is lower than window register value (window_value). Therefore, the value to be stored in the 00h (wwdt_en and down_cnt_time) register must be between ffh and c0h.

- Enable WWDT: The Window watch dog timer is always disabled after reset. It is enabled by setting the (wwdt_en = 1) register. It will not be disabled again once it's enabled except by a reset.
- Down counter control: When WWDT is enabled, the down_cnt_time[6] must be set to prevent generating an immediate reset. The down_cnt_time[5:0] bits contain the number of increments which represents the time delay before the WWDT produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of pre-scaler when writing to the *time_base* register. The *window_value* register contains the high limit of the window. To prevent a reset, the down counter must be reloaded when its value is lower than the window register value and greater than 3fh after setting WWDT enable (wwdt_en = 1). Another way to reload the counter is to use the early interrupt. When the down counter reached the value 40h, this interrupt is generated and the corresponding interrupt service routine can be used to reload the down counter to prevent WWDT reset. The interrupt is cleared by setting (early_int_flag) register. A software reset also can be generated by clearing down_cnt_time[6] when WWDT is active (wwdt_en = 1).
- Program the Watchdog Timeout: The timing diagram of WWDT is depicted as Figure 53. Below is the formula to calculate the time out value:

$$T_{WWDT} = T_{clk_apb} \times 4096 \times 2^{time_base} \times (down_cnt_time[5:0] + 1) \text{ (ms)}$$

Min./Max timeout value at 24.00MHz is shown as Table 44.

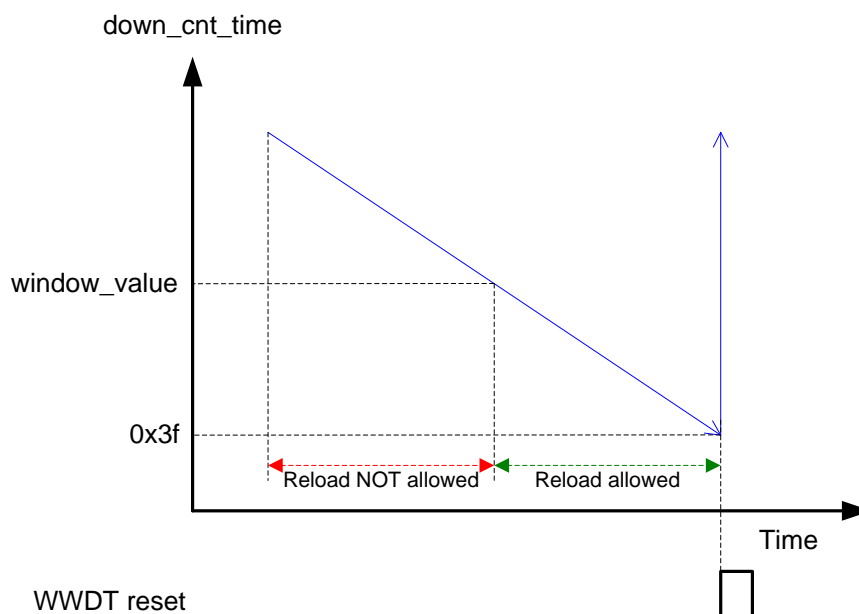


Figure 53 Timing Diagram of WWDT

Table 43 WWDT Min./Max Timeout Value at 24.00MHz

time_base	Minimun Timeout (μs) down_cnt_time = 40	Maximum Timeout (ms) down_cnt_time = 7f
0	170.67	10.92
1	341.33	21.85
2	682.67	43.69
3	1365.33	87.38
4	2730.66	174.76
5	5461.32	349.52
6	10922.64	699.04
7	21845.28	1398.08
8	43690.56	2796.16

18 PWM

18.1 Main Features

The “Pulse Width Modulation” (PWM) module

- Four independent period PWM
 - independent user-defined period and duty cycle
 - Duty range from 0/65536 to 65535/65536
- Support double buffer function for PWM duty cycle
- One interrupt output

Block Diagram

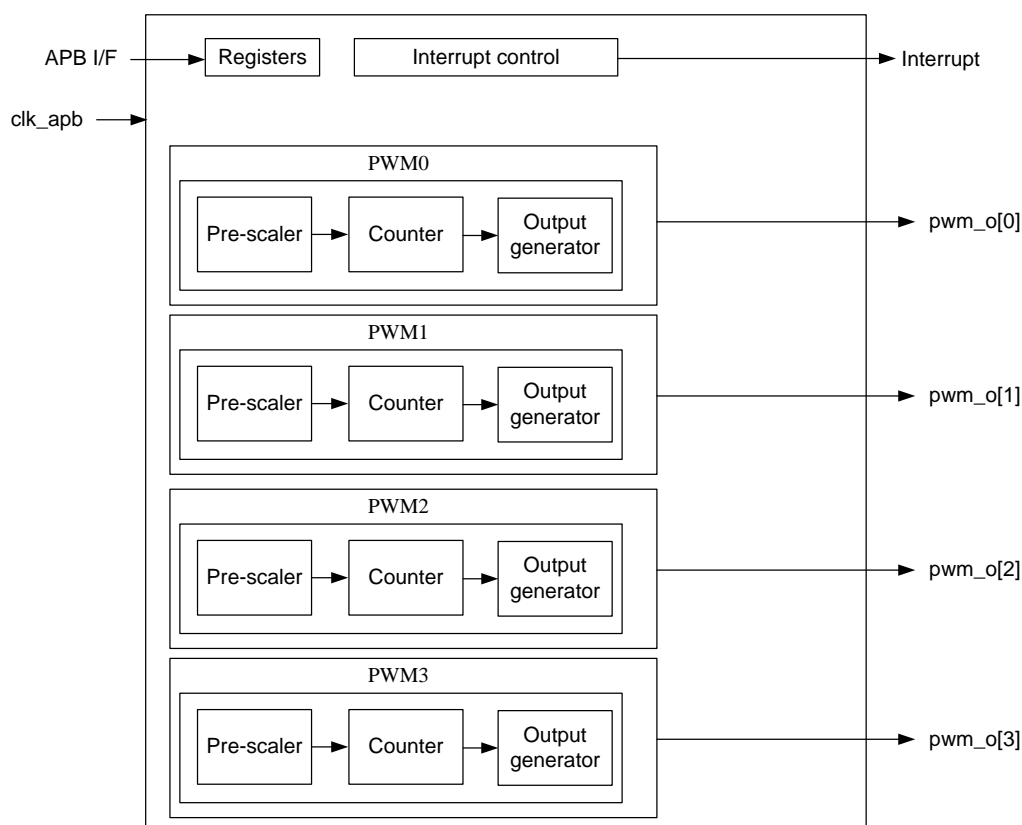


Figure 54 PWM Block Diagram

18.2 Register Definition

Table 44 PWM Control Register

Index	Bit	R/W	Default	Name	Description
PWM_OEN: PWM output enable & interrupt					
00h	31:8	R/W	0	PWMO_EN	PWMx output enable bit 0: Disable PWMx output 1: Enable PWMx output
	27:24	R/W	0	PWMIE	PWMx interrupt enable bit 0: Disable PWMx interrupt 1: Enable PWMx interrupt
	3:0	R/W	0	PWMIF	PWMx interrupt flag, write 1 to clear this bit. 0: No Event of PWMx interrupt 1: Event of PWMx interrupt
PWM_CR0: PWM control register 0					
04h	31	R/W	0	PWMS0	0: Stop PWM0 function (default) 1: Start PWM0 function
	30:4	-	-	-	Reserved
	3	R/W	0	PWMDDB0	Double buffer function 0: Apply new PWMDV0/PWMCV0 immediately 1: Apply new PWMDV0/PWMCV0 at next PWM cycle
	2:0	R/W	0	PWMCK0	Pre-scaler for system clock as PWM0 source clock 000: /2 ¹¹ (default) 001: /2 ⁹ 010: /2 ⁷ 011: /2 ⁵ 100: /2 ³ 101: /2 ² 110: /2 111: /1
PWM_PWMCV0: PWM Period setting for PWM0					
08h	31:16	-	-	-	Reserved
	15:0	R/W	0	PWMCV0	Period setting for PWM0
PWM_PWMDV0: PWM Duty setting for PWM0					
0Ch	31:16	-	-	-	Reserved
	15:0	R/W	0	PWMDV0	Duty setting for PWM0 output
PWM_CR1: PWM control register 1					
10h	31	R/W	0	PWMS1	0: Stop PWM1 function (default) 1: Start PWM1 function
	30:4	-	-	-	Reserved
	3	R/W	0	PWMDDB1	Double buffer function 0: Apply new PWMDV1/PWMCV1 immediately 1: Apply new PWMDV1/PWMCV1 at next PWM cycle
	2:0	R/W	0	PWMCK1	Pre-scaler for system clock as PWM1 source clock 000: /2 ¹¹ (default) 001: /2 ⁹ 010: /2 ⁷

Index	Bit	R/W	Default	Name	Description
					011: /2 ⁵ 100: /2 ³ 101: /2 ² 110: /2 111: /1
PWM_PWMCV1: PWM Period setting for PWM1					
14h	31:16	-	-	-	Reserved
	15:0	R/W	0	PWMCV1	Period setting for PWM1
PWM_PWMDV1: PWM Duty setting for PWM1					
18h	31:16	-	-	-	Reserved
	15:0	R/W	0	PWMDV1	Duty setting for PWM1 output
PWM_CR2: PWM control register 2					
1Ch	31	R/W	0	PWMS2	0: Stop PWM2 function (default) 1: Start PWM2 function
	30:4	-	-	-	Reserved
	3	R/W	0	PWMDV2	Double buffer function 0: Apply new PWMDV2/PWMCV2 immediately 1: Apply new PWMDV2/PWMCV2 at next PWM cycle
	2:0	R/W	0	PWMCK2	Pre-scaler for system clock as PWM2 source clock 000: /2 ¹¹ (default) 001: /2 ⁹ 010: /2 ⁷ 011: /2 ⁵ 100: /2 ³ 101: /2 ² 110: /2 111: /1
PWM_PWMCV2: PWM Period setting for PWM2					
20h	31:16	-	-	-	Reserved
	15:0	R/W	0	PWMCV2	Period setting for PWM2
PWM_PWMDV2: PWM Duty setting for PWM2					
24h	31:16	-	-	-	Reserved
	15:0	R/W	0	PWMDV2	Duty setting for PWM2 output
PWM_CR3: PWM control register 3					
28h	31	R/W	0	PWMS3	0: Stop PWM3 function (default) 1: Start PWM3 function
	30:4	-	-	-	Reserved
	3	R/W	0	PWMDV3	Double buffer function 0: Apply new PWMDV3/PWMCV3 immediately 1: Apply new PWMDV3/PWMCV3 at next PWM cycle
	2:0	R/W	0	PWMCK3	Pre-scaler for system clock as PWM3 source clock 000: /2 ¹¹ (default) 001: /2 ⁹ 010: /2 ⁷ 011: /2 ⁵ 100: /2 ³ 101: /2 ² 110: /2

Index	Bit	R/W	Default	Name	Description
					111: /1
PWM_PWMCV3: PWM Period setting for PWM3					
2Ch	31:16	-	-	-	Reserved
	15:0	R/W	0	PWMCV3	Period setting for PWM3
PWM_PWMDV3: PWM Duty setting for PWM3					
30h	31:16	-	-	-	Reserved
	15:0	R/W	0	PWMDV3	Duty setting for PWM3 output

18.3 Functional Description

18.3.1 Independent PWM

Duty range: 0/65536 to 65535/65536

$PWMDVx \leq PWMCVx$, PWMx outputs low level

If $PWMDVx > PWMCVx$, PWMx outputs high level

In normal mode:

$PWMCVx \geq 1$ and $PWMDVx \geq 1$

$PWMDVx \leq PWMCVx$

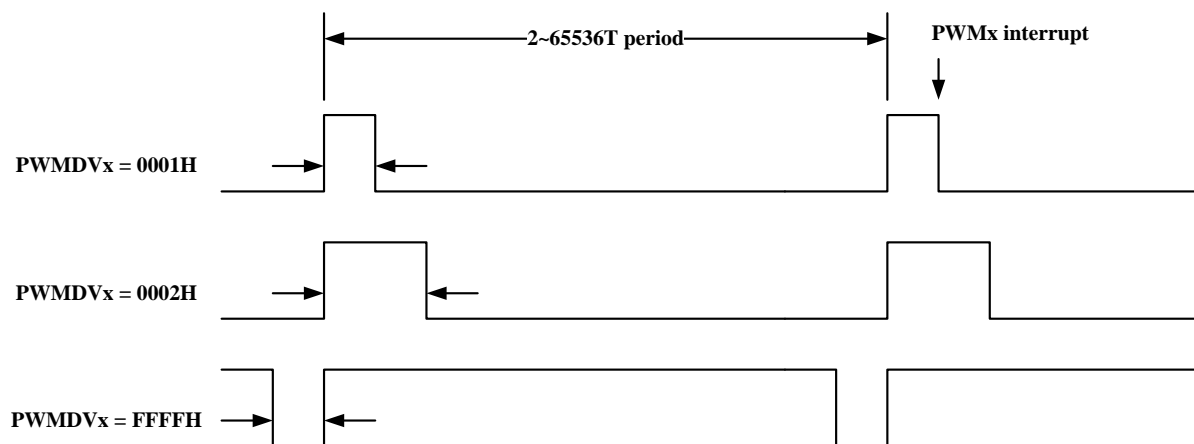


Figure 55 PWM0~3 Function

19 Timer

19.1 Main Features

The general-purpose timers consist of a 16-bit counter driven by a programmable pre-scaler, they are designed to count cycles of the peripheral clock (PCLK) or an external clock, and may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM) based on match registers. The timers can also generate interrupts or DMA requests as system requirements.

- A 16 bit timer/counter with a programmable 16-bit pre-scaler
- Counter or timer operation
- Two 16-bit capture channels per timer, includes PWM input detection
- Synchronization circuit to control the timer with external signals and other timers
- Interrupt/DMA generation on the following events:
 - Input capture
 - Output match
- Four 16-bit Match Registers that allow:
 - Continuous operation with optional interrupt generation on match
 - Stop timer on match with optional interrupt generation
 - Reset timer on match with optional interrupt generation
- Two external outputs corresponding to match registers, with the following capabilities:
 - Set low on match
 - Set high on match
 - Toggle on match
- The combination of paired match registers generates PWM output

19.2 Block Diagram

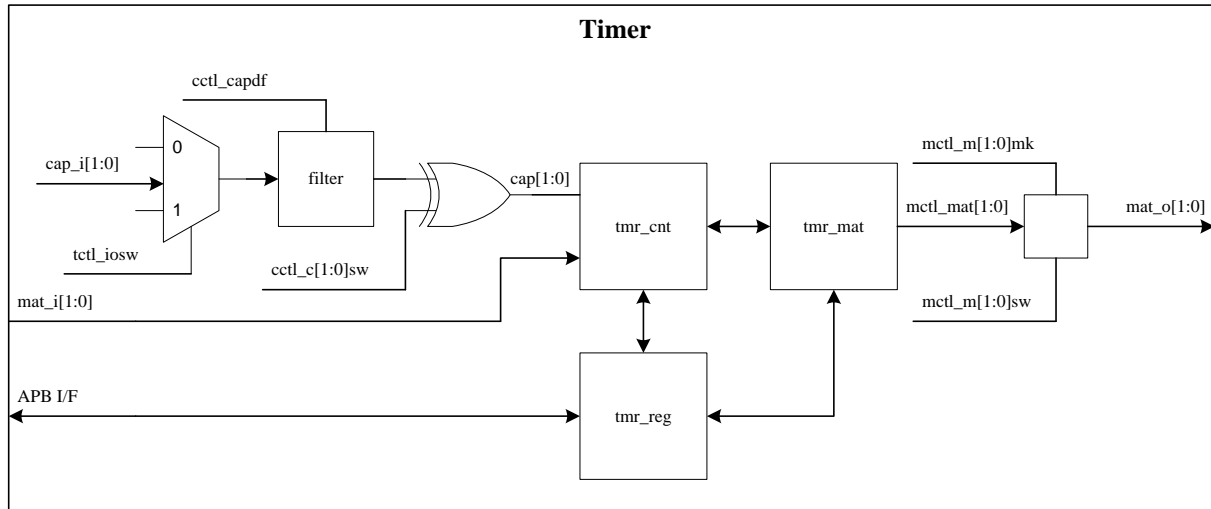


Figure 56 Timer Module Block Diagram

Notes:

1. tmr[n].mat0x connects to tmr[n+1].mat1x and output pads
2. capix connect to input pads
3. If tctl_iosw = '0', Capture input = cap_i[1:0] = TMR_CH[1:0],
Match output = mat_o[1:0] = TMR_CH[3:2]
If tctl_iosw = '1', Capture input = cap_i[1:0] = TMR_CH[3:2],
Match output = mat_o[1:0] = TMR_CH[1:0]

19.3 Register Definition

Table 45 Timer Control Register

Index	Bit	R/W	Default	Name	Description
TIMER_TCTL: Timer control register					
00h	15	R/W	0	tctl_en	Timer enable
	14:12	-	-	-	Reserved
	11:10	R/W	0	tctl_clkssel	timer clock select 00: clk_apb 01: clk_msi 10: clk_hsi 11: tmr1_cap0 Note: Timer1 could use b'00~b'11 clock source, Timer0/2 only use b'00 APB clock source.
	9	R/W	0	tctl_matisw	mati source swap 0: mati = original match input 1: mati[1:0] = {lse_clk, lsi_clk}
	8	R/W	0	tctl_matco	mat0/mat1 control off 0: write mctl changes mat1/mat0 value accordingly 1: write mctl doesn't affect mat1/mat0 value

Index	Bit	R/W	Default	Name	Description
	7	R/W	0	tctl_de	debug enable 0: Timer hold during debug mode 1: Timer operates normally during debug mode
	6	R/W	0	tctl_iosw	I/O swap 0: no operation 1: swap capix/matox I/O definition and direction
	5:4	R/W	0	tctl_sel	counter input select 00: cap0 01: cap1 10: mati0 11: mati1
	3:2	R/W	0	tctl_mode	counter/timer mode 00: timer 01: rising edge counter 10: falling edge counter 11: both edge counter
	1	R/W	0	tctl_rst	counter reset 0: no action on counter 1: reset {tcnt, pcnt}
	0	R/W	0	tctl_st	timer start/stop control 0: stop 1: start
TIMER_TCNT: Timer counter register					
04h	15:0	R/W	0	Tcnt	Timer counter TCNT incremented 1 until PCNT overflow then PCNT clear to 0, see below illustration
TIMER_PSCL: Timer prescaler setting					
08h	15:0	R/W	0	pscl	Prescaler value PSCL, see below illustration
TIMER_PCNT: prescaler counter register					
0ch	15:0	R/W	0	Pcnt	Prescaler counter PCNT incremented 1 until PSCL value then PCNT clear to 0, see below illustration
TIMER_CCTL: filter setting for capi					
10h	15:12	-	-	-	Reserved
	11:10	R/W	0	cctl_capdf	digital filter setting for capi0 & capi1 00: no filter (1 clock) 01: 4 clocks 10: 16 clocks 11: 64 clocks
	9	R/W	0	cctl_c1sw	invert capi1 for cap1
	8	R/W	0	cctl_c0sw	invert capi0 for cap0
	7:4	-	-	-	Reserved
	3	R/W	0	cctl_cap1f	capture enable on cap1 falling edge (stored in cap1f)
	2	R/W	0	cctl_cap1r	capture enable on cap1 rising edge (stored in cap1r)
	1	R/W	0	cctl_cap0f	capture enable on cap0 falling edge (stored in cap0f)
	0	R/W	0	cctl_cap0r	capture enable on cap0 rising edge (stored in cap0r)

Index	Bit	R/W	Default	Name	Description
TIMER_CICTL: capi control					
14h	15:12	-	-	-	Reserved
	11:10	R/W	0	cictl_c1actx	cap1 extended operation on counter 00: no operation 01: hold counter on cap1 low level 10: reset counter on cap1 falling edge 11: reserved
	9:8	R/W	0	cictl_c0actx	cap0 extended operation on counter
	7:6	R/W	0	cictl_m1act	mati1 operation on counter (Note-3) 00: no operation 01: hold counter on mati1 high level 10: reset counter on mati1 rising edge 11: trigger counter on mati1 rising edge (set tctl_st)
	5:4	R/W	0	cictl_m0act	mati0 operation on counter
	3:2	R/W	0	cictl_c1act	cap1 operation on counter 00: no operation 01: hold counter on cap1 high level 10: reset counter on cap1 rising edge 11: trigger counter on cap1 rising edge (set tctl_st)
	1:0	R/W	0	cictl_c0act	cap0 operation on counter
TIMER_MCTL: mat control					
18h	15:14	R/W	0	mctl_m1mk	matm1 output mask (Note-4) 00: same as mat1 01: reserved 10: clear matm1 if cap0 = 1 11: clear matm1 if cap1 = 1
	13:12	R/W	0	mctl_m0mk	matm0 output mask
	11	R/W	0	mctl_mat1	mat1 value
	10	R/W	0	mctl_mat0	mat0 value
	9	R/W	0	mctl_m1sw	invert mat1 for mato1
	8	R/W	0	mctl_m0sw	invert mat0 for mato0
	7:6	R/W	0	mctl_mat1b	match control for mat1b (Note-5) 00: no operation 01: clear the corresponding external match bit to 0 10: set the corresponding external match bit to 1 11: toggle the corresponding external match bit Note: if mctl_m1br = '1' and mat1a > mat1b, the output is toggled for 01/10 settings
	5:4	R/W	0	mctl_mat1a	match control for mat1a
	3:2	R/W	0	mctl_mat0b	match control for mat0b
	1:0	R/W	0	mctl_mat0a	match control for mat0a
TIMER_MOCTL: mo control					
1ch	15:8	-	-	-	Reserved
	7	R/W	0	mocltl_m1bs	mat1b stop counter (clear tctl_st)
	6	R/W	0	mocltl_m1br	mat1b reset counter
	5	R/W	0	mocltl_m1as	mat1a stop counter
	4	R/W	0	mocltl_m1ar	mat1a reset counter

Index	Bit	R/W	Default	Name	Description
	3	R/W	0	mocctl_m0bs	mat0b stop counter
	2	R/W	0	mocctl_m0br	mat0b reset counter
	1	R/W	0	mocctl_m0as	mat0a stop counter
	0	R/W	0	mocctl_m0ar	mat0a reset counter
TIMER_MAT0A: mat0a/ cap0r register					
30h	15:0	R/W	0	mat0a (cap0r)	match mat0a register for output match mode capture cap0r register for input capture mode
TIMER_MAT0B: mat0b/ cap0f register					
34h	15:0	R/W	0	mat0b (cap0f)	match mat0b register
TIMER_MAT1A: mat1a/ cap1r register					
38h	15:0	R/W	0	mat1a (cap1r)	match mat1a register
TIMER_MAT1B: mat1b/ cap1f register					
3ch	15:0	R/W	0	mat1b (cap1f)	match mat1b register
TIMER_OFIF: overflow and interrupt flag					
40h	15	R/W1c	0	of_tcmt	overflow flag for tcmt (tcmt loop from ffffh to 0000h) (Note-6)
	14:12	-	-	-	Reserved
	11	R/W1c	0	of_cap1f	overflow flag for cap1f (twice or more interrupts occur)
	10	R/W1c	0	of_cap1r	overflow flag for cap1r
	9	R/W1c	0	of_cap0f	overflow flag for cap0f
	8	R/W1c	0	of_cap0r	overflow flag for cap0r
	7	R/W1c	0	if_mat1b	interrupt flag for mat1b
	6	R/W1c	0	if_mat1a	interrupt flag for mat1a
	5	R/W1c	0	if_mat0b	interrupt flag for mat0b
	4	R/W1c	0	if_mat0a	interrupt flag for mat0a
	3	R/W1c	0	if_cap1f	interrupt flag for cap1f
	2	R/W1c	0	if_cap1r	interrupt flag for cap1r
	1	R/W1c	0	if_cap0f	interrupt flag for cap0f
	0	R/W1c	0	if_cap0r	interrupt flag for cap0r
TIMER_IE: interrupt enable					
44h	15:8	-	-	-	Reserved
	7	R/W	0	ie_mat1b	interrupt enable for mat1b
	6	R/W	0	ie_mat1a	interrupt enable for mat1a
	5	R/W	0	ie_mat0b	interrupt enable for mat0b
	4	R/W	0	ie_mat0a	interrupt enable for mat0a
	3	R/W	0	ie_cap1f	interrupt enable for cap1f
	2	R/W	0	ie_cap1r	interrupt enable for cap1r
	1	R/W	0	ie_cap0f	interrupt enable for cap0f
	0	R/W	0	ie_cap0r	interrupt enable for cap0r
TIMER_RF: DMA request flag					
48h	15:8	-	-	-	Reserved
	7	R/W1c	0	rf_mat1b	DMA request flag for mat1b (Note-7) (Note-8)
	6	R/W1c	0	rf_mat1a	DMA request flag for mat1a
	5	R/W1c	0	rf_mat0b	DMA request flag for mat0b
	4	R/W1c	0	rf_mat0a	DMA request flag for mat0a
	3	R/W1c	0	rf_cap1f	DMA request flag for cap1f
	2	R/W1c	0	rf_cap1r	DMA request flag for cap1r
	1	R/W1c	0	rf_cap0f	DMA request flag for cap0f
	0	R/W1c	0	rf_cap0r	DMA request flag for cap0r

Index	Bit	R/W	Default	Name	Description
TIMER_RE: DMA request enable					
4ch	15:8	-	-	-	Reserved
	7	R/W	0	re_mat1b	DMA request enable for mat1b
	6	R/W	0	re_mat1a	DMA request enable for mat1a
	5	R/W	0	re_mat0b	DMA request enable for mat0b
	4	R/W	0	re_mat0a	DMA request enable for mat0a
	3	R/W	0	re_cap1f	DMA request enable for cap1f
	2	R/W	0	re_cap1r	DMA request enable for cap1r
	1	R/W	0	re_cap0f	DMA request enable for cap0f
	0	R/W	0	re_cap0r	DMA request enable for cap0r

R/W1C: read & write one clear

Note-3: reset counter has higher priority than hold counter

Note-4: function is ineffective if tctl_en or tctl_st is off

Note-5: mctl_matxa has higher priority than mctl_matxb if both events happen simultaneously

Note-6:

all if_xxx/of_xxx need to write '1' to clear

all if_xxx/of_xxx are cleared if tctl_en=0

Note-7:

all rf_xxx need to write '1' to clear

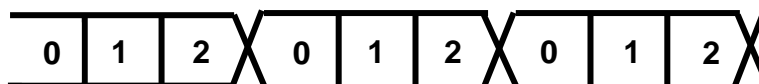
all rf_xxx are cleared if tctl_en=0

Note-8: reg_rf can be used as event indicator with reg_re = ffh and DMA channel disabled

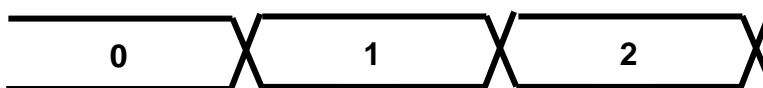
**Clock Source Input
(APB/HSI/MSI/HSE)**

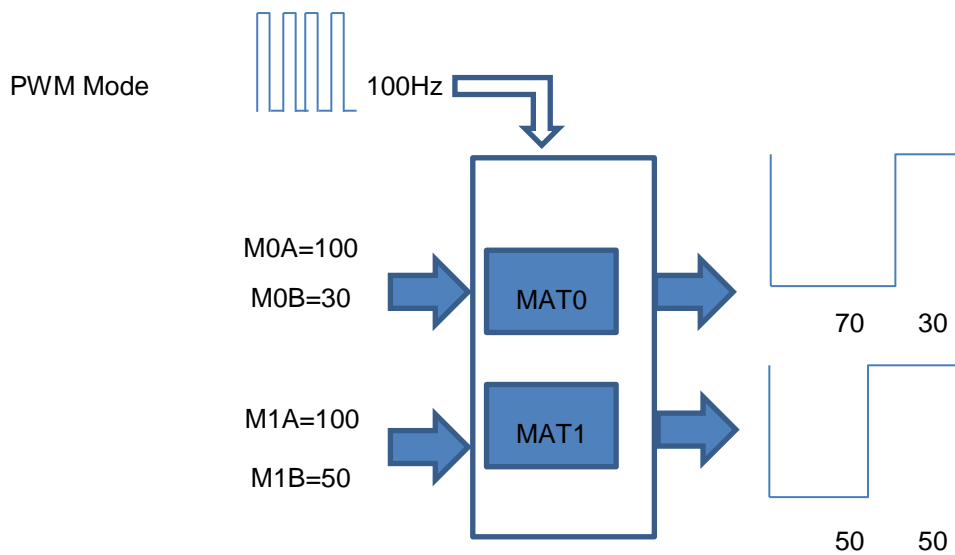
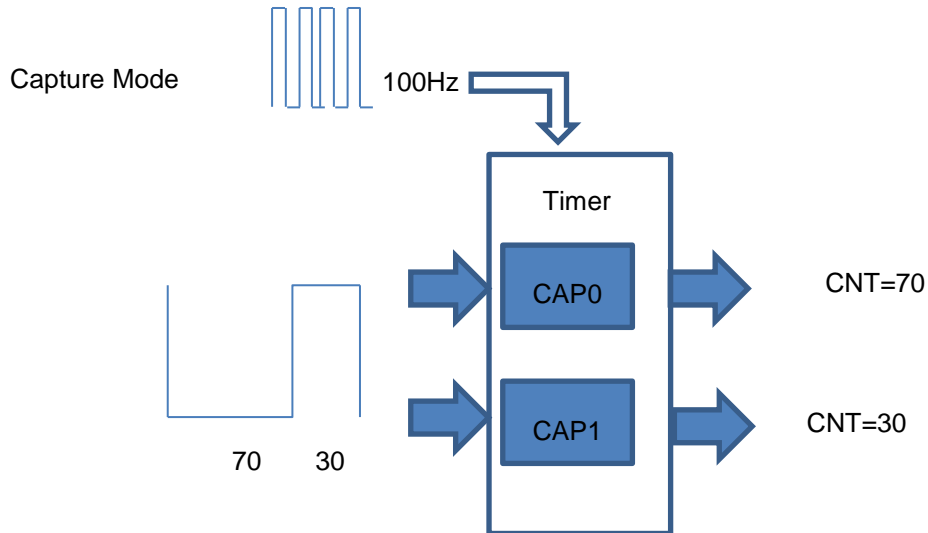


**Prescaler Counter
(PCNT) @Setting
PSCL=2**



**Timer Counter
(TCNT)**





19.4 Functional Description

19.4.1 Multiple Capture and Match Pins

System can select from multiple pins for the capture or match functions in the GPIO registers. When more than one pin is selected for match outputs, all such pins are driven by MATOx of timer individually. When more than one pin is selected for a capture inputs, different pins are filtered and processed by the internal logic of timer with corresponding capture settings. Note that match conditions may be used internally without the use of a device pin.

19.4.2 Interrupts

The source of timer interrupt consists of 4 bits for the match interrupts and 4 bits for the capture interrupts. If an event of capture/match defined in capture/match registers is generated and corresponding interrupt enable register in IE_REG is set, then the corresponding bit in the IF_REG will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IF_REG bit will reset the interrupt. Writing a zero has no effect. The timer interrupt is issued only interrupt flag in IF_REG is under non-zero state. Additionally, IF_REG contains overflow flags of capture event and main counter. If capture event happens twice or more without flag clearing by system, it means the captured data is overwritten and must be given up, and the OF_CAPxx flag is set. In normal timer operation, main counter should not turn around from 0xffff to 0x0000, else it may result in illegal capture data or match output, and the OF_TCNT flag is set. System should step in timer error handling by resolution tuning or fix event processing while overflow flag OF_CAPxx or OF_TCNT flag is set.

19.4.3 DMA

The DMA function contains DMA request enable register and DMA request flag register for corresponding timer capture or match events. The corresponding RF_REG is set and timer DMA request is issued to DMA circuit to start DMA operation if the corresponding DMA request enable register is set and corresponding timer event occurs. Writing a logic one to the corresponding RF_REG bit will clear the RF_xxx flag and terminate DMA operation. Writing a zero has no effect. The RF_REG is also cleared after DMA operation normally completed. Note that only single DMA channel is applied for single timer, so single bit enable of RE_REG is reasonable for normal DMA application.

19.4.4 Count Control

The count control includes TCTL, TCNT, PSCL and PCNT. Register TCTL_EN controls whole timer function, The IF_xxx and RF_xxx are all cleared and {TCNT, PCNT} is held discarding TCTL_ST setting if TCTL_EN is cleared. Register TCTL_ST controls timer running of {TCNT, PCNT}, this bit is also controlled by capture/match event generated by H/W. Register TCTL_MODE is used to select between timer and counter mode, and in counter mode to select the pin and edge(s) for counting. When counter mode is chosen as a mode of operation, the counter input (selected by the TCTL_SEL) is sampled on every rising edge of the PCLK clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected counter input. Only if the identified event occurs and the event corresponds to the one selected by TCTL_MODE register, will the {TCNT, PCNT} register be incremented. Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the PCLK clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input cannot exceed one quarter of the PCLK clock. Consequently, duration of the high/low levels on the same CAP input in this case cannot be shorter than $1/(2 \times PCLK)$.

The TCNT register work with a pre-scaling PSCL register, the TCNT will increment if PCNT equals PSCL, and PCNT increments for every PCLK in timer mode and for every count tick in counter mode, the PCNT register will be reset while PCNT equals PSCL setting. Actual capture/match function is work with {TCNT, PCNT}, controls {TCNT, PCNT} to hold, reset or count up. All the operations are only dependent to TCNT value with a full PCNT loop.

19.4.5 Capture Control

Each capture register is associated with a device pin plus signal edge and may be loaded with the timer counter value when a specified event occurs on that pin. The settings in the capture enable register (CCTL_CAPxx) determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated pin, the falling edge, or on both edges.

The capture control register (CCTL) is used to control whether one of the four capture registers (CAP0R, CAP0F, CAP1R & CAP1F, shared with match function) is loaded with the value in the timer counter when the capture event occurs, and whether an digital filter with 2/4/8 clocks is applied to filter noises coming from external devices. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. All the two capture inputs with rising/falling edge detection can work at the same time.

The capture inputs can also hold/reset counter by their signal levels or signal edges as description of CICTL_CxACTX and CICTL_CxACT, match inputs (coming from prior timer output) have the similar but simpler capabilities as description of CICTL_MxACT.

The interrupt flag is generated if CCTL_CAPxx is enabled, the corresponding IE_CAPxx is set and capture event is occurred.

19.4.6 Match Control

The match register values, including MAT0A, MAT0B, MAT1A and MAT1B are continuously compared to the timer counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the timer counter, or stop the timer. Actions are controlled by the settings in the MOCTL_Mxxx register.

The MCTL_MATxx register is used to control what operations are performed on timer MATOx output when one of the match registers matches the timer counter. One of the following operation can be performed, including clear MATOx, set MATOx or toggle MATOx. Capture input may be used to control intermediate stage of MATOx by clear it when capture input is high, using the MCTL_MxMK register.

The interrupt flag is generated if TCNT equals match register MATxx, and the corresponding IE_MATxx is set.

20 ADC

20.1 Main Features

- Operating voltage:
 - VDA: 1.8V~3.6V @ VDA reference
 - VDA: 2.3V~3.6V @ VBG:1.0V reference
 - The best ENOB (>10bit) @VDA:3.3V & VDA 3.3V reference
- 16-channel 12-bit ADC control interface
- Support DMA
- Conversion mode
 - when ADC clock \leq APB clock
 - ◆ Single conversion of a sequence
 - ◆ Continuous conversion of a sequence
 - when ADC clock $>$ APB clock
 - ◆ Single conversion of one channel
 - ◆ Continuous conversion of one channel
- Analog Watchdog

20.2 Block Diagram

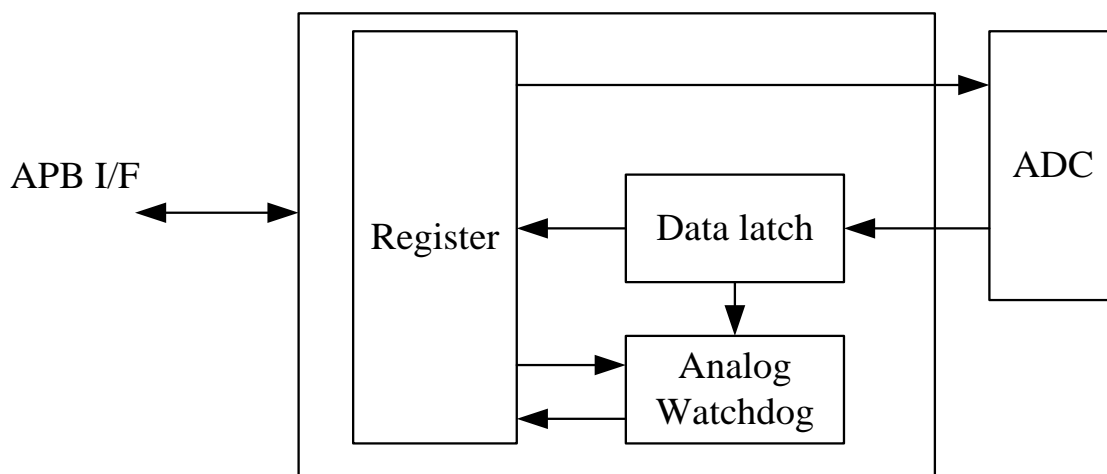


Figure 57 ADC Block Diagram

20.3 Register Definition

Table 46 ADC Control Register

Index	Bit	R/W	Default	Name	Description
ADC_CFG1: ADC configuration					
00h	31:13	-	-	-	Reserved

Index	Bit	R/W	Default	Name	Description
	12:11	R/W	10	ADC_REF_SEL	ADC Reference Voltage Select 00: VDDA 01: V1P0 10: External Vref 11: External Vref *When ADC_REF_SEL is set to 2'b01 (VREF selects V1P0), the IC working current will increase by 150μA, so when saving power, please set ADC_REF_SEL to 2'b00.
	10	R/W	0	ADC_SLOW	0: Disable, When ADC clock ≤ APB clock 1: Enable, When ADC clock > APB clock
	9	R/W	0	ADC_EOCIE	0: Disable ADC EOC interrupt 1: Enable ADC EOC interrupt
	8	R/W	0	ADC_WDIE	0: Disable ADC Watchdog interrupt 1: Enable ADC Watchdog interrupt
	7	R/W	0	ADC_DMA	Direct memory access mode 0: Disable DMA mode 1: Enable DMA mode
	6	R/W	0	ADC_AWD	1: Enable ADC watchdog 0: Disable ADC watchdog
	5	R/W	0	ADC_CONT	Single/continuous conversion of a sequence 0: Single conversion of a sequence 1: Continuous conversion of a sequence
	4:2	R/W	0	ADC_SLT_CLK	Select ADC clock: 000: 16MHz 001: 16MHz /2 010: 16MHz /4 011: 16MHz /8 100: 16MHz /16 101: 16MHz /32 110: 16MHz /64 111: 16MHz /128 Note: The software can program the register field only when ADC PD disable or write "1" to ADC_STOP
	1:0	R/W	0	ADC_DO	AD data output format: (0,0): 12-bit (0,1): 10-bit (1,x): 8-bit
ADC_DO_IDX_F: ADC data & index enable flag					
04h	31:16	R	0	ADC_DINDEX	ADC Data channel index 1: enable 0: disable bit16: ADC channel 0 Bit31: ADC channel 15
	11:0	R	0	ADC_DATA	ADC convert data
ADC_FLAG: AWD and EOC flag					
08h	31:2	-	-	-	Reserved
	1	R/W1c	0	AWD	Analog watchdog flag It is cleared by software writing 1 to it.
	0	R/W1	0	EOC	End of conversion flag

Index	Bit	R/W	Default	Name	Description
		c			It is cleared by software writing 1 to it or by reading the ADC_DATA register.
ADC_ADCCR: ADC control bits					
0Ch	31:18	-	-	-	Reserved
	17	R/W	0	ADC_START	ADC control start Note: The software is allowed to set ADC_START only when ADC PD disable
	16	R/W	0	ADC_STOP	ADC control stop, write "1" to stop
	15:0	R/W	0	ADC_CH	AD channel selection example 0000_0000_0000_0001: channel 0 enable 0000_0000_0000_0010: channel 1 enable 0000_0000_0000_1111: channel 0~3 enable Note: When ADC_SLOW = 1, ADC_CH only set one channel, and ADC_CH change other channel only when write "1" to ADC_STOP
ADC_AWDCH: AWD channel configuration					
10h	31:5	-	-	-	Reserved
	4	R/W	0	AWDSGL	Enable the watchdog on a single channel or on all channel 0: Analog watchdog enabled on all channel 1: Analog watchdog enabled on single channel
	3:0	R/W	0	AWD_CH	Analog watchdog channel selection 0000: channel 0 0001: channel 1 0010: channel 2 1111: channel 15
ADC_THRHD: AWD threshold setting					
14h	31:28	-	-	-	Reserved
	27:16	R/W	0	AWD_HT	Analog watchdog higher threshold
	15:12	-	-	-	Reserved
	11:0	R/W	0	AWD_LT	Analog watchdog lower threshold
ADC_CFG2: ADC standby and VCM setting					
18h	31:3	-	-	-	Reserved
	2	R/W	0	ADC_STANDBY	ADC standby mode When the ADC is running at 240kHz Clock rate, the current dissipation of the stop operation in Standby Mode will drop from 460μA down to 75μA
	1:0	R/W	10	ADC_VCM_LS	When VDDA=1.8V, VDDA/2=0.9V≈NMOS Threshold Voltage, and let Pre-Amplifier Input NMOS OFF, so VCM need more +0.18V. 00: VCM=VDDA*(10/20) 01: VCM=VDDA*(11/20) 10: VCM=VDDA*(12/20) 11: VCM=VDDA*(13/20)

R/W1C: read & write one clear

20.4 Functional Description

20.4.1 Conversion function

20.4.1.1 ADC_SLOW=0, When ADC clock ≤ APB clock

In Single conversion mode, the ADC performs a single sequence of conversions, converting all the channels once. This mode is selected when ADC_CONT=0 & ADC_SLOW=0.

In continuous conversion mode, the ADC performs a sequence of conversions, converting all the channels once and automatically re-starts and continuously performs the same sequence of conversions. This mode is selected when ADC_CONT=1 & ADC_SLOW=0.

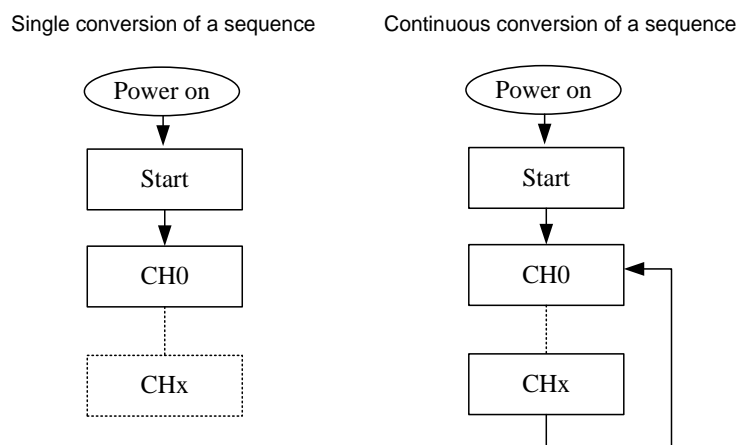


Figure 58 Conversion of a Sequence by ADC_SLOW=0

20.4.1.2 ADC_SLOW=1, When ADC clock > APB clock

In Single conversion mode, the ADC performs a single conversion. This mode is selected when ADC_CONT=0 & ADC_SLOW=1.

In continuous conversion mode, the ADC performs a sequence of conversions, converting all the channels once and automatically re-starts and continuously performs the same sequence of conversions. This mode is selected when ADC_CONT=1 & ADC_SLOW=1.

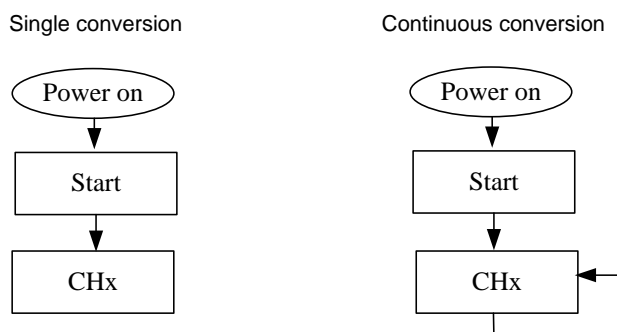


Figure 59 Conversion by ADC_SLOW=1

20.4.2 Conversion timing

$$t_{\text{CONV}} = t_{\text{SMPL}} + t_{\text{SAR}} = 2 \cdot \text{ADCCLK} + 12 \cdot \text{ADCCLK} = 14 \cdot \text{ADCCLK}$$

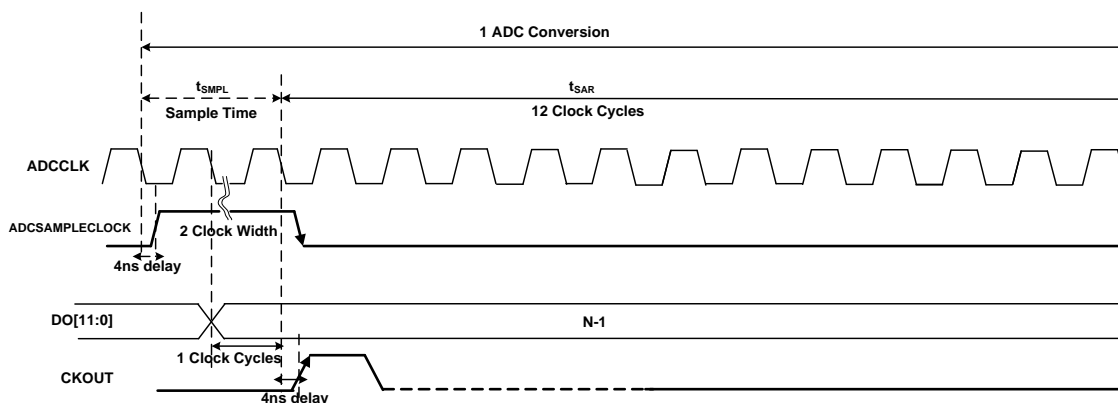


Figure 60 ADC Conversion Time

Example timing diagrams

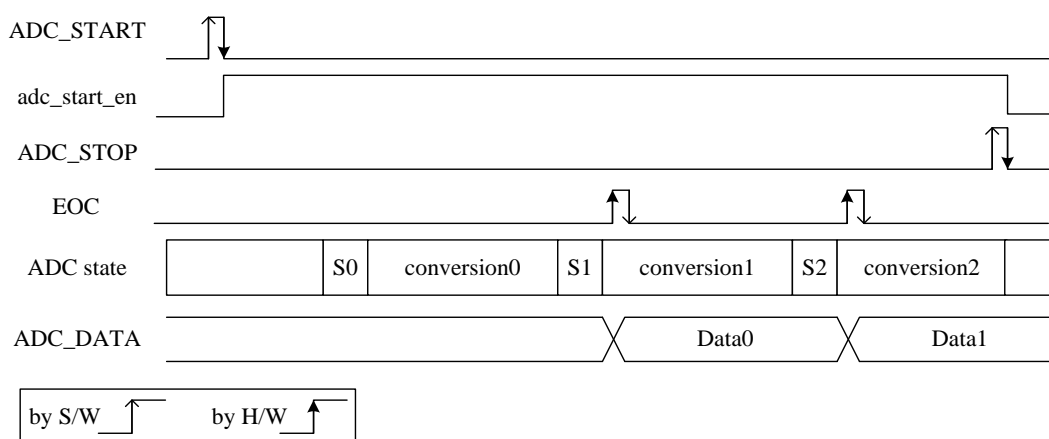


Figure 61 Timing Diagram by ADC_SLOW=0

20.4.3 Analog Watchdog Window

The ADC_WD analog Watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. These thresholds are programmed in HT[11:0] and LT[11:0] bit. An interrupt can be enabled by setting the ADC_WDIE bit.

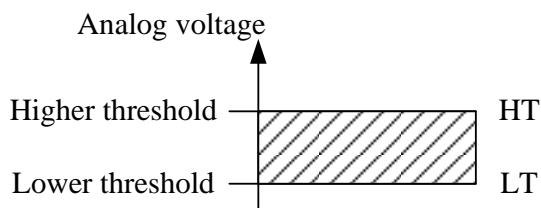


Figure 62 Analog Watchdog Window

21 DAC

21.1 Main Features

- 12-bit DAC
- DAC Rail-to-Rail Amplifier output

21.2 Block Diagram

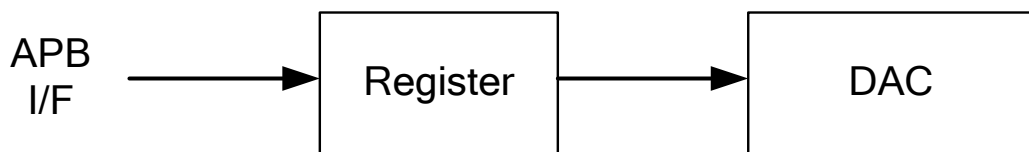


Figure 63 DAC Block Diagram

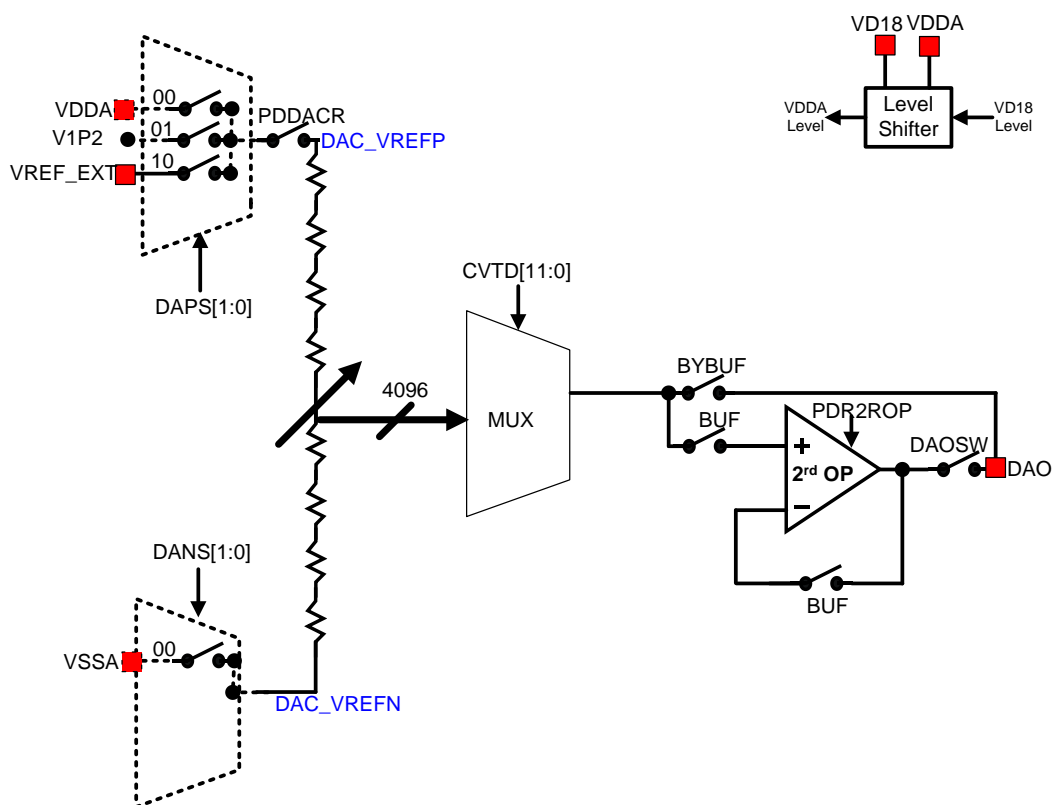


Figure 64 DAC Analog Block Diagram

21.3 Register Definition

Table 47 DAC Control Register

Index	Bit	R/W	Default	Name	Description
DAC_CFG: DAC configuration					
00h	31:7	-	-	-	Reserved
	6	R/W	0	DAC_BYBUF	DAC output bypass Rail-to-Rail Amplifier(no driving ability)
	5	R/W	0	DAC_BUF	DAC output + Rail-to-Rail Amplifier(can driving output capacitance up to 1500pF) *DAOSW need write to 1
	4	R/W	0	DAOSW	Rail-to-Rail Amplifier output to PAD
	3:2	R/W	0	DAPS	DAC positive input selection 00: VDDA 01: V1P0V 10: VREF_EXT 11: Reserved *When DAPS is set to 2'b01 (VREF selects V1P0), the IC working current will increase by 150μA, so when saving power, please set DAPS to 2'b00.
	1:0	R/W	0	DANS	DAC negative input selection *Must set 00 Select VSSA
DAC_CVTD: DAC 12-bit data input					
04h	31:12	-	-	-	Reserved
	11:0	R/W	0	CVTD	DAC 12-bit data input

22 Ultra-Low Power Comparators

22.1 Main Features

WT32L064/032 built-in two voltage comparators with features as listed below.

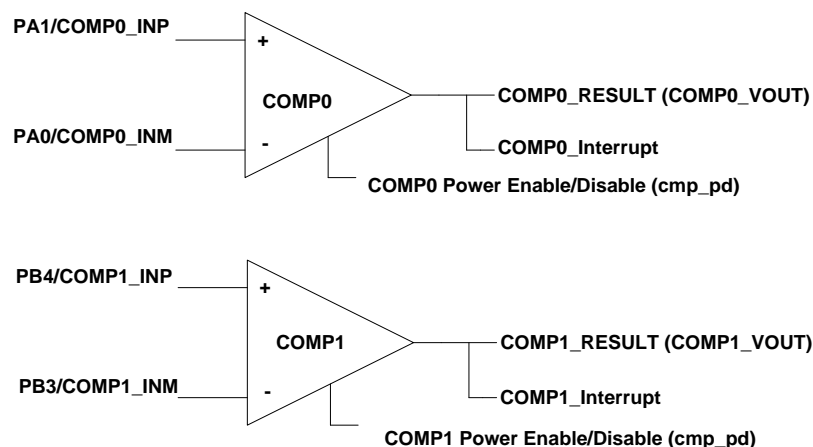
- Ultra low power consumption (operating current less than 10μA/per-unit)
- Comparators can be enabled or disabled individually
- No comparator output pin but can read output status by register
- Each comparator output change can generate interrupt

Range	Index	Function	Description
AHB	0x5001_F000~0x5001_FFFF	system control	

Index	Bit	R/W	Default	Name	Description
SYSCON_CMP_OUT: Comparator output					
14h	1	R	-	CMP1_VOUT	Comparator 1 output
	0	R	-	CMP0_VOUT	Comparator 0 output

Range	Index	Function	Description
APB0	0x4001_A000~0x4001_AFFF	PMU	power management unit

Index	Bit	R/W	Default	Name	Description
PMU_VD_CR: Voltage detector control signals					
0Ch	3:2	R/W	11	cmp_hs	Select the operation mode of the CMP High Speed Mode: 1 Low Speed Mode: 0 cmp_hs[1]: CMP1 cmp_hs[0]: CMP0
PMU_CLK_PD: Clock power signals					
14h	8:7	R/W	11	cmp_pd	CMP0 & CMP1 power down [7]:CMP0,[8]:CMP1 0: Power up 1: Power down



23 CRC32

23.1 Main Features

A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.

In the scope of the EN/IEC 60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link-time and stored at a given memory location.

- CRC-32 polynomial (0x04C11DB7) $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Single input/output register
- CRC computation done in 4 clock cycles
- Support word, half-word & byte transfer size of AHB-Lite I/F
- Support big/little endian

23.2 Register Definition

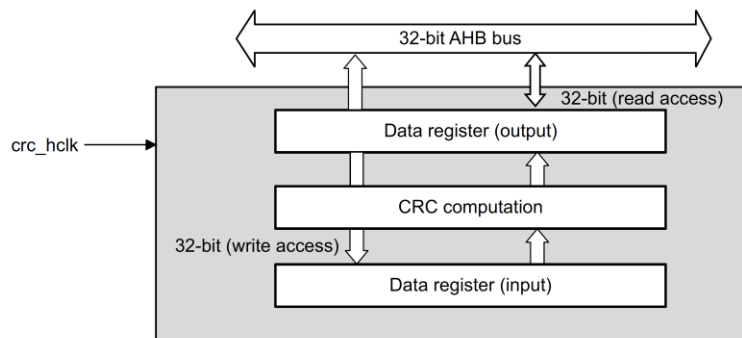
Table 48 CRC32 Control Register

Index	Bit	R/W	Default	Name	Description
CRC32_DR: CRC32 data register					
00h	31:0	R/W	0	DR	Writing new data to this register for CRC calculation. Holds the previous CRC calculation result when it is read.
CRC32_RST: CRC32 reset bit					
04h	31:2	-	-	-	Reserved
	1	R/W	0	LE	Little endian enable: 0: big endian 1: little endian
	0	W	-	RST	Write 1 to reset the CRC calculation and set the data register as 0x00. It is automatically clear by hardware.

23.3 Functional Description

The CRC calculation has a single 32-bit data register, which is used as an input register to enter new data in the CRC calculator (when writing into the register), and holds the result of the previous CRC calculation (when reading the register). Please note that the write operation is halted until the end of the CRC computation.

Each write operation into the data register creates a combination of the previous CRC value and the new one. The CRC calculator can be reset to 0x00 by writing 1 to the RST register.



23.4 CRC32 example code: using WT32L064/032

```
#define BUFFER_SIZE    64
static const uint32_t DataBuffer[BUFFER_SIZE] =
{
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //0
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //1
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //2
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //3
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //4
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //5
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //6
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //7
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //8
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //9
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //10
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //11
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //12
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //13
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //14
    0x00000001, 0x00000002, 0x00000003, 0x00000004 //15
};
```

```
__IO uint32_t CRCValue = 0;
```

```
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_CRC32, ENABLE);
```

```
CRC_ResetDR();
```

```
CRCValue = CRC_CalcBlockCRC((uint32_t *)DataBuffer, 64);
```

```
printf("64 words, crc=0x%08x\r\n",CRCValue);
```

Output:

```
64 words, crc=0x13388b4a
```

CRC 32 example code on PC

/ The result should be in accordance with ISO 3309, ITU - T V.42, Gzip and PNG.*

This code is a translation from Ruby, with an adjustment to use 32-bit integers.

*This code happens to resemble the examples from RFC 1952 section 8 and from PNG annex D, because those examples use an identical table. */*

```

uint32_t crc_table[256];
void CrcTable(void)
{
    uint32_t rem;

    for (int i = 0; i < 256; i++) {
        rem = i; /* remainder from polynomial division */
        for (int j = 0; j < 8; j++) {
            if (rem & 1) {
                rem >>= 1;
                rem ^= 0xedb88320;
            }
            else
                rem >>= 1;
        }
        crc_table[i] = rem;
    }
}

uint32_t rc_crc32(uint32_t crc, unsigned char *buf, size_t len)
{
    uint8_t raw_byte;
    unsigned char *start_p, *end_p;
    uint32_t rotate_H8;
    uint32_t nibble_H8;
    uint32_t rem_result;

    crc = ~crc; /*invert first CRC input
    end_p = buf + len;
    for (start_p = buf; start_p < end_p; start_p++)
    {
        raw_byte = *start_p; /* Cast to unsigned octet(8). */

        nibble_H8= (crc & 0xff) ^ raw_byte;
        rem_result = crc_table[nibble_H8];
        rotate_H8 = (crc >> 8);
        crc = rotate_H8 ^ rem_result;
    }

    return ~crc; /*invert first CRC output
}

int main()
{
    uint32_t crc_result;
    unsigned char test_data[] =
    { //0 //1 //2
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //0
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //1
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //2
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //3
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //4
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //5
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //6
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //7
    }
}

```



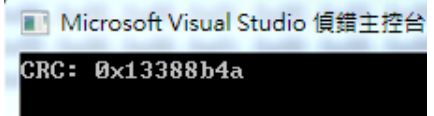
```

0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //8
0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //9
0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //10
0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //11
0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //12
0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //13
0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //14
0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04 //15
};

CrcTable();
crc_result = rc_crc32(0, test_data, sizeof(test_data));
printf("CRC: 0x%08x\n", crc_result);
return 0;
}

```

Output:



Microsoft Visual Studio 偵錯主控台
CRC: 0x13388b4a

24 Boot ROM & IAP

24.1 Introduction

There are 8K bytes Boot ROM built in WT32L064/032, which contains the Boot Code namely IAP(in application program)code for upgrade User Code purpose in far side.

24.2 How to enter the IAP

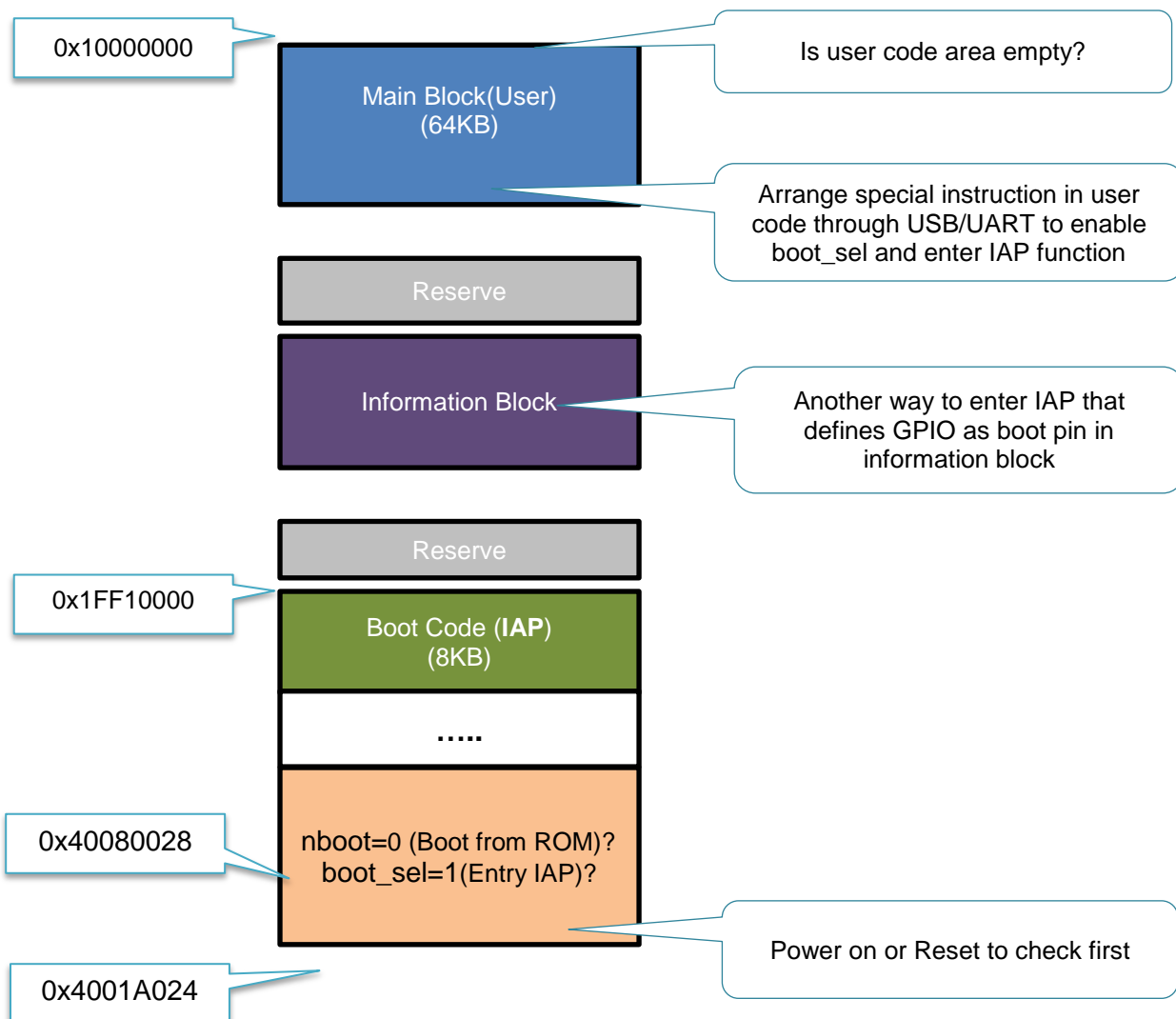
While CPU RESET (any kinds) will go to boot code first, later jump to Flash area to run the user code. In general, by means of USB/UART application code to run the IAP function. There are several ways allow to enter Boot Code for IAP execution.

1. The control register nboot (0x40080028) of Flash unit will decide which memory run first? Boot ROM (default), Flash or SRAM?
2. User code includes USB/UART application program, which receives the special instructions to enable control register boot_sel (0x4001A024) of PMU unit then through software reset to enter boot code. The boot code will check boot_sel status if enable then enter IAP function and clear boot__sel status for next time check else back to user code.
3. Another way to enter Boot Code directly is defined in information block belong to system host area. It could define which GPIO as boot pin to low enable or high enable to enter Boot Code directly. For example, before plant production, GPIOA1 was defined as boot pin and low trigger to enable IAP function. If GPIOA1 is high level then go back to user code.
4. Boot Code also check the address range like as stack pointer (SP) and program counter (PC) whether normal or abnormal? If normal flash exists user code, else flash is empty then enter IAP.

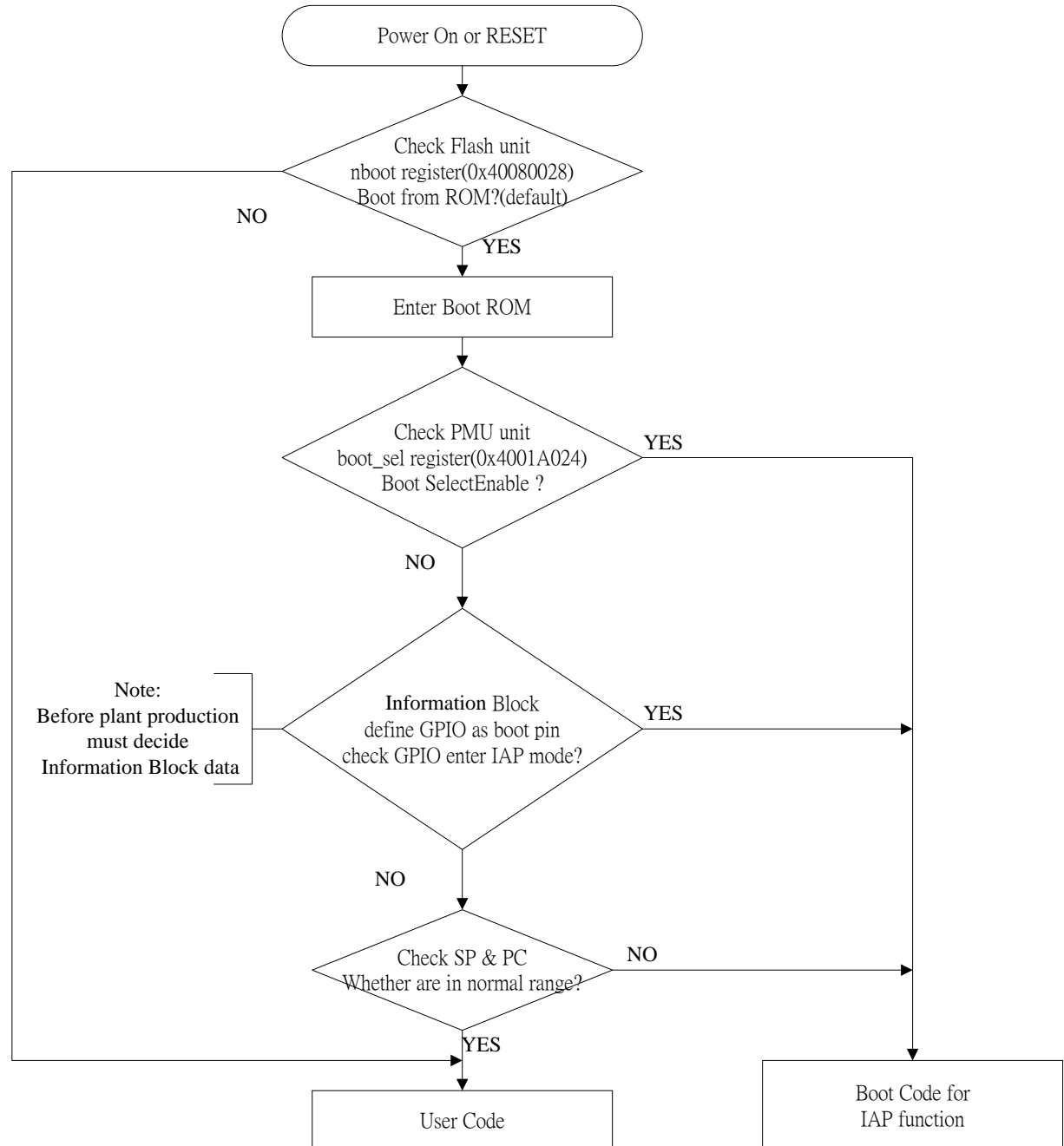
24.3 Boot & IAP Memory Mapping

Below figure describes the Boot & IAP memory mapping and enter condition & Illustration diagram

Index	Function	Description
0x1000_0000~0x1000_FFFF	embedded flash	64KB
0x1FF1_0000~0x1FF1_1FFF	boot ROM	8KB
0x4000_0000~0x4003_FFFF	APB0 memory space	256KB
0x4008_0000~0x400B_FFFF	AHB memory space	256KB



24.4 Flowchart



25 Electrical Characteristics

25.1 Absolute Maximum Ratings

Parameter	Symbol	Condition	Range	Units
D.C. Supply Voltage	V_{D33}		-0.3 ~ 3.6	V
Input Voltage	V_I		-0.3 to $V_{D33} + 0.3$	V
Output Voltage	V_O		-0.3 to $V_{D33} + 0.3$	V
Total current source by all GPIO	ΣI_{OH}	125mA @3mA(6mA)/single pad	@-40°C ~ +85°C	mA
	ΣI_{OH}	85mA @3mA(6mA)/single pad	@-40°C ~ +105°C	mA
Total current sink by all GPIO	ΣI_{OL}	125mA @3mA(6mA)/single pad	@-40°C ~ +85°C	mA
	I_{OH}	85mA @3mA(6mA)/single pad	@-40°C ~ +105°C	mA
Ambient Temperature	T_A		-40 ~ 105	°C
Storage Temperature	T_{STG}		-60 ~ 125	°C

Note: Stresses above those listed may cause permanent damage to the devices.

(*): These parameters are presented for design guidance only and not tested or guaranteed.

Parameter	Symbol	Conditions	Min.	Max.	Units
Internal AHB clock frequency	fHCLK		0	32	MHz
Internal APB0 clock frequency	fPCLK1		0	32	MHz
Internal APB1 clock frequency	fPCLK2		0	32	MHz
Standard operating voltage	VD33	BOR detector disabled	1.65	3.6	V
		BOR detector enabled, at power-on	1.8	3.6	V
		BOR detector disabled, after power-on	1.65	3.6	V
Analog operating voltage (all features)	VDDA	Must be the same voltage as VD33	1.65	3.6	V
Digital power from Internal LDO 1.2V~1.8V	VDD	need to connect capacitor 1μF	1.2	1.8	V
Standard operating voltage, USB domain	VD33 (USB)	USB peripheral used	3	3.6	V
		USB peripheral not used	1.65	3.6	V
VBUS:5V input, USB domain	VDD_5V	USB peripheral used	4.5	5.5	V
Input voltage on 5V Tolerant I/O & reset pins	VIN		-0.3	5.5	V
Input voltage on 3.3V I/O pin			-0.3	$VD33 + 0.3$	V

(*): These parameters are presented for design guidance only and not tested or guaranteed.

25.2 DC Characteristics

25.2.1 Power consumption

VD33=ADDA=3.0V, Vcore=1.8V

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
Run mode	IVDD32M-1	sysclk= (HIS 16M x2) =32M (PLL on) AHB=sysclk no load & peripheral off		8.9		mA
Run mode	IVDD32M-2	sysclk= (HIS 16M x2) =32M (PLL on) AHB=sysclk/2 no load & peripheral off		5.5		mA
Run mode	IVDD16M-1	sysclk= HSI=16M AHB=sysclk no load & peripheral off		5.2		mA
Run mode	IVDD16M-2	sysclk= HSI=16M AHB=sysclk/2 no load & peripheral off		3		mA
Low Power Run mode	IVDD1M	sysclk= MSI=1M AHB=sysclk/2 no load & peripheral off		202		μA
Low Power Run mode	IVDD131K	sysclk= MSI=131K AHB=sysclk/2 no load & peripheral off		57		μA
Sleep mode	ISleep16M	sysclk= HSI=16M AHB=sysclk/2 no load & peripheral off		1.1		mA
Sleep mode	ISleep1M	sysclk= MSI=1M AHB=sysclk/2 no load & peripheral off		113		μA
Stop mode	ISTOP1	SRAM(retain) , All clock (off) Flash (off) no load & peripheral off		0.6		μA
Stop mode+ RTC	ISTOP2	SRAM(retain), MSI, PLL, HSI, HSE (off)		1.3		μA

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
		LSI, LSE (on) Flash (off) no load & peripheral off				
Standby mode	ISTBY1	SRAM(lost) All clock (off) Flash (off) Vcoer (off) no load & peripheral off		0.3		μA
Standby mode+ RTC	ISTBY2	SRAM(lost) MSI, PLL, HSI, HSE (off) LSI, LSE (on) Flash (off) Vcoer (off) no load & peripheral off		1.0		μA

(*): These parameters are presented for design guidance only and not tested or guaranteed.

25.2.2 Digital I/O Characteristics

VDDA=3.3V

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
Schmitt Trigger Low-to-High threshold Point	V_{T+}		0.65*Vdd			V
Schmitt Trigger High-to-Low threshold Point	V_{T-}				0.35*Vdd	V
Output High Voltage (DS=0)	V_{OH}	$I_{OH} = 3 \text{ mA}$	2.4			V
Output Low Voltage (DS=0)	V_{OL}	$I_{OL} = 3 \text{ mA}$			0.4	V
Output High Voltage (DS=1)	V_{OH}	$I_{OH} = 6 \text{ mA}$	2.4			V
Output Low Voltage (DS=1)	V_{OL}	$I_{OL} = 6 \text{ mA}$			0.4	V
Input Schmitt trigger hysteresis voltage	V_{HYS}		300			mV
Input Leakage	I_{OZ}	$V_O = 0V \text{ or } 3.3V$			±0.05	μA

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
Current						
Pull-Up Resistor	R _{PH}			30		K Ω
Pull-Down Resistor	R _{PD}			50		K Ω

(*): These parameters are presented for design guidance only and not tested or guaranteed.

25.2.3 LVR Characteristics

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
Low V1D35 Reset Voltage	V _{LVR13_H}	VDDA=3.3V		1.40		V
Low V1D35 Reset Voltage	V _{LVR13_L}	VDDA=3.3V		1.35		V
Low V1D35 Reset Current	I _{LVR13}	VDDA=3.3V		< 2		μ A
Low V1D35 Reset Voltage				± 3		%

(*): These parameters are presented for design guidance only and not tested or guaranteed.

25.2.4 PLL Characteristics

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
PLL input reference frequency (after pre-divider)	f _{REF}		2		24	MHz
PLL VCO range	f _{VCO}	VDDA < 1.8V	32		96	MHz
		VDDA \geq 1.8V	32		192	
PLL lock time	t _{LOC}	f _{REF} = 16MHz, f _{VCO} = 96MHz		160		μ S
PLL operation current	I _{PLL}	f _{REF} = 4.2MHz, f _{VCO} = 64MHz		0.75		mA

(*): These parameters are presented for design guidance only and not tested or guaranteed.

25.2.5 ADC Characteristics

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
Resolution				12		bits
Integral Nonlinearity Error (including INL, DNL)		$V_{REF} = 3.3V, V_{DDA} = 3.3V$		± 2		LSB
Input Voltage	V_{ADCIN}		VSS		V_{REF}	V
ADC Reference Voltage (V_{REF})	VDDA	$1.8V < V_{DDA} < 3.6V$		VDDA		V
	V1P0	$V_{DDA} > 2.2V$ & $F_{ADC} \leq 7MHz$		1.0		V
	V_{REF_EXT}	$1.8V < V_{DDA} < 3.6V$	1.65		VDDA	V
ADC Frequency	F_{ADC}			7		MHz
ADC Current	I_{ADC}	ADC CLK=6MHz		400		μA
		ADC CLK=1MHz		314		μA
ADC Input impedance	R_{AIN}				20	k Ω
Conversion time	T_{conv}	$f_{ADC} = 6 MHz,$		2.33		μs

(*): These parameters are presented for design guidance only and not tested or guaranteed.

Parameter	Condition	Specification			Units
		Min.	Typ.	Max.	
ENOB	$V_{DDA}=V_{ref}=3.6V$		10		bit
	$V_{DDA}=V_{ref}=3.3V$		10		bit
	$V_{DDA}=V_{ref}=1.8V$		9		bit

25.2.6 DAC Characteristics

Parameter	Symbol	Conditions	Specification			Units
			Min.	Typ.	Max.	
Analog supply voltage	VDDA		2.2		3.6	V
VREFP=VDDA Channel	Resolution			10		bits
VREFP=Internal 1.0V Channel				9		bits
VREFP=External Channel				10		bits
Analog supply voltage	VDDA		2.2		3.6	V
VREFP=External Channel of 0.6V				10		bits
Current consumption on VDDA Supply, VDDA = 3.3 V @0X0800	IDAC			86.2		μA
Current consumption on VDDA Supply, VDDA = 3.3 V @0X0F1C	IDAC			118.7		μA
Differential non linearity	DNL	@VDDA=2.2V~3.6V		±2		LSB
Integral non linearity	INL	@VDDA=2.2V~3.6V		±4		LSB
Resistive Load	RL	DAC Output Buffer ON RL to VSSA	5			kΩ
Capacitive Load	CL	DAC Output Buffer ON CL to VSSA			50	pf

*If DAC used at VD33 in 1.65V~2.2V, External Channel must be provided 0.6V at PB0 for 8 bits resolution.

*It must be back to default channel of VDDA when DAC off to reduce current consumption of 150μA.

25.2.7 LDO Characteristics

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
LDO 3.3V Power Current @VDD5V = 5.0V	I _{VDD5V}	VD33 ≥ 3.0V		30		mA
LDO 3.3V Power Current @VDD5V = 4.5V	I _{VDD5V45}	VD33 ≥ 3.0V		20		mA
Voltage of LDO 3.3V @Loading=3mA	VD33		3.17	3.3	3.43	V
Voltage of LDO 3.3V Tolerance @Loading=3mA				±4		%

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
LDO 1.8V Driven Current @VDD33 = 2.2V	I _{VDDC22}	BLDO ON VD18 ≥ 1.7V		30		mA
LDO 1.8V Driven Current @VDD33 = 2.2V	I _{VDDC22S}	BLDO OFF VD18 ≥ 1.7V		285		μA
Voltage of LDO 1.8V @Loading=3mA	VD18		1.774	1.81	1.846	V
Voltage of LDO 1.8V Tolerance @Loading=3mA				±2		%

(*): These parameters are presented for design guidance only and not tested or guaranteed.

25.2.8 HSI 16MHz

Parameter	Symbol	Conditions	Specification			Units
			Min.	Typ.	Max.	
Frequency	f _{HSI16}			16		MHz
HSI16 trimming step	TRIM			±0.4		%
Duty cycle	DuCy		45		55	%
Accuracy of the HSI16 oscillator (factory calibrated)	ACC	VDDA=3V, T _A = 25 °C	-1		1	%
		VDDA=3V, T _A = 0~70 °C		±2		%
		VDDA=3V, T _A = -40°C ~85°C		±3		%
		VDDA=3V, T _A = -40°C ~125°C		±4		%
		VDDA=1.65V ~3.6V, T _A = -40°C ~125°C		±5		%
HSI16 oscillator startup time	t _{su}				20	μs
HSI16 oscillator power consumption	I _{DDA}			100	140	μA

(*): These parameters are presented for design guidance only and not tested or guaranteed.

25.2.9 MSI 4.2MHz

Parameter	Symbol	Conditions	Specification			Units
			Min.	Typ.	Max	
Frequency	f _{MSI}			4.2		MHz
Duty cycle	DuCy		45(49)		55 (51)	%
Accuracy of the MSI oscillator (factory calibrated)	ACC	VDDA=3V, T _A = 25 °C	-1		1	%
MSI oscillator startup time	t _{su}				60	μs
MSI oscillator power consumption	I _{DDA}			15	20	μA

(*): These parameters are presented for design guidance only and not tested or guaranteed.

Parameter	Symbol	Conditions	Specification			Units
			Min.	Typ.	Max.	
Frequency after factory calibration @VDDA=3.3V T _A =25°C	f _{MSI}	MSI range=0		65.625		kHz
		MSI range=1		131.25		
		MSI range=2		262.5		
		MSI range=3		525		MHz
		MSI range=4		1.05		
		MSI range=5		2.1		
		MSI range=6		4.2		
MSI oscillator power consumption	I _{D_{DA}}	MSI range=0		7		μA
		MSI range=1		7.5		
		MSI range=2		8		
		MSI range=3		9		
		MSI range=4		10		
		MSI range=5		12		
		MSI range=6		15		
MSI oscillator frequency drift 0 °C ≤ T _A ≤ 85 °C	D _{TEMP}			±3.0		%
MSI oscillator frequency drift VDDA=3.3V -40°C < T _A < 110°C		MSI range=0		±10		
		MSI range=1		±10		
		MSI range=2		±9		
		MSI range=3		±8		
		MSI range=4		±7		
		MSI range=5		±6		
MSI range=6		±5				
MSI oscillator frequency drift 1.65V < VDDA < 3.6V T _A =25°C	D _{VOLT}		-2.5		2.5	%

(*): These parameters are presented for design guidance only and not tested or guaranteed.

25.2.10 IRC48MHz Characteristics

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
RC Frequency	F _{RC}	VDDA = 2.2V~3.6V		48		MHz
Frequency Tolerance	ΔF _{RC} /F _{RC}	Without crystal oscillator calibration		± 4		%
		With USB calibration		± 0.25		%

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
RCOSC Current	I_{RCOSC}	12/24/48 MHz		0.25		mA

(*): These parameters are presented for design guidance only and not tested or guaranteed

25.2.11 IRC37kHz Characteristics

VDDA=3.3V

Parameter	Symbol	Specification			Units
		Min.	Typ.	Max.	
LSI frequency	f_{LSI}	32.66		55.19	kHz
LSI oscillator frequency drift $0^{\circ}\text{C} \leq T_A \leq 85^{\circ}\text{C}$	D_{LSI}	-3.1	-	+8.1	%
LSI oscillator startup time	$t_{SU(LSI)}$	-	-	235	μs
LSI oscillator power consumption	$I_{DD(LSI)}$	-	393	640	nA

(*): These parameters are presented for design guidance only and not tested or guaranteed

VDDA=1.65V

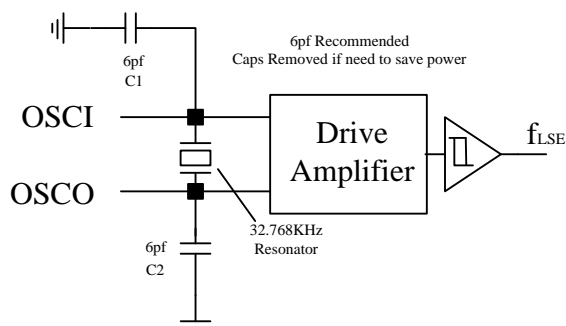
Parameter	Symbol	Specification			Units
		Min.	Typ.	Max.	
LSI frequency	f_{LSI}	32.56		55.1	kHz
LSI oscillator frequency drift $0^{\circ}\text{C} \leq T_A \leq 85^{\circ}\text{C}$	D_{LSI}	-3.4	-	+8.2	%
LSI oscillator startup time	$t_{SU(LSI)}$	-	-	166	μs
LSI oscillator power consumption	$I_{DD(LSI)}$	-	390	592	nA

(*): These parameters are presented for design guidance only and not tested or guaranteed.

25.2.12 Crystal 32768 Characteristics

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
LSE oscillator frequency	f_{LSE}	-		32.768		kHz
Startup time	$t_{SU(LSE)}$	V_{DD} is stabilized		2		Sec.
Crystal transconductance	Gm			0.3	0.5	$\mu\text{A/V}$

(*): These parameters are presented for design guidance only and not tested or guaranteed.



25.2.13 CMP Characteristics

Parameter	Symbol	Conditions	Specification			Units
			Min	Typ	Max	
Analog supply voltage	V_{DDA}	-	1.65	-	3.6	V
Comparator input voltage range	V_{IN}	-	+0.2	-	$V_{DDA}-0.2$	V
Comparator startup time	t_{START}	Fast mode		22	29.8	μs
		Slow mode		27.6	36.4	
Propagation delay in slow mode	$t_{d\ SLOW}$	$1.65\ V \leq V_{DDA} \leq 2.7\ V$		1.21	2.23	
		$2.7\ V \leq V_{DDA} \leq 3.6\ V$		1.65	2.13	
Propagation delay in fast mode	$t_{d\ fast}$	$1.65\ V \leq V_{DDA} \leq 2.7\ V$		0.43	0.72μ	
		$2.7\ V \leq V_{DDA} \leq 3.6\ V$		0.55	0.68μ	
Comparator offset error	V_{offset}			± 4	± 20	mv
Current consumption	I_{COMP2}	Fast mode		3.67	4.67	μA
		Slow mode		0.95	1.15	

(*): These parameters are presented for design guidance only and not tested or guaranteed

25.2.14 BOR/PVD Characteristics

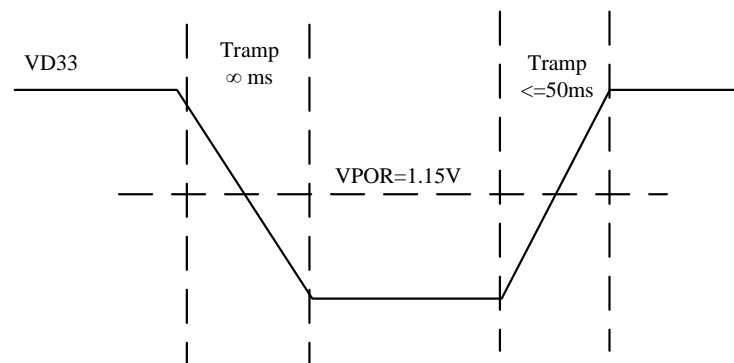
Parameter	Symbol	Conditions	Specification			Units
			Min.	Typ.	Max.	
VDD rising time rate	t _{VDD}	BOR detector enabled			15	ms/V
		BOR detector disabled			15	ms/V
VDD falling time rate		BOR detector enabled			∞	ms/V
BOR detector disabled				∞	ms/V	
Analog supply Operating voltage	V _{DDA}	-	1.65	-	3.6	V
BOR disabled			1.65		3.6	V
BOR enabled			1.8		3.6	V
Brown-out reset threshold 0	V _{BOR0}	Rising edge		1.75		V
		Falling edge		1.7		V
Brown-out reset threshold 1	V _{BOR1}	Rising edge		2.03		V
		Falling edge		1.93		V
Brown-out reset threshold 2	V _{BOR2}	Rising edge		2.4		V
		Falling edge		2.3		V
Brown-out reset threshold 3	V _{BOR3}	Rising edge		2.66		V
		Falling edge		2.55		V
Brown-out reset threshold 4	V _{BOR4}	Rising edge		2.9		V
		Falling edge		2.8		V
Programmable Voltage detector threshold 0	V _{PVD0}	Rising edge		1.95		V
		Falling edge		1.85		V
Programmable Voltage detector threshold 1	V _{PVD1}	Rising edge		2.14		V
		Falling edge		2.04		V
Programmable Voltage detector threshold 2	V _{PVD2}	Rising edge		2.34		V
		Falling edge		2.24		V
Programmable Voltage detector threshold 3	V _{PVD3}	Rising edge		2.54		V
		Falling edge		2.44		V
Programmable Voltage detector threshold 4	V _{PVD4}	Rising edge		2.74		V
		Falling edge		2.64		V
Programmable Voltage detector threshold 5	V _{PVD5}	Rising edge		2.93		V
		Falling edge		2.83		V
Programmable Voltage detector threshold 6	V _{PVD6}	Rising edge		3.15		V
		Falling edge		3.05		V
Programmable Voltage detector Tolerance	V _{PVD}			±3		%

(*): These parameters are presented for design guidance only and not tested or guaranteed.

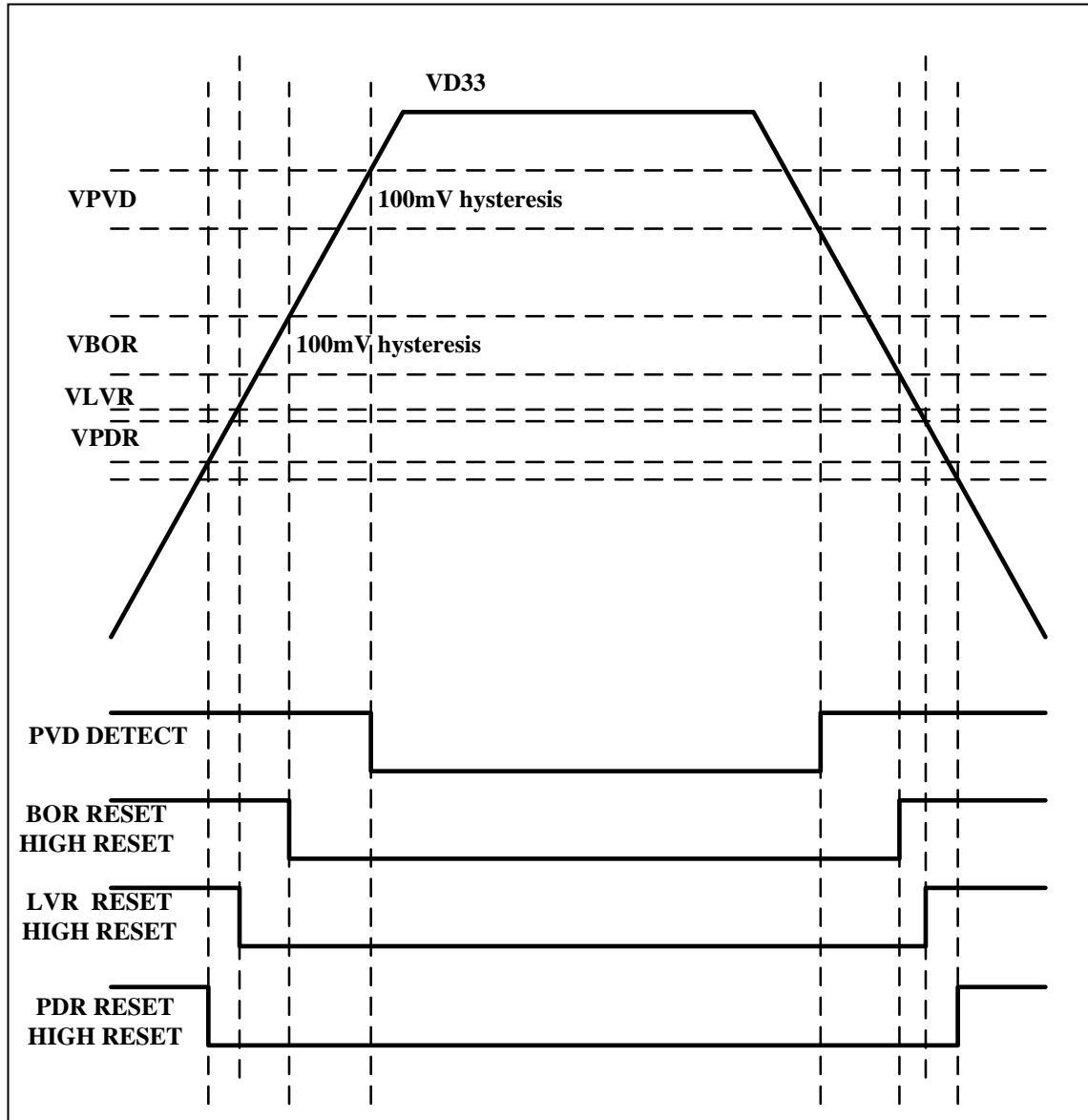
25.2.15 POR Characteristics

Parameter	Symbol	conditions	Specification			Units
			Min.	Typ.	Max.	
VDD Rising time rate	t_{VDD}				15	ms/V
VDD Falling time rate					∞	ms/V
POR	V_{POR}	Rising		1.15		V
		Falling		0.4		V

(*): These parameters are presented for design guidance only and not tested or guaranteed.



25.2.16 POWER Supply Supervisors



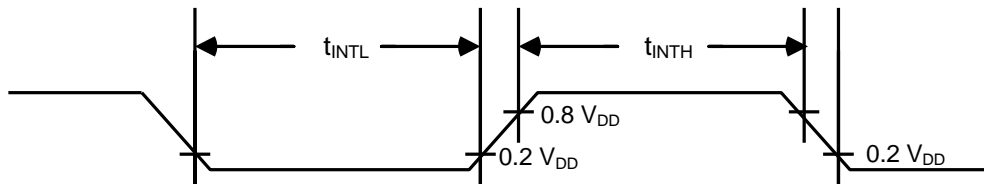
1. PVD is enabled or disabled by software
2. BOR is enabled of operating from 1.8V to 3.6V
3. PDR reset level is around at 1.0V. There is no BOR when VD33 operating from 1.65V to 3.6V and the reset is enabled when VD33 goes below PDR level.
4. LVR reset level is around at 1.35V.

25.3 AC Characteristics

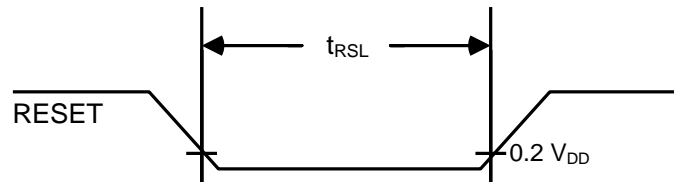
Parameter	Symbol	Pin/condition	Specification			Units
			Min	Typ.	Max	
Main Operation Frequency	F_{MCP}	X_{IN}	0.032		32	MHz
Main Crystal Stabilization Time(*)		$V_{D33} = 1.8V \sim 3.6V$ at HSE 16 MHz			10	ms
		$V_{D33} = 1.8V \sim 4.5V$ at MSI MHz			30	ms
		$V_{D33} = 4.5V \sim 5.5V$ at Hz		0.5	1	s
		$V_{D33} = 1.8V \sim 3.6V$ at 32768 Hz			10	s
Interrupt Input High, Low Width (IRQx)	t_{INTH} , t_{INTL}	MCU clock = 12 MHz	167			ns
RESET Input Low Width	t_{RSL}	RST_NDF = 1, main clock = 12 MHz	334			ns

(*): These parameters are presented for design guidance only and not tested or guaranteed.

Input Timing for External Interrupts



Input Timing for RESET



25.4 Thermal Characteristics

Parameter	Symbol	Feature	Condition	Typ.	Units
TH01	θ_{JA}	Thermal Resistance (Junction to Air)	64-pin LQFP package	57	°C/W
TH02	θ_{JC}	Thermal Resistance (Junction to Case)	64-pin LQFP package	15	°C/W
TH03	TJMAX	Maximum Junction Temperature	64-pin LQFP package	125	°C

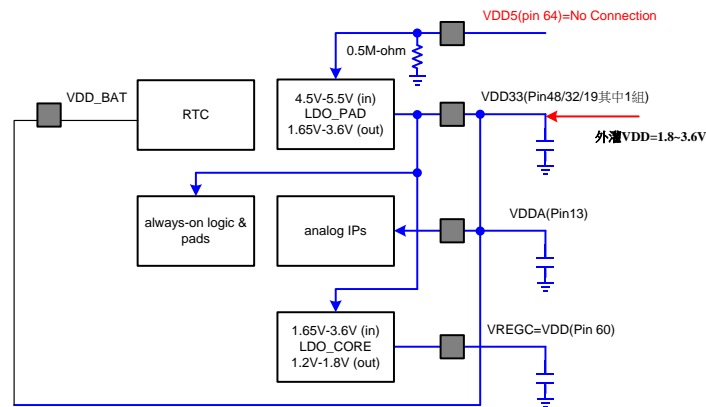
Parameter	Symbol	Feature	Condition	Typ.	Units
TH01	θ_{JA}	Thermal Resistance (Junction to Air)	48-pin LQFP package		°C/W
TH02	θ_{JC}	Thermal Resistance (Junction to Case)	48-pin LQFP package		°C/W
TH03	TJMAX	Maximum Junction Temperature	48-pin LQFP package		°C

Parameter	Symbol	Feature	Condition	Typ.	Units
TH01	θ_{JA}	Thermal Resistance (Junction to Air)	28-pin SOP package		°C/W
TH02	θ_{JC}	Thermal Resistance (Junction to Case)	28-pin SOP package		°C/W
TH03	TJMAX	Maximum Junction Temperature	28-pin SOP package		°C

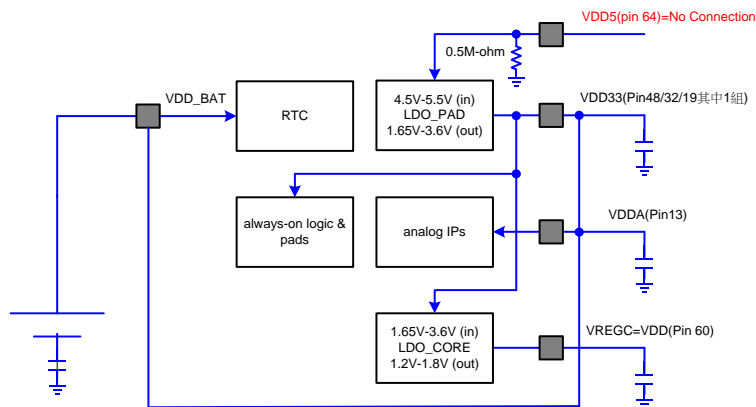
Parameter	Symbol	Feature	Condition	Typ.	Units
TH01	θ_{JA}	Thermal Resistance (Junction to Air)	20-pin SSOP package		°C/W
TH02	θ_{JC}	Thermal Resistance (Junction to Case)	20-pin SSOP package		°C/W
TH03	TJMAX	Maximum Junction Temperature	20-pin SSOP package		°C

26 Application Circuit

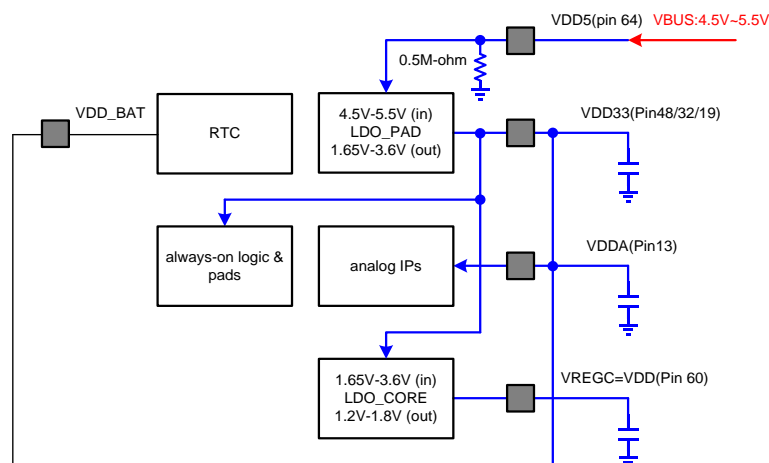
26.1 VDD33 power supply



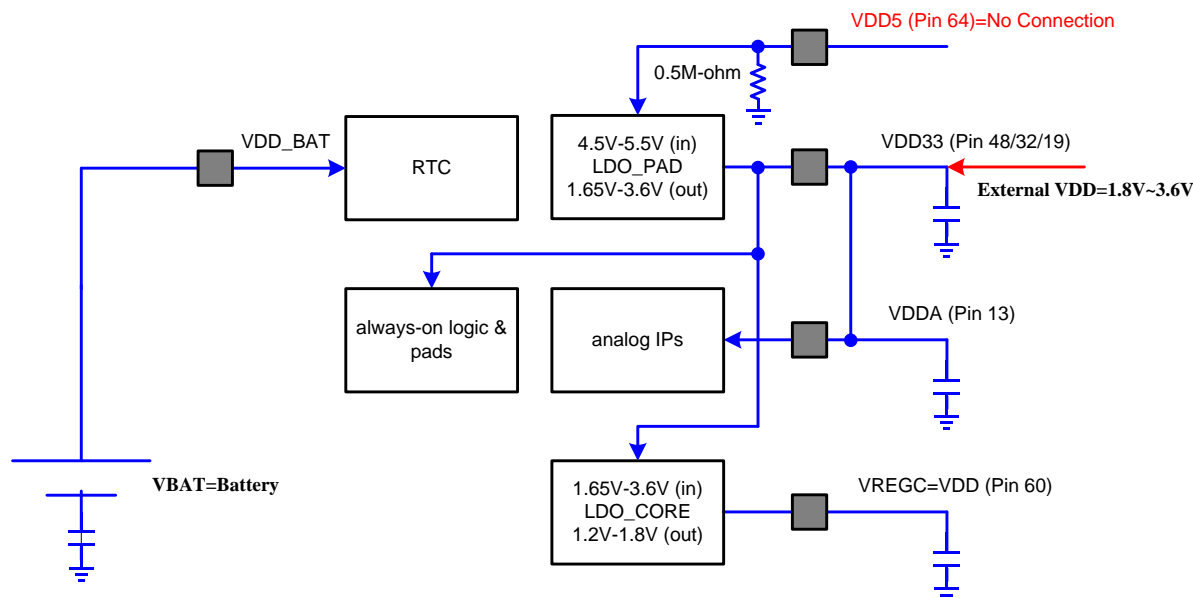
26.2 VDDBat power supply



26.3 USB VBus 5V power supply



26.4 VBAT & VDD33 dual power supply



VD33	VBAT	RESET?	Current Leakage?	Note
VBAT exist				
N→Y	Y	Exception RTC, All are RESET	No	
Y→N	Y	Exception RTC, All are RESET	No	
VD33 exist				
Y	N→Y	RTC RESET The Others No RESET	No	
Y	Y→N	RTC RESET The Others No RESET	No VD33 Domain	VBAT needs to change new battery
VD33 not exist				
N	Y→N	All No RESET	No	All are no power supply
N	Y→N→Y	RTC RESET The Others No RESET	No VBAT Domain	Unreasonable VD33 needs to offer power supply for system operation

27 Product Naming Rule

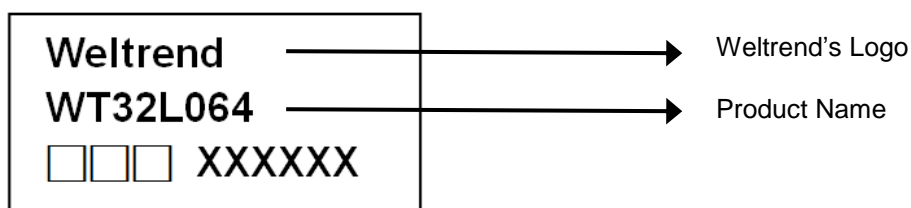
WT	Consumption market	Family	Core	Flash Size (K Bytes)		Note	Remarks
WT	32	L	0	6	4	M0/64KB Flash	32: 32-bit MCU embedded, used in general-purpose or consumption market related product. L/F: L: Ultra low power family F: Generic power family 0/3: 0: ARM Cortex M0 3: ARM Cortex M3
			0	3	2	M0/32KB Flash	
WT	32	F	0	6	4	M0/64KB Flash	
			0	3	2	M0/32KB Flash	
			0	3	3	M0/32KB Flash	
			3	C(12)	8	M3/128KB Flash	
			3	C(12)	9	M3/128KB Flash	
			3	6	4	M3/64KB Flash	

28 Ordering Information

28.1 WT32L064

Package Type	Package Outline	Part Number
64-pin LQFP	7mm x 7mm	WT32L064-RG64AWT
48-pin LQFP	7mm x 7mm	WT32L064-RG48AWT
QFN32	5mm x 5mm	WT32L064-UG32BWT

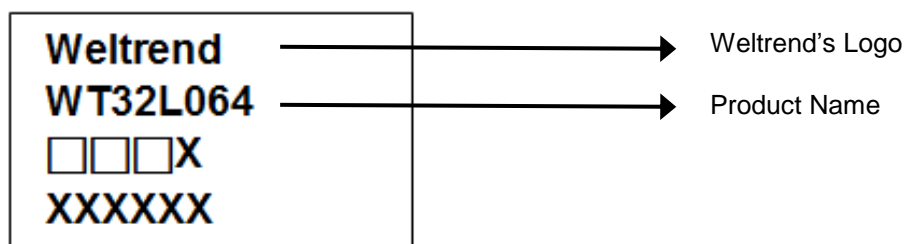
28.1.1 Top Marking – LQFP64/48



□ Date Code

X Production Tracking code

28.1.2 Top Marking – QFN32



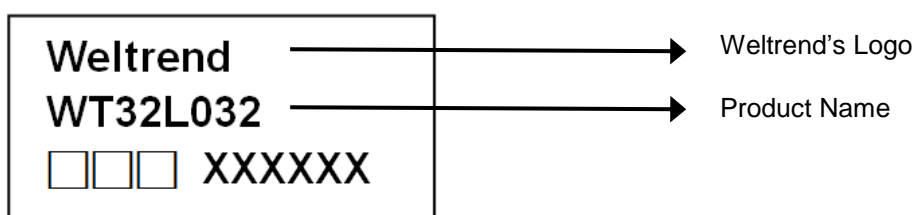
□ Date Code

X Production Tracking code

28.2 WT32L032

Package Type	Package Outline	Part Number
64-pin LQFP	7mm x 7mm	WT32L032-RG64AWT
48-pin LQFP	7mm x 7mm	WT32L032-RG48AWT
QFN32	5mm x 5mm	WT32L032-UG32BWT

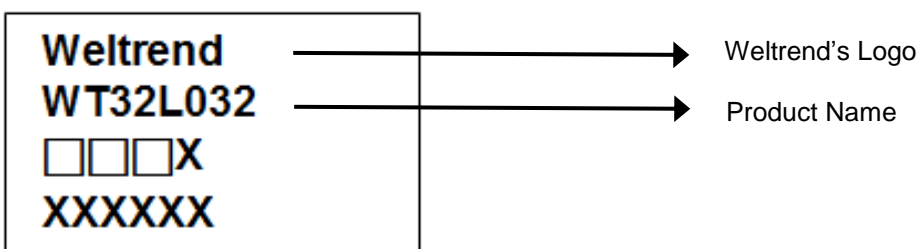
28.2.1 Top Marking – LQFP64/48



□ Date Code

X Production Tracking code

28.2.2 Top Marking – QFN32

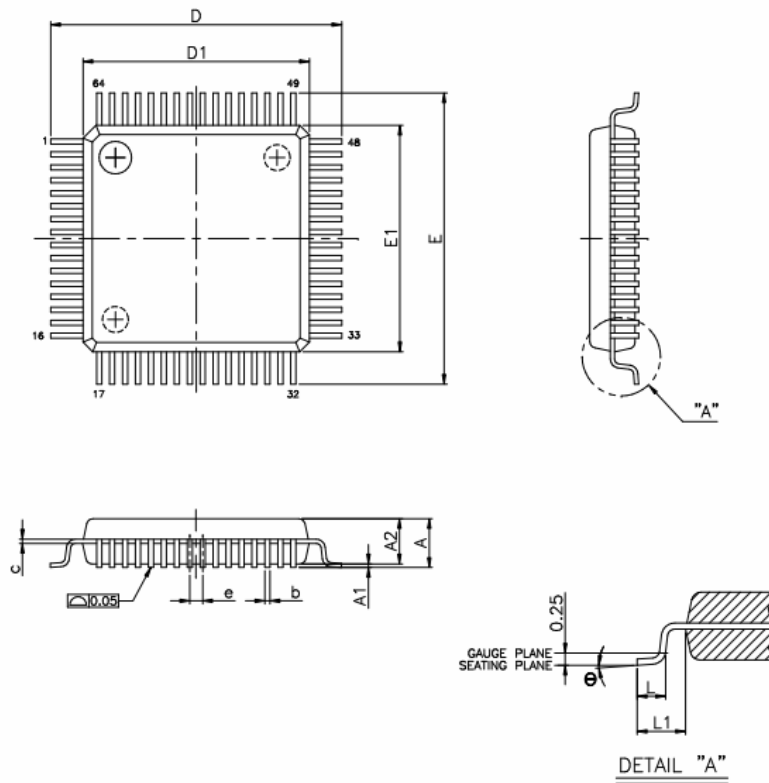


□ Date Code

X Production Tracking code

29 Package Outline Drawing

29.1 LQFP-64 Outline Drawing



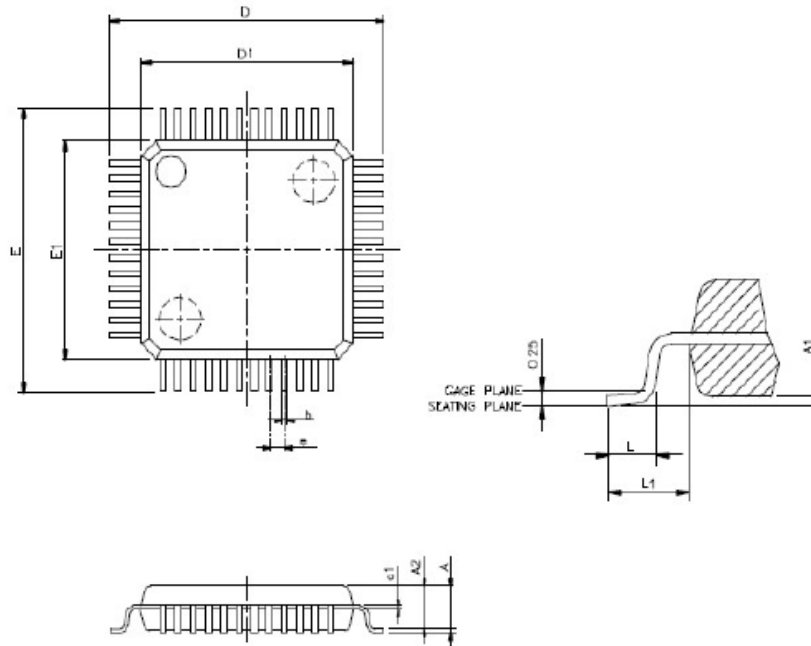
VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	—	—	1.60
A1	0.05	—	0.15
A2	1.35	1.40	1.45
b	0.13	0.18	0.23
c	0.09	—	0.20
D	9.00 BSC		
D1	7.00 BSC		
e	0.40 BSC		
E	9.00 BSC		
E1	7.00 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		
θ	0°	3.5°	7°

NOTES:

1. JEDEC OUTLINE : MS-026 BBD
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25mm PER SIDE. D1 AND E1 ARE MAXIMUM PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.
3. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM b DIMENSION BY MORE THAN 0.08mm.

29.2 LQFP-48 Outline Drawing



VARATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	MAX.
A	--	1.6
A1	0.05	0.15
A2	1.35	1.45
c1	0.09	0.16
D	9.00 BSC	
D1	7.00 BSC	
E	9.00 BSC	
E1	7.00 BSC	
e	0.5 BSC	
b	0.17	0.27
L	0.45	0.75
L1	1 REF	

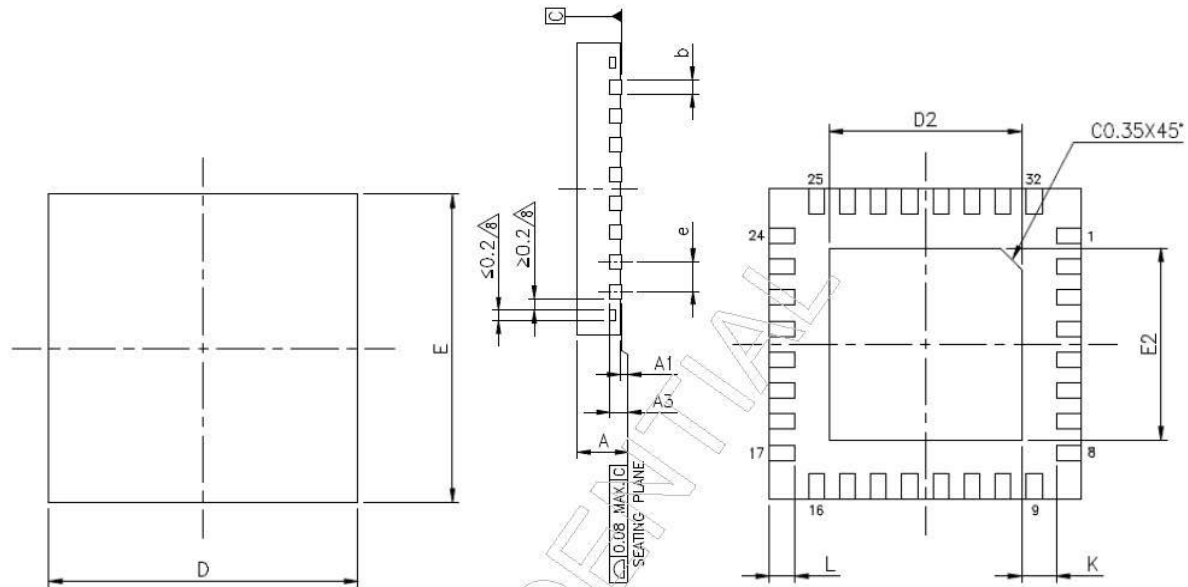
NOTES:

1. JEDEC OUTLINE: MS-026 BBC
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25mm PER SIDE. D1 AND E1 ARE MAXIMUM PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.
3. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM b DIMENSION BY MORE THAN 0.08mm.

29.3 QFN-32 Outline Drawing

Quad Flat No-Lead Plastic Package

QFN-32 PIN



SYMBOLS	MIN	NOR	MAX
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	0.203 REF		
b	0.18	0.25	0.30
D	5.00 BSC		
E	5.00 BSC		
e	0.50 BSC		
L	0.35	0.40	0.45
K	0.20	-	-
D2	3.10	3.20	3.25
E2	3.10	3.20	3.25

UNIT: mm

NOTES:

- JEDEC outline : MO-220
- Dimension "b" applies to metallized terminal and is measured between 0.15mm and 0.30mm from the terminal tip. If the terminal has the optional radius on the other end of the terminal, the dimension "b" should not be measured in that radius area.
- Bilateral coplanarity zone applies to the exposed heat sink slug as well as the terminals.

PREPARE	Cynthia	DATE: 2012/8/1
CHECK	Lawrence	DATE: 2012/8/1
APPROVE	Eric	DATE: 2012/8/1

30 Development Tool

WT32L064/032 development kit can work together with compiler software Keil MDK. They can perform In-Circuit Emulator (ICE) and In-System Programming (ISP) in Windows 2000/XP/Vista/Win7/Win10.

The development kit is illustrated in the figure 65 below:

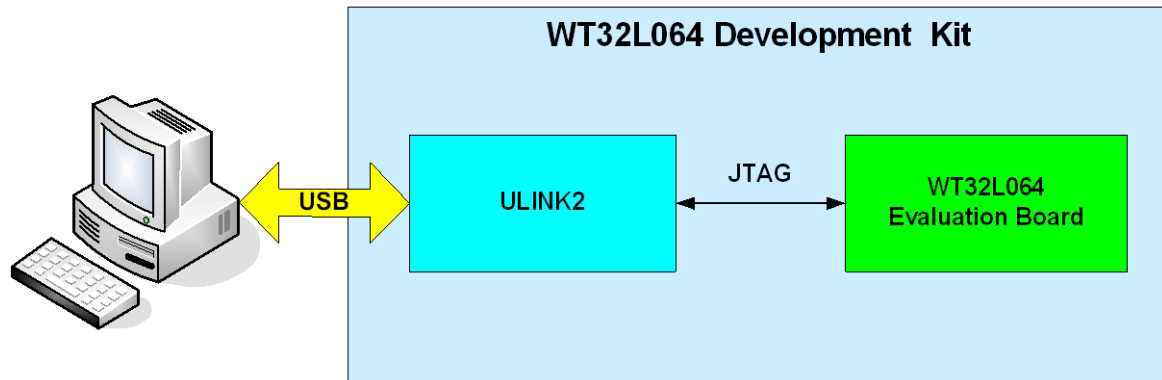


Figure 65 WT32L064/032 Development Kit

31 Revision History

Version	Section/Page	Contents	Date
0.10	All	Initial version	Nov. 12, 2019
0.20	Pin description	Add A1 type statement during reset period	April 01, 2020
	All	Add WT32L032 part No.	
	RCC	Add RTC clock enable	
0.30	System clock tree & RCC & ADC & Electrical Characteristics	Add LSI:37kHz of RTC, Notice of RCC & ADC Add RTC low power clock enable bit	June 15, 2020
0.40	All		Sep. 8, 2020
0.50	7.1	Modify DMA descriptions	June 11, 2021
0.60	4.2.2.1	Update PLL descriptions	July 28, 2021
1.00	Ordering Information	Add Top Marking	Sep. 9, 2021
1.01	Ordering Information	Modify the Bonding version as B Delete “Wafer form or Chip form”	Sep. 16, 2021
	CH29	Remove “Pad Diagram & Location Table” section	
1.02	1.1 CH5 20.3	Update Features. Delete “Low-Power Sleep” descriptions Update ADC descriptions	Feb. 10, 2022