Content.

Figure Catalog

table directory

1. Introduction

1.1 Purpose of writing.

The W800 chip is an embedded Wi-Fi SoC chip launched by Lianshengde Microelectronics. The chip is highly integrated, requires less peripheral devices, and is cost-effective
high. It is suitable for various smart products in the field of IoT (smart home). Highly integrated Wi-Fi and Bluetooth 4.2 Combo functions are its main features;
In addition, the chip integrates XT804 core, built-in QFlash, SDIO, SPI, UART, GPIO, I²C, PWM, I²S, 7816, LCD, Interfaces such as Touch Sensor, support a variety of hardware encryption and decryption algorithms. In addition, the chip MCU contains a security kernel that supports code security permission settings.
The whole system supports firmware encrypted storage, firmware signature, security debugging, security upgrade and other security measures to improve product security features.
This document mainly describes the internal structure of the W800 chip, information on each functional module and detailed register usage information; it is a guide for developers to develop drivers,
The main reference for the application. There are open source implementations of various functions in the SDK provided by Lianshengde Microelectronics, and developers can refer to the corresponding drivers
Programs, application examples to increase understanding of chip functions and register descriptions. There are no register descriptions for the Wi-Fi/BT part of this document.

1.2 References

For information on W800 chip package parameters, electrical characteristics, RF parameters, etc., please refer to "W800 Chip Product Specifications";
The W800 chip integrates the ROM program. The ROM program provides functions such as downloading firmware, MAC address reading and writing, and Wi-Fi parameter reading and writing.
For information, please refer to "WM_W800_ROM Function Brief";
The W800 chip has a built-in 2Mbytes QFlash memory, which is used as a storage space for codes and parameters. This document provides basic QFlash operations
for information. For requirements beyond the scope of this document, you need to refer to the QFlash manual;
W800 chip adopts Hangzhou Pingtou Ge XT804 core, 804 related function introduction, development materials, etc. can refer to Pingtou Ge company release information;

2 Features

● Chip Packaging

➢ QFN32 package, 4mm x 4mm.

● Chip integration

➢ Integrated XT804 processor, up to 240MHz

➢ Integrated 288KB SRAM

➢ Integrated 2MB FLASH

➢ Integrated 8-channel DMA controller, supports 16 hardware applications, and supports software linked list management

➢ Integrated PA/LNA/TR-Switch

➢ Integrated 32.768KHz clock oscillator

➢ Integrated voltage detection circuit

➢ Integrated LDO

➢ Integrated power-on reset circuit

● Chip interface

➢ Integrate 1 SDIO2.0 Device controller, support SDIO 1-bit/4-bit/SPI three operating modes, operating clock range 0~50MHz

➢ Integrate 1 SDIO 2.0 HOST controller, support SDIO and SD card operation, operating clock range 0~50MHz

➢ Integrate 1 QSPI PSRAM interface, support PSRAM with a maximum capacity of 64MB, and a maximum operating clock frequency of 80MHz;

➢ Integrate 5 UART interfaces, support RTS/CTS, baud rate range 1200bps~2Mbps

➢ Integrate a high-speed SPI slave interface, the operating clock range is 0~50MHz

➢ Integrate 1 SPI master/slave interface, the working clock of the master device is up to 20MHz, and the slave device supports up to 6Mbps data transfer rate

➢ Integrate an I2C controller, support 100/400Kbps rate

➢ Integrated PWM controller, support 5-channel PWM

Single output or 2 PWM inputs. Maximum output frequency 20MHz, maximum input frequency 20MHz

➢ Integrated duplex I2S controller, support 32KHz to 192KHz I2S interface codec

➢ Integrate one 7816 interface, compatible with UART interface, support ISO-7816-3 T=0.T=1 mode; support EVM2000 Protocol

➢ Support a variety of hardware encryption and decryption modes, including RSA/AES/RC4/DES/3DES/RC4/SHA1/MD5/CRC8/CRC16/CRC32/TRNG

➢ Integrate one differential, or two single-ended 16bit ADC interfaces;

➢ Integrate 11-way Touch Sensor;

➢ Support up to 17 GPIO ports, each IO port has rich reuse relationships. With input and output configuration options.

● WIFI protocol and function

➢ Support GB15629.11-2006, IEEE802.11 b/g/n;

➢ Support WMM/WMM-PS/WPA/WPA2/WPS

➢ Support WiFi Direct;

➢ Support EDCA channel access mode;

➢ Support 20/40M bandwidth working mode;

➢ Support STBC, GreenField, Short-GI, support reverse transmission;

➢ Support RIFS frame interval;

➢ Support AMPDU, AMSDU;

➢ Support 802.11n MCS 0~7, MCS32 physical layer transmission rate gear, the transmission rate is up to 150Mbps;

➢ Support HT-immediate Compressed BlockAck, normal ACK, no ACK response mode;

➢ Support CTS to self;

➢ Support AP function; AP and STA are used at the same time;

➢ In the BSS network, multiple multicast networks are supported, and each multicast network supports different encryption methods.

32 multicast networks and network access STA encryption;

➢ When the BSS network supports as an AP, the total number of supported sites and groups is 32;

➢ Receive Sensitivity:
 20MHz MCS7@-71dBm@10%PER;
 40MHz MCS7@-67dBm@10%PER;
 54Mbps@-73dBm@10%PER;
 11Mbps@-86dBm@8%PER;
 1Mbps@-96dBm@8%PER;
➢ Support a variety of different received frame filtering options;
➢ Support monitoring function;
● Bluetooth protocol and function
➢ Integrated Bluetooth baseband processor/coprocessor, support BT/BLE4.2 protocol
➢ Support various rates of DR/EDR;
➢ Support BLE 1Mbps rate;
● Power supply and power consumption
➢ 3.3V single power supply;
➢ Support Wi-Fi power saving mode power management;
➢ Support work, sleep, standby, shutdown working modes;
➢ Standby power consumption is less than 15uA;

3 Overview

This chip is a SOC chip that supports multi-interface, multi-protocol wireless local area network 802.11n (1T1R). The SOC chip integrates
RF Transceiver, CMOS PA Power Amplifier, Baseband Processor/Media Access Control, SDIO, SPI,
Low-power WLAN chip with interfaces such as UART and GPIO.
W800 chip supports GB15629.11-2006, IEEE802.11 b/g/n protocol, and supports STBC, Green Field,
Short-GI, Reverse Transmission, RIFS Interframe Interval, AMPDU, AMSDU, T-immediate Compressed Block Ack,
Rich protocols and operations such as normal ACK, no ACK, and CTS to self.
The W800 chip integrates the RF transceiver front-end, A/D and D/A converters. It supports DSSS (Direct Sequence Spread Spectrum) as well as OFDM
(Orthogonal Frequency Division Multiplexing) modulation mode, with data descrambling capability, supports a variety of different data transmission rates. in the analog front end of the transceiver
The equipped transceiver AGC function enables the system-on-a-chip to obtain the best performance. The W800 chip also includes a built-in enhanced signal monitor,
The influence of multipath effect can be largely eliminated.
In terms of security, the W800 chip not only supports the national standard WAPI encryption, but also supports the international standard WEP, TKIP, CCMP encryption,
These hardware components enable data transmission systems based on the chip to obtain data similar to those of non-encrypted communications during secure communications.
data transmission performance.
In addition to supporting the energy-saving operations specified by the IEEE802.11 and Wi-Fi protocols, the W800 chip also supports user-customized energy-saving solutions. chip
It supports four working modes: work, sleep, standby and shutdown, so that the whole system can achieve low power consumption, and it is convenient for users to adapt to their own use fields.
different scenarios for energy saving.
The W800 chip integrates a high-performance 32-bit embedded processor, a large number of memory resources, and a wealth of peripheral interfaces, which are convenient for users
It is easy to apply the chip to the secondary development of a specific product.
The W800 chip supports the AP function, which can realize the establishment of 5 SSID networks at the same time, and realize the function of 5 independent APs. Support for creating multiple
Multicast network function. It can realize the function of establishing a BSS network as an AP while joining other networks as an STA.
The W800 chip supports the WPS method, so that users can realize an encrypted complete network with one-click operation to ensure the security of information
sex.
The multi-function and high integration of the W800 chip ensures that the WLAN system does not need too many off-chip circuits and external memory.

4 Chip Structure

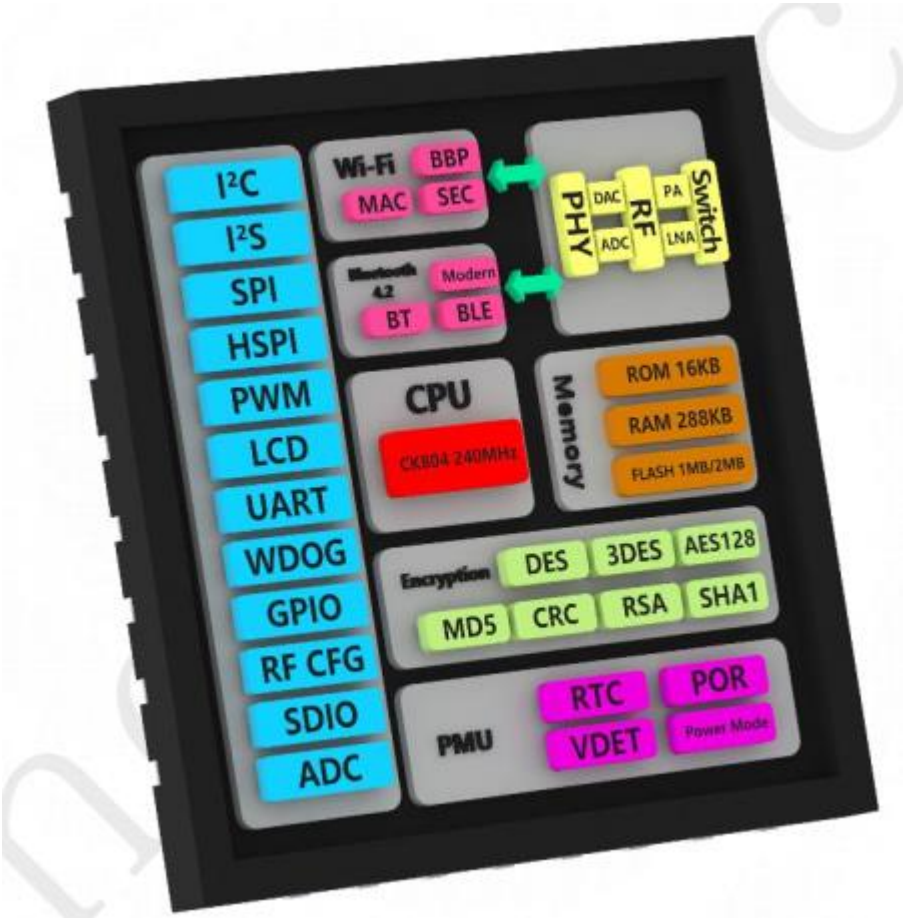
4.1 Chip structure

The following figure describes the overall structure of the W800 chip, the core part includes the XT804 CPU, 288KB SRAM and 20KB ROM storage space.

As the constant power supply module of the chip, the PMU part provides power-on sequence management, start-up clock, and real-time clock functions. Provides a wealth of peripherals

function and hardware encryption and decryption functions. The Wi-Fi part integrates MAC, BB and RF.

Figure 1 W800 chip structure diagram



4.2 Bus Structure

The W800 chip consists of a two-level bus, as shown in the figure below

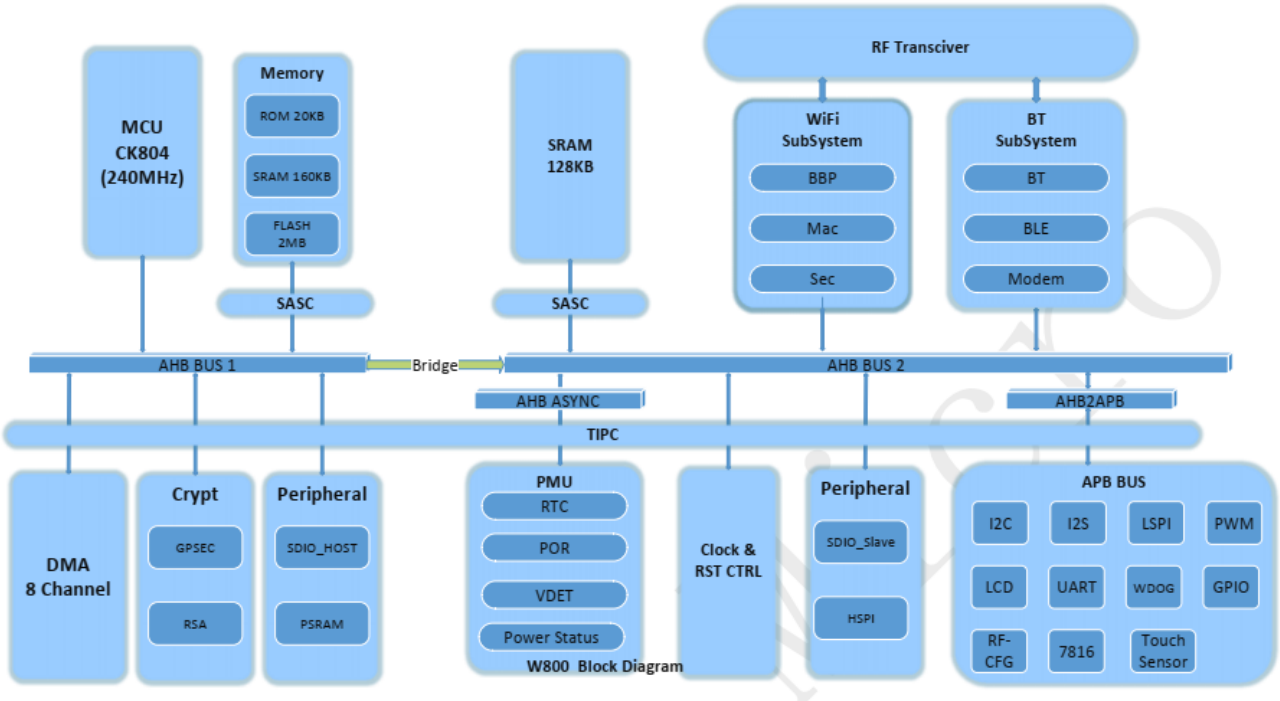


Figure 2 W800 bus structure diagram

(1) AHB-1 bus

This bus has four masters - XT804, DMA, GPSEC and 5 slaves.

XT804 is a 32-bit high-efficiency embedded CPU core for the control field, using a 16/32-bit mixed coding instruction system, designed a streamlined and efficient 3-stage pipeline.

The XT804 provides a variety of configurable functions, including hardware floating point unit, on-chip cache, DSP acceleration unit, trusted protection technology, on-chip tightly coupled IP, etc., users can configure according to application needs. In addition, XT804 provides multi-bus interface, supports system bus, finger flexible configuration of the order bus and data bus. XT804 has made special acceleration for interrupt response, and the interrupt response delay only needs 13 cycles.

The bus clock operates at the fastest frequency of 240MHz and can be configured to 240/160/120/80/40MHz, or lower.

Table 1 AHB-1 bus master list

Main device	function
CPU	Complete chip register configuration, memory management and use, and complete 802.11MAC protocol. Highest operating frequency 240MHz
DMA	Independent 8-channel DMA module supporting linked list structure, supporting 16 on-chip hardware DMA request sources.
GPSEC	Universal encryption module, supports DES/3DESSHA1/AES/MD5/RC4/CRC/PRDN. automatic completion data blocks in the specified memory space are encrypted and written back.

Table 2 List of AHB-1 Bus Slaves

Slave	Function
ROM	ROM is used to store the initialization firmware after the CPU is powered on. It mainly completes the initial configuration of the chip register space and other work. After completing the above work, the CPU control The control is given to the firmware stored in FLASH.
AHB2AHB	Complete the conversion of CPU bus clock domain to BusMatrix2 bus clock domain master access. Require The clock domains must be of the same origin, and the ratio of the CPU clock to the BusMatrix2 clock frequency is M:1, M is an integer greater than or equal to 1.
FLASH	Store firmware code and operating parameters
SRAM 160KB	Can be used to hold instructions or data, and firmware can use this memory as needed.
RSA	Supports RSA encryption and decryption operations up to 2048bit
GPSEC	Universal encryption/decryption module, supports SHA1/AES/MD5/RC4/CRC/TRNG. autocomplete Encrypt/decrypt and write back data blocks in a given memory space.
SDIO_HOST	SDIO 2.0 standard SDIO HOST controller; SDIO interface peripherals can be accessed through this interface. The SDIO interface clock is obtained by dividing the bus clock and supports up to 50MHz.
PSRAM_CTR	PSRAM controller with QSPI interface. An external PSRAM can be accessed through this controller. QSPI connection The port clock is obtained by dividing the frequency of the bus clock and supports a maximum clock of 80MHz.

2) AHB-2 bus

This bus has 4 master devices and 3 slave devices. Using the crossbar connection structure, different master devices can connect to different slave devices.

access at the same time, thereby increasing the bandwidth. The bus clock operates at the fastest frequency of 40MHz and can be configured lower as required.

Table 3 AHB-2 bus master list

Main device	function
MAC	802.11MAC control protocol processing module. Operations on the bus mainly include sending read data Data, receive write data, and send write-back completion descriptors.
SEC	802.11MAC control protocol processing module. Operations on the bus mainly include sending read data Data, receive write data, and send write-back completion descriptors.

AHB2AHB	Transition of bus master access from the AHB-1 bus to the AHB-2 bus.
SDIO/HSPI	Connect the host to the chip through the SDIO2.0 device controller or high-speed SPI slave device controller Accesses translate to AHB bus signals and access content memory and register space.

Each master device adopts a fixed priority, and the priority decreases from top to bottom.

Table 4 AHB-2 bus slave list

Slave	Function
SRAM 128KB	Used to store upstream and downstream data buffer, SDIO/SPI/UART interface uses this RAM as data cache
Configuration	Register configuration space, high-speed module configuration registers are uniformly addressed here
APB	All low-speed modules access the space, and various low-speed modules are connected using the APB bus.
BT_CORE	Bluetooth controller.

4.3 Clock Structure

W800 uses 24/40MHz crystal as SoC clock source, built-in 1 DPLL output 480MHz, supply used by CPU, system bus, data bus and WiFi system; additionally built-in 32.768KHZ RC oscillator for PMU and LCD modules. An overview of the clock structure is shown in the figure below.

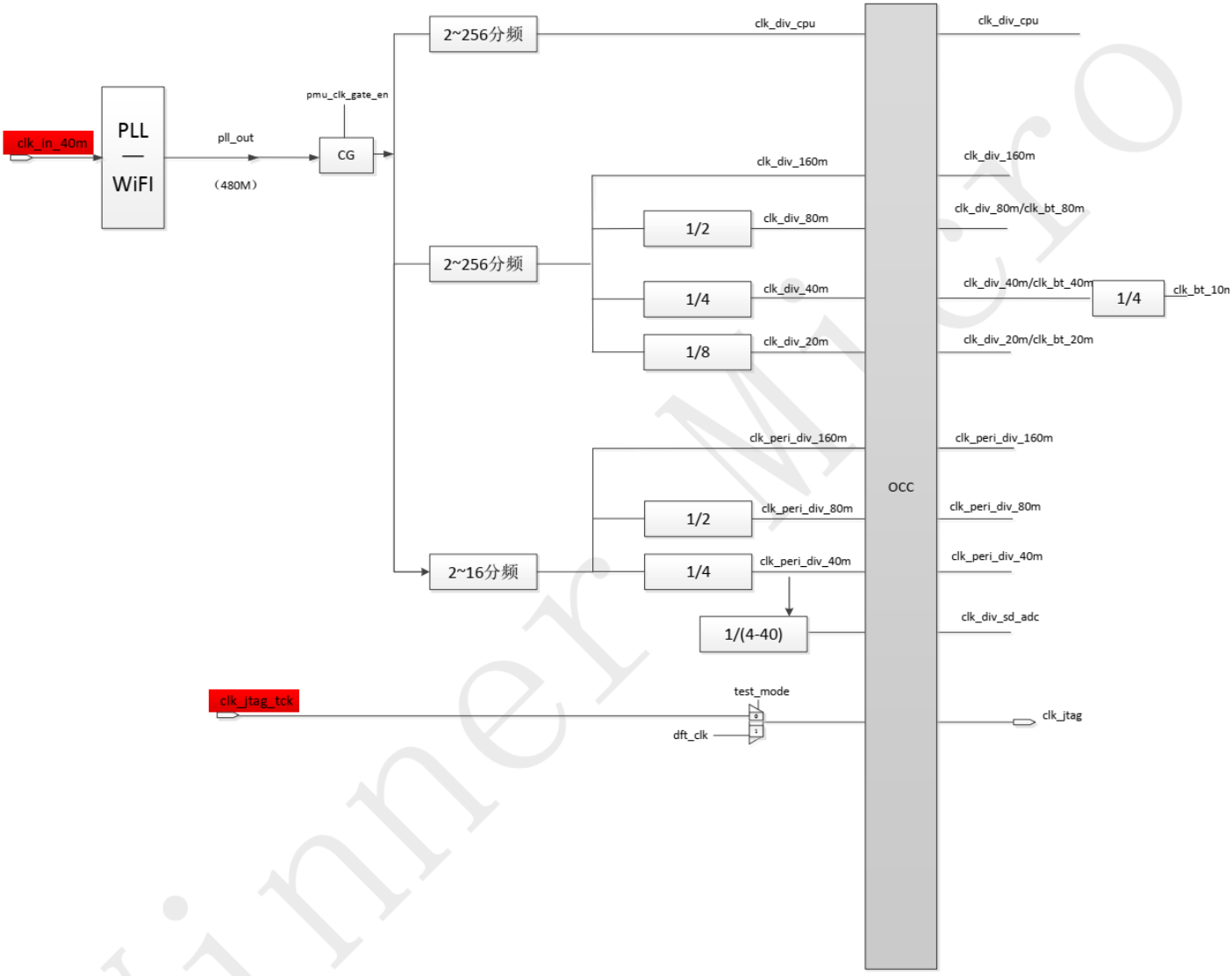
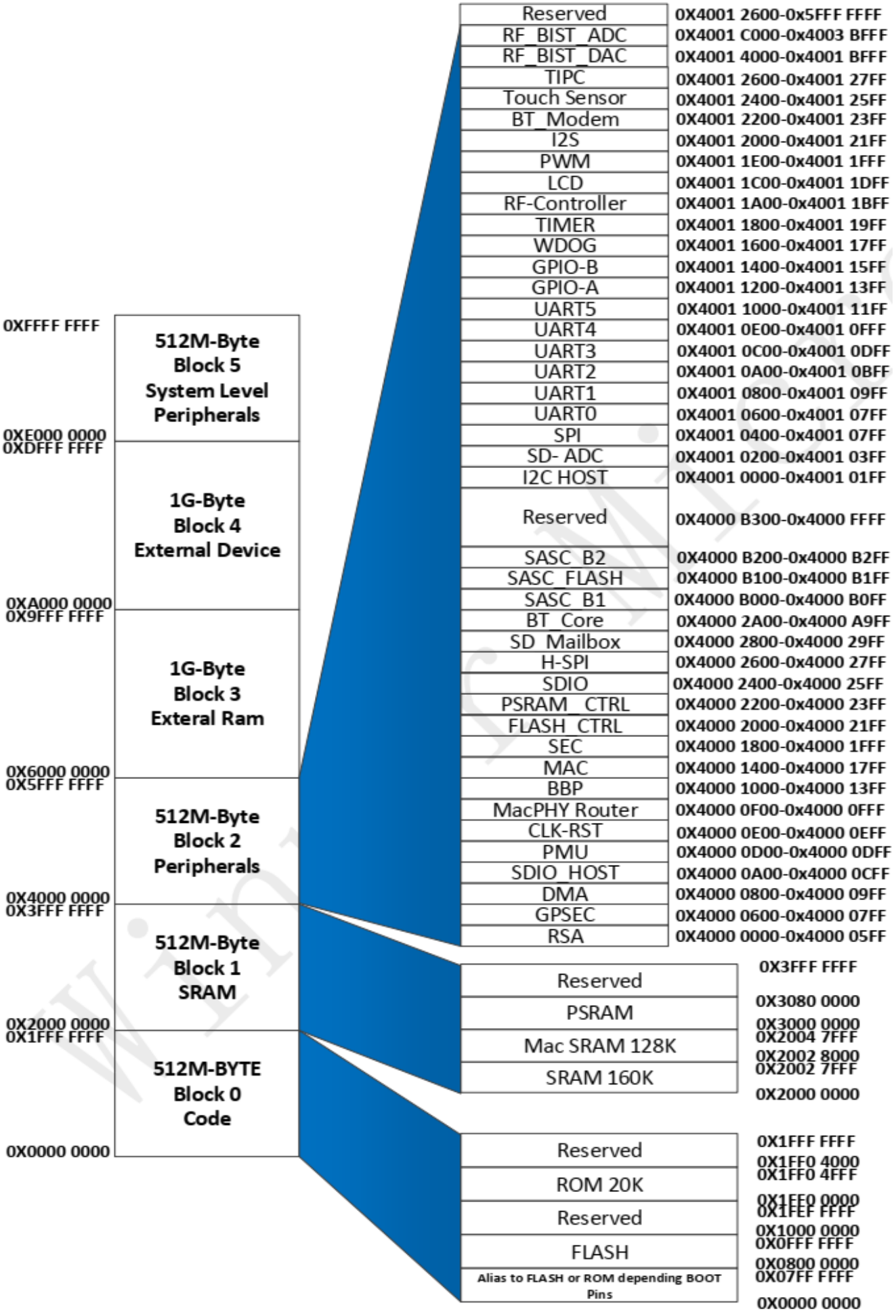


Figure 3 W800 Clock Structure

4.4 Address space



XT804 supports 4G storage space, which is divided into 6 blocks as shown in the figure above, which are code area, memory area, on-chip peripherals, and off-chip storage.

area, off-chip peripherals and system peripherals area. According to the requirements, the on-chip storage space of w800 is mapped to the first three areas as shown in Figure 3.

Table 5 Detailed division of bus device address space

bus from equipment	BootMode=0	Address space breakdown	Remark
ROM	0x0000 0000 ~ 0x0004 FFFF		Store the solidified firmware code
FLASH	0x0800 0000 ~ 0x0FFF FFF		Stored as a dedicated instruction device.
SRAM	0x2000 0000 ~ 0x2002 7FFF		Firmware memory and instruction storage Area
Mac RAM	0x2002 8000 ~ 0x2004 7FFF		SDIO/H-SPI/UARTdata cache
PSRAM	0x3000 000~0x3080 0000		Peripheral memory
CONFIG	0x4000 0000 ~ 0x4000 2FFF	0x4000 0000 ~ 0x4000 05FF	RSA configuration space
		0x4000 0600 ~ 0x4000 07FF	GPSEC configuration space
		0x4000 0800 ~ 0x4000 09FF	DMA configuration space
		0x4000 0A00 ~ 0x4000 0CFF	SDIO_HOST configuration is empty between
		0x4000 0D00 ~ 0x4000 0DFF	PMU configuration space
		0x4000 0E00 ~ 0x4000 0EFF	Clock and Reset Configuration space
		0x4000 0F00 ~ 0x4000 0FFF	MacPHY Router configuration space
		0x4000 1000 ~ 0x4000 13FF	BBP configuration space
		0x4000 1400 ~ 0x4000 17FF	MAC configuration space
		0x4000 1800 ~ 0x4000 1FFF	SEC configuration space
		0x4000 2000 ~ 0x4000 21FF	FLASH Controller configuration space
		0x4000 2200 ~ 0x4000 23FF	PSRAM_CTRL configuration space
		0x4000 2400 ~ 0x4000 25FF	SDIO Slave configuration is empty between
		0x4000 2600 ~ 0x4000 27F	H-SPI configuration space
		0x4000 2800 ~ 0x4000 29FF	SD Wrapper configuration space
		0x4000 2A00 ~ 0x4000 A9FF	BT Core configuration space
		0x4000 B000 ~ 0x4000 B0FF	SASC-B1 Level 1 Bus Memory Security Con-figuration Mode piece
		0x4000 B100 ~ 0x4000 B1FF	SASC-Flash Flash Security Configu-ration Module
		0x4000 B200 ~ 0x4000 B2FF	SASC-B2 Secondary bus memory security configuration module piece
APB	0x4001 0000 ~ 0x 4001 C000	0x4001 0000 ~ 0x4001 01FF	I2C master
		0x4001 0200 ~ 0x4001 03FF	Sigma ADC
		0x4001 0400 ~ 0x4001 07FF	SPI master
		0x4001 0600 ~ 0x4001 07FF	UART0
		0x4001 0800 ~ 0x4001 09FF	UART1
		0x4001 0A00 ~ 0x4001 0BFF	UART2

		0x4001 0C00 ~ 0x4001 0DFF	UART3
		0x4001 0E00 ~ 0x4001 0FFF	UART4
		0x4001 1000 ~ 0x4001 11FF	UART5
		0x4001 1200 ~ 0x4001 13FF	GPIO-A
		0x4001 1400 ~ 0x4001 15FF	GPIO-B
		0x4001 1600 ~ 0x4001 17FF	WatchDog
		0x4001 1800 ~ 0x4001 19FF	Timer
		0x4001 1A00 ~ 0x4001 1BFF	RF_Controller
		0x4001 1C00 ~ 0x4001 1DFF	LCD
		0x4001 1E00 ~ 0x4001 1FFF	PWM
		0x4001 2000 ~ 0x4001 22FF	I2S
		0x4001 2200 ~ 0x4001 23FF	BT-modem
		0x4001 2400 ~ 0x4001 25FF	Touch Sensor
		0x4001 2600 ~ 0x4001 25FF	TIPC Interface Security Settings
		0x4001 4000 ~ 0x4000 BFFF	RF_BIST DAC transmit RAM
		0x4001 C000 ~ 0x4003 BFFF	RF_BIST ADC receive RAM
		0x4001 3C00 ~ 0x5FFF FFFF	RSV

4.4.1 SRAM

W800 has built-in 288KB SRAM. Among them, 160KB is mounted on the first-level AHB bus, and 128KB is mounted on the second-level AHB bus. CPU

Devices on the first-level bus can access all memory areas, but devices on the second-level bus can only access 128KB of memory on the second-level bus.

4.4.2 Flash

4.4.2.1 QFlash

W800 integrates 2MBytes QFlash inside. The XIP method is implemented on the QFlash through the integrated 32KB cache inside the chip.

sequence. When the program is running, the CPU first reads the instruction from the Cache, and when the instruction cannot be obtained, it reads the instruction from the

QFlash reads the instruction and stores it in the Cache. Therefore, when the continuous running code size is less than 32K, the CPU will not need to read from QFlash

Instructions are fetched, at which point the CPU can run at a higher frequency. The above method is the operation mode of the read command, and the RO segment of the entire Image will be

Operate in this way. This process requires no user intervention.

QFlash can also store data. When the user program needs to read and write data in QFlash, it needs to be performed by the built-in QFlash controller.

Operation, QFlash provides the corresponding address, instruction and other registers to assist the user to achieve the desired operation. For specific description, please refer to QFlash

The controller corresponds to the chapter.

Users need to pay attention that when the program reads or writes data, there is no need to perform state judgment, wait and other operations, because the QFlash control

The device itself will judge. When the QFlash controller returns, the read or write is complete.

4.4.2.2 SPI Flash

In addition to supporting the 6PIN QFlash interface (built-in PIN, not packaged), the W800 chip also supports low-speed SPI interface access. The SPI maximum operating frequency of the interface can reach 20MHz, and it supports the master-slave function. For a detailed description, please refer to the corresponding chapter of the SPI interface.

4.4.3 PSRAM

W800 has a built-in PSRAM controller with SPI/QSPI interface, supports external PSRAM device access with a maximum capacity of 64Mb, and provides a bus mode of PSRAM read, write and erase operations. The maximum read and write speed is 80MHz. When the storage capacity needs to be expanded, the off-chip PSRAM can be used to expand fill code storage space or data storage space. PSRAM also supports program execution in XIP mode, and CPU Cache also supports cache data in PSRAM.

4.5 Startup Configuration

After the W800 chip is powered on, the CPU will start to execute the firmware in the ROM and load the user image at the specified address in the Flash.

When the ROM firmware starts to run, it will read the BootMode (PA0) pin, and judge to enter the boot state according to the signal of the pin:

Table 6 Startup Configurations

BootMode	Start condition	Start mode
high		normal startup process
Low	Continuous <30ms, quick test mode is invalid	normal startup process
	Last >=30ms	Enter functional mode
Note: Test mode: chip test function, user cannot operate. Function mode: Enter the basic functions implemented by ROM, such as: downloading firmware, programming MAC address, etc. For details, please refer to "WM_W800_ROM Function Brief.pdf"		

Typically, the BootMode pins should be used in production or debug stages. During production, the user continuously pulls the BootMode pin

If it is lower than 30ms, it will enter the function mode, which can quickly burn the Flash.

In the scenario of product rework or repair, the chip does not enter the "highest security level" (for the description of the security level, please refer to

"WM_W800_ROM Function Brief"), you can enter the function mode through this pin, erase the old Image, write the new

Image.

In the debugging stage, no matter what the firmware is faulty, you can enter the serial port by continuously pulling the BootMode pin down for more than 30ms

Download function, burn new firmware.

5 Clock and reset module

5.1 Function overview

The clock and reset module completes the software control of the chip clock and reset system. Clock control includes clock frequency conversion, clock shutdown and self-adaptation

should be gated; reset control includes soft reset control of the system and sub-modules.

5.2 Main Features

- Supports clock shutdown of each module
- Support some modules clock adaptive shutdown
- Support software reset of each module
- Support CPU frequency setting
- Support ADC/DAC loopback test
- Support I2S clock setting

5.3 Functional Description

5.3.1 Clock Gating

By configuring the clock gating enable register CLK_GATE_EN, the clock of the specified function can be controlled to shut down, so as to shut down the function of a certain module.

able purpose.

In order to provide the flexibility of firmware to control the power consumption of the system, the clock and reset module provides the clock gating function of each module in the system. when closed

When the clock of the corresponding module is stopped, the digital logic and clock tree of the module will stop working, which can reduce the dynamic power consumption of the system.

The switch of each module corresponds to the detailed description of the register SW_CLKG_EN.

5.3.2 Clock Adaptive Shutdown

The chip adaptively shuts down the clocks of certain functional modules according to the transition of certain internal states.

Users, please do not change the configuration, otherwise it may cause system abnormality when configuring the PMU function.

5.3.3 Function reset

The chip provides the soft reset function of each subsystem, and the subsystem reset can be achieved by setting the corresponding BIT of SW_RST_CTRL to 0.

However, the reset state will not be cleared automatically, and the corresponding BIT bit of SW_RST_CTRL needs to be set to 1 to resume normal operation.

The soft reset function does not reset the CPU and WatchDog.

In this register, the reset operation of APB/BUS1/BUS2 (corresponding to APB bus, system bus and data bus) is not recommended, which will cause the system to fail.

The system access device is abnormal.

5.3.4 Clock division

The W800 system uses 40MHz/24MHz crystal as the system clock source, the system has built-in DPLL, and the fixed output 480MHz clock is used as the clock source.

The clock source of the whole system (as shown in the figure below).

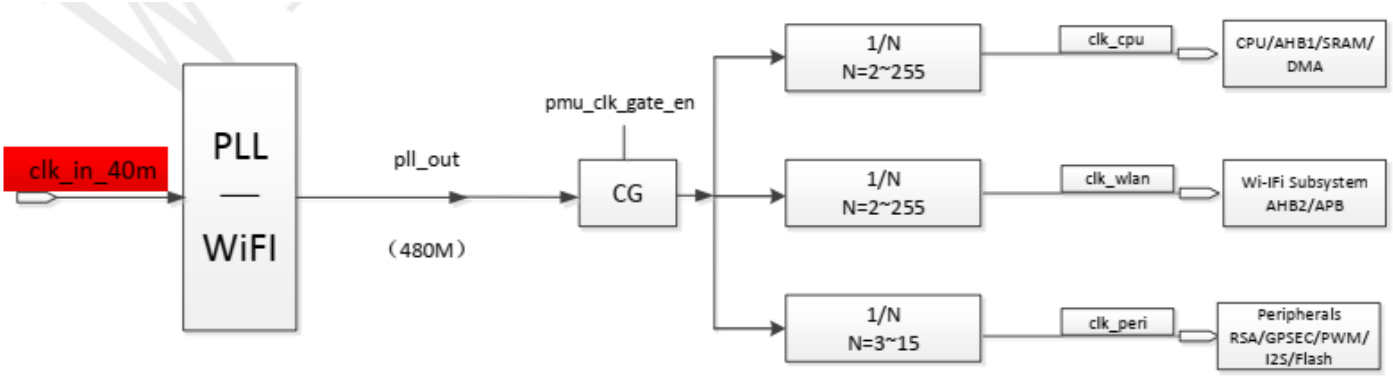


Figure 4 System clock frequency division relationship

The clock of the system bus is the same as the CPU clock, and the clock of the data bus is fixed at 1/4 of the WLAN root clock.

The WLAN root clock is also the clock source for the entire WLAN system.

This module provides the function of setting CPU clock and WLAN root clock for firmware to adjust system performance and power consumption.

Set the BIT[7:0] of the SYS_CLK_DIV register to adjust the CPU clock division factor. The source clock of the CPU clock frequency division is the DPLL output, fixed at 480MHz. The default value of the CPU clock frequency division factor is 6, that is, the default operating frequency of the CPU is 480MHz divided by 6, i.e. 80MHz. This parameter can be reconfigured when the clock required by the CPU needs to be adjusted.

The CLK_PERI clock provides the root clock of the operating clock of the cryptographic modules in the SoC system, as well as the root clock of the operating clock of some interfaces, more than such as PWM interface, I2S interface, Flash interface clock. This clock is also derived by dividing the 480MHz output from the DPLL. normal working condition

The lower frequency should be fixed at 3 to get the CLK_PERI root clock of 160MHz. Divide by 2 and divide by 4 by CLK_PERI root clock

80MHz and 40MHz are provided for encryption module and interface module.

Set the BIT[15:8] of the SYS_CLK_DIV register to adjust the WLAN clock frequency division factor. The default divide factor is 3, i.e. for DPLL 480MHz output is divided by 3 to get a 160MHz clock, which is sent to the WLAN as the root node clock (the WLAN continues to divide the frequency to get a more

For the detailed low frequency clock used by the WLAN system.

Note: If you want the WLAN system to work normally, the WLAN root clock needs to be kept at 160MHz, otherwise the WLAN system will fail.

When the WLAN system is not required to work, the WLAN root clock can be lowered to reduce the dynamic power consumption of the system.

When changing the system clock configuration, it should be noted that the ratio of the system bus to the data bus needs to be maintained at M:1, where M is an integer,

The minimum is 1. When changing the system clock configuration, it is also necessary to update the BIT [23:16] of the register SYS_CLK_DIV at the same time, set the correct ratio

example coefficient. Otherwise, accessing the data bus will result in abnormal data.

[15:8] of SYS_CLK_SEL provides the frequency division factor for setting the operating frequency of SAR_ADC, which is divided by 40M as the clock source. Frequency division

The number is the assigned frequency division value.

BIT[4] of SYS_CLK_SEL is to configure the clock frequency selection of the core operation of the RSA module, which can be 80MHz or 160MHz.

BIT[5] is to configure the clock frequency selection of the core operation of the GPSEC module, which can be 80MHz or 160MHz.

BIT[6] is to configure the clock frequency selection of the external bus of the FLASH module, which can be 40MHz or 80MHz.

When you need to reconfigure cpu_clk_divider, wlan_clk_divider, bus2_syncdn_factor, sdadc_fdiv, you need to set

BIT[31] of SYS_CLK_DIV, the hardware automatically updates the above four parameters to the frequency divider, and then clears BIT[31].

I2S_CLK_CTRL provides the clock configuration function of the I2S module.

5.3.5 Debug function control

The user can enable and disable the JTAG function by setting the value of DEBUG_CTRL (SYS_CLK_SEL-BIT[16]).

5.4 Register Description

5.4.1 Register List

Table 7 Clock Reset Module Register List

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x0000	Software Clock Gating Enable Register	**SW_CLKG_EN**	RW	Whether the software configuration module shuts down the clock	0x0000_7FFF
0x0004	Software Clock Mask Register	**SW_CLK_MASK**	RW	Whether the software configuration module is automatically shut down Bell	0x0000_007E
0x0008	Reserved				
0x000C	Software Reset Control Register	**SW_RST_CTRL**	RW	Software configuration reset module	0x01FF_FFFF
0x0010	Clock Divider Configuration Register	**SYS_CLK_DIV**	RW	Configure the clock divider ratio	0x0000_2212
0x0014	Debug Control Register	**DEBUG_CTRL**	RW	Configure ADC/DAC loopback test	0x0000_0000
0x0018	I2S Clock Control Register	**I2S_CLK_CTRL**	RW	Configure the I2S clock	0x0000_0000
0x001C	Reset status register	**RESET_STATUS**	RW	View CPU Soft Reset and Watchdog	0x0000_0000

5.4.2 **SW_CLKG_EN** - Software Clock Gating Enable Register.

Bit	Access	Operation Description	Reset Value
[31:22]	RO	Reserved	
[21]	RW	soft_touch_gate_en Configure the gating of the touch_sensor module clock. By default, the gating of the touch_sensor module is enabled. 0: touch_sensor module clock is off 1: touch_sensor clock on	1'b1
[20]	RW	soft_bt_gate_en	1'b1

		Configure the gating of the BT./BLE module clock. By default, the gating of the BT/BLE module is enabled. 0: BT/BLE module clock is off 1: BT/BLE clock on	
[19]	RW	soft_qsram_gate_en Configure the gating of the qspi_ram module clock. By default, the gating of the qspi_ram module is enabled. 0: qspi_ram module clock is off 1: qspi_ram clock on	1'b1
[18]	RW	soft_sdio_m_gate_en Configure the gating of the clock of the sdio_master module. By default, the gating of the sdio_master module is enabled. 0: sdio_master module clock is off 1: sdio_master clock on	1'b1
[17]	RW	soft_gpsec_gate_en Configure gpsec module clock gating, gpsec module gating is enabled by default 0: gpsec module clock is off 1: gpsec clock on	1'b1
[16	RW	soft_rsa_gate_en Configure the gating of the RSA clock, the default RSA gating is enabled 0: RSA module clock is off 1: RSA clock on	1'b1
[15]	RW	soft_i2s_gate_en 1'b162 Configure the gating of the i2s clock, the default i2s gating is enabled 0: i2s clock is off 1: i2s clock on	1'b1
[14]	RW	soft_lcd_gate_en Configure the gating of the lcd clock, the default lcd gating is enabled 0: LCD clock off 1: LCD clock on	1'b1
[13]	RW	Soft_pwm_gate_en Configure the gating of the pwm clock, the default pwm gating is enabled 0: pwm clock off 1: pwm clock on	1'b1
[12]	RW	soft_sd_adc_gate_en Configure the gating of sd_adc_clock, by default sd_adc_gating is enabled 0: sd_adc_clock off 1: sd_adc_clock on	1'b1
[11]	RW	soft_gpio_gate_en Configure the gating of the GPIO clock, the default GPIO gating is enabled 0: GPIO clock off 1: GPIO clock on	1'b1
[10]	RW	soft_timer_gate_en Configure the gating of the timer clock, the default timer gating is enabled 0: timer clock off 1: timer clock on	1'b1
[9]	RW	soft_rf_cfg_gate_en: used internally, do not modify Configure the gating of the rf_cfg clock, the default rf_cfg gating is enabled 1'b0: rf_cfg clock off 1'b1: rf_cfg clock on	1'b1
[8]	RW	soft_dma_gate_en Indicates whether the clock supplied to the dma clock domain is turned off 1'b0: dma clock off 1'b1: dma clock on	1'b1
[7]	RW	soft_ls_spi_gate_en Configure the gating of the low-speed spi clock, the default low-speed spi gating is enabled 1'b0: low speed spi clock off 1'b1: low-speed spi clock on	1'b1
[6]	RW	soft_uart5_gate_en Configure the gate control of uart5, default uart5 is enabled	1'b1

		0: uart5 is off 1: uart5 is on	
[5]	RW	soft_uart4_gate_en 0: uart4 is off 1: uart4 is on	1'b1
[4]	RW	soft_uart3_gate_en 0: uart3 is off 1: uart3 is on	1'b1
[3]	RW	soft_uart2_gate_en 0: uart2 is off 1: uart2 is on	1'b1
[2]	RW	soft_uart1_gate_en 0: uart1 is off 1: uart1 is on	1'b1
[1]	RW	soft_uart0_gate_en 0: uart0 is off 1: uart0 is on	1'b1
[0]	RW	soft_i2c_gate_en Configure i2c clock gating, i2c gating is enabled by default 1'b0: i2c clock off 1'b1: i2c clock on	1'b1

5.4.3 **SW_CLK_MASK** - Software Clock Mask Register.

Table 9 Software Clock Mask Register

Bit	Access	Operation Description	Reset Value
[6]	RW	soft_cpu_clk_gt_mask Indicates whether the clock supplied to the CPU clock domain (including CPU, bus1, ROM, SRAM) can be adaptive Shutdown (when the CPU needs to enter the WFI state, do not set the adaptive shut-down) 1'b0: Allows adaptive turn-off and turn-on 1'b1: Adaptive turn-off and turn-on are not allowed	1'b1
[5 : 2]	RW	Reserved for internal use, do not modify	
[1]	RW	soft_sdioahb_clk_gt_mask Indicates whether the clock supplied to the sdio ahb clock domain can be shut down adaptively 1'b0: Allows adaptive turn-off and turn-on 1'b1: Adaptive turn-off and turn-on are not allowed	1'b1
[0]	RW	soft_pmu_clk_gt_mask There is a gating unit after the clock output by pll, which is configured by this register to indicate whether it is allowed to be turned off by the PMU. 1'b0: Allows the PMU to shut down the gate unit, thereby shutting down the clock 1'b1: PMU is not allowed to shut down the gate unit	1'b1

5.4.4 **SW_RST_CTRL** - Software Reset Control Register.

Table 10 Software Reset Control Register

Bit	Access	Operation Description	Reset Value
[31]	RW	soft_touch_rst_n Software reset touch_sensor module 0: reset 1: Reset release	0
[30]	RW	soft_rst_flash_n Software reset Flash controller module 0: reset 1: Reset release	0
[29]	RW	soft_rst_bt_n Software reset BT module 0: reset	0

		1: Reset release	
[28]	RW	soft_rst_qspi_ram_n Software reset qspi_ram module 0: reset 1: Reset release	0
[27]	RW	soft_rst_sdio_m_n Software reset sdio_master module 0: reset 1: Reset release	0
[26]	RW	soft_rst_gpsec_n Software reset gpsec module 1'b0: reset 1'b1: reset release	0
[25]	RW	soft_rst_rsa_n Software reset RSA module 1'b0: reset 1'b1: reset release	0
[24]	RW	soft_rst_i2s_n Software reset i2s module 1'b0: reset 1'b1: reset release	0
[23]	RW	soft_rst_lcd_n software reset lcd module 1'b0: reset	0
[22]	RW	soft_rst_pwm_n software reset pwm module 1'b0: reset 1'b1: reset release	0
[21]	RW	soft_rst_sar_adc_n Software reset sar_adc module 1'b0: reset 1'b1: reset release	0
[20]	RW	soft_rst_timer_ Software reset timer 1'b0: reset 1'b1: reset release	0
[19]	RW	soft_rst_gpio_n software reset the gpio module 1'b0: reset 1'b1: reset release	0
[18]	RW	soft_rst_rf_cfg_n Software reset to configure the RF register module (for internal use, do not modify) 1'b0: reset 1'b1: reset release	0
[17]	RW	soft_rst_spis_n Software reset high-speed spi module 1'b0: reset 1'b1: reset release	0
[16]	RW	soft_rst_spim_n software reset low speed spi module 1'b0: reset 1'b1: reset release	0
[15]	RW	soft_rst_uart5_n Software reset on-chip uart5 module 1'b0: reset 1'b1: reset release	0
[14]	RW	soft_rst_uart4_n	0
[13]	RW	soft_rst_uart3_n	0
[12]	RW	soft_rst_uart2_n	0
[11]	RW	soft_rst_uart1_n	0
[10]	RW	soft_rst_uart0_n	0
[9]	RW	soft_rst_i2c_n Software reset on-chip i2c module 1'b0: reset 1'b1: reset release	0
[8[	RW	soft_rst_bus2_n Software reset on-chip bus2 module 1'b0: reset 1'b1: reset release	0
[7]	RW	soft_rst_bus1_n Software reset on-chip bus1 module 1'b0: reset 1'b1: reset release	0
[6]	RW	soft_rst_apb_n software reset abp bridge module	0

		1'b0: reset 1'b1: reset release	
[5]	RW	soft_rst_mem_mng_n Software reset mem_mng module (internal use, do not modify) 1'b0: reset 1'b1: reset release	0
[4]	RW	soft_rst_dma_n software reset dma module 1'b0: reset 1'b1: reset release	0
[3]	RW	soft_rst_sdio_ahb_n software reset sdio ahb clock domain module 1'b0: reset 1'b1: reset release	0
[2]	RW	soft_rst_sec_n Software reset security module (internal use, do not modify) 1'b0: reset 1'b1: reset release	0
[1]	RW	soft_rst_mac_n Software reset mac module (internal use, do not modify) 1'b0: reset 1'b1: reset release	0
[0]	RW	soft_rst_bbp_n Software reset bbp module (internal use, do not modify) 1'b0: reset 1'b1: reset release	0

5.4.5 **SYS_CLK_DIV** - Clock Divider Configuration Register.

Table 11 Clock Divider Configuration Register

Bit	Access	Operation Description	Reset Value
[31]		**divide_freq_en** When it is necessary to reconfigure cpu_clk_divider , wlan_clk_divider, bus2_syncdn_factor, When sdadc_fdiv, set this register, the hardware will automatically update the above four parameters to the frequency divider, and then clear this register	0
[30:28]		Reserved	
[27:24]	RW	Peripheral_divider 160M clock frequency division factor: Divided by the DPLL as the clock source. The frequency division factor is the assigned frequency division value. The divided output should be 160MHz. The DPLL output is 480MHz and should be configured to 3.	0x03
[23:16]	RW	bus2_syncdn_factor The clock ratio between bus1 and bus2 should be N:1 Among them, N is an integer. In actual adjustment, it mainly depends on the ratio of the operating frequency of the CPU to the clock frequency of bus2. Since the default cpu uses 80MHz clock and bus2 uses 40MHz clock, then N=2	0x02
[15:8]	RW	wlan_clk_divider The clock from the PLL is divided and sent to the wlan system. This register is the frequency division factor, which is >=2. The default frequency division factor is 3, that is, the 480MHz output of the pll is divided by 3, and the 160MHz clock is obtained as the root The node clock is sent to wlan (wlan continues to divide the frequency to obtain a more detailed low-frequency clock); Note 1: If the WLAN system needs to work normally, this clock needs to be fixed at 160MHz; if the WLAN system is turned off, Then this clock can be downclocked to save power. This clock must not be configured higher than 160MHz. Note 2: The secondary bus clock and APB clock are divided by four;	0x03
[7:0]	RW	cpu_clk_divider The clock from the PLL is divided and sent to the CPU. This register is the frequency division factor, which is >=2. The default frequency division factor is 6, that is, after the reset is released, the 480MHz clock output by the PLL is divided by 6 and sent to the cpu is an 80MHz clock. When you need to adjust the clock required by the cpu, you can reconfigure this parameter	0x06

5.4.6 **DEBUG_CTRL** - Debug Control Register

Table 12 Clock Select Register

Bit	Access	Operation Description	Reset Value
[16]	RW	JTAG enable 1'b0: Disable JTAG debugging 1'b1: Enable JTAG debug function	0
[15:8]	RW	sd_adc_div sigma-delta ADC clock division factor: Divide by 40MHz as the clock source. The frequency division factor is the assigned frequency division value. After configuring this register, Divide_freq_en in the register clk_divider must be configured to take effect;	d10
[7]	RW	RSV	0
[6]	RW	qflash_clk_sel QSPI_FLASH clock selection 1: Use 80MHz; 0: 40 Mhz	0
[5]	RW	gpsec_sel GPSEC clock selection 1: Use 160MHz; 0: use 80MHz;	0
[4]		rsa_sel RSA clock selection 1: Use 160MHz; 0: use 80MHz;	0
[3:0]	RW	reserved, do not modify	d0

5.4.7 **I2S_CLK_CTRL** - I2S Clock Control Register

Table 13 I2S Clock Control Register

Bit	Access	Operation Description	Reset Value
[31:18]		Reserved	
[17:8]	RW	BCLKDIV BCLK splitter: F_BCLK = F_I2SCLK / BCLKDIV Note: If EXTAL_EN is not selected and internal PLL is used then F_I2SCLK = F_CPU (same as CPU frequency). Assuming F_CPU = 160MHz, F_I2SCLK = external crystal frequency when WXTAL_EN is enabled, **BCLKDIV = round (F_I2SCLK/(Fs*W*F))** Where Fs is the sampling frequency of the audio data, and W is the word width; 10'b075 F = 1 when the data is mono; F = 2 when the data is stereo. For example, if the internal PLL is used and the data width is 24 bits, the format is stereo and the sampling frequency is 128KHz, BCLKDIV should be configured as (160 * 10e6 / 128 * 10e3 * 24 * 2) = 10'h1a.	0
[7:2]	RW	MCLKDIV If an external clock is selected, this MCLK divider is used to generate the appropriate MCLK frequency. F_mclk = F_I2SCLK / (2 * MCLKDIV); F_I2SCLK is the external clock when MCLKDIV = 0; F_mclk = F_I2SCLK when MCLKDIV >= 1; Note: F_mclk should be configured as 256 * fs, where fs is the sampling frequency.	0
[1]	RW	MCLKEN MCLK enable switch 1'b0: MCLK disabled	0

		1’b1: enable MCLK	
[0]	RW	EXTAL_EN External clock selection, choose whether to use internal I2S block clock or external clock 1'b0: Internal clock 1'b1: External clock Note: When using an external clock, the external clock must be 2 * N * 256 fs, where fs is the sampling frequency, and N must be an integer.	0

5.4.8 **RESET_STATUS** - Reset Status Register.

Bit	Access	Operation Description	Reset Value
[31:18]			
[17:8]	WO	CPU soft reset state clear Write 1 to clear CPU soft Reset Status.	0
[7:2]	WO	CPU soft reset state clear Write 1 to clear CPU soft Reset Status.	0
[1]	RO	CPU soft reset state 1: The CPU has generated a soft reset; 0: The CPU does not generate a soft reset;	0
[0]	RO	Wdog reset state 1: Wdog generates a Reset; 0: Wdpg does not generate a Reset	0

6 DMA module

6.1 Function overview

DMA is used to provide high-speed data transfer between peripherals and memory and between memory and memory. Can operate without any CPU

Fast data movement via DMA without The CPU resources saved in this way do not affect the operation of other instructions by the CPU.

DMA is mounted on the AHB bus, supports up to 8 channels, 16 hardware peripheral request sources, supports linked list structure and register control.

6.2 Main Features

- Amba2.0 standard bus interface, 8 DMA channels
- Support DMA operation based on memory linked list structure
- Supports 16 hardware peripheral request sources
- Support 1, 4-burst operation mode
- Support byte, half-word, word as unit transfer operation
- Support source and destination address unchanged or sequentially incremented or configurable to cycle operations within a predefined address range
- Supports data transfer methods from memory to memory, memory to peripherals, and peripherals to memory

6.3 Functional Description

6.3.1 DMA channel

W800 supports a total of 8 DMA channels, DMA channels do not interfere with each other and can run at the same time. Request different data streams can choose different DMA channel.

Each DMA channel is allocated in a different register address offset segment, you can directly select the address segment of the corresponding channel for configuration and use. No 78

The register configuration method of the same channel is exactly the same.

Table 15 DMA address assignment

DMA base address	0x4000 0800
DMA_CH0	offset (0x10~0x38)
DMA_CH1	offset (0x40~0x68)
DMA_CH2	offset (0x70~0x98)
…	…
DMA_CH7	offset (0x160~0x188)

6.3.2 DMA data flow

Eight DMA channels enable a unidirectional data transfer link between source and destination.

The source and destination addresses of DMA can be set to remain unchanged, incremented or cyclic after each DMA operation is completed:

- DMA_CTRL[2:1] controls how the source address changes after each DMA operation;
- DMA_CTRL[4:3] controls how the destination address changes after each DMA operation.

DMA can set the handling unit of byte, half-word and word, and the final quantity of data to be handled is an integer multiple of the handling unit.

DMA_CTRL[6:5] to set.

DMA can set how many units of data to transfer each time through burst, and choose to transfer 1 or 4 units at a time through DMA_CTRL[7]

Bit data, if DMA_CTRL[6:5] is set to word and burst is set to 4, then 4 words of data are transferred each time.

DMA can set the number of Bytes to start DMA transfer each time, the maximum is 65535 Bytes, which is set by DMA_CTRL[23:8]. 79

6.3.3 DMA Cycling Mode

The DMA loop address mode means that after the source and destination addresses of DMA are set, after the data transfer reaches the set loop boundary, it will jump to the loop start address, and this loop executes until the set transfer byte is reached.

The source and destination addresses of the round-robin address mode need to be set with the SRC_WRAP_ADDR and DEST_WRAP_ADDR registers, and passed WRAP_SIZE to set the loop length value.

6.3.4 DMA transfer mode

DMA supports 3 transfer modes:

- RAM to RAM

Both the source address and the destination address are configured as memory addresses to be transferred, DMA_MODE[0] is set to 0, software mode.

- Memory to Peripherals

The source address is set to the memory address, the destination address is set to the peripheral address, DMA_MODE[0] is set to 1, the hardware mode,

DMA_MODE[5:2] selects the peripheral used.

- Peripherals to memory

The source address is set to the peripheral address, the destination address is set to the memory address, DMA_MODE[0] is set to 1, the hardware mode, DMA_MODE[5:2] selects the peripheral used.

6.3.5 DMA peripheral selection

When using the transfer method of peripheral to memory or memory to peripheral, in addition to the corresponding peripheral needs to be set to DMA TX or RX, DMA_MODE[5:2] also needs to select the corresponding peripheral.

Note: Because there are 3 UART ports, when the UART uses DMA, it is necessary to select the corresponding port through UART_CH[1:0]. UART.

6.3.6 DMA linked list mode

DMA supports linked list working mode. Through the linked list mode, when DMA transfers the current linked list memory data, we can advance to the next

The data is filled in each linked list. After the DMA finishes moving the current linked list, it judges that the next linked list is valid, and can directly move the data of the next linked list.

The linked list method can effectively improve the efficiency of DMA and CPU cooperation.

Linked list operation mode: Set the DMA to linked list working mode through the DMA_MODE[1] register, and then set the DESC_ADDR register is the starting address of the linked list structure, and then enables DMA through the CHNL_CTRL register. When the DMA process finishes moving the current memory after that, the software notifies the DMA that there is still valid data in the linked list by setting the valid flag, and the DMA processes the data according to the valid flag of the linked list a data to be moved.

6.3.7 DMA Interrupts

DMA transfer completion or burst can generate interrupts, INT_MASK register can mask the interrupt corresponding to the DMA channel.

When the corresponding DMA interrupt is generated, the current interrupt status can be queried through the INT_SRC register, indicating what is currently generating the interrupt.

The corresponding status bit needs to be cleared by software by writing 1 to 0.

6.4 Register Description

6.4.1 Register List

Table 16 DMA register list

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x0000	interrupt mask register	**INT_MASK**	RW	Set DMA interrupts that need to be masked	0x0000_FFFF

0X0004	Interrupt Status Register	**INT_SRC**	RW	Indicates the interrupt status of the current DMA	0x0000_0000
0X0008	DMA channel selection register	**DMA_CH**	RW	Which UART to choose when UART peripheral	0x0000_0000
0X000C	reserved		RW		0X0000_0000
DMA CHNL0 registers					
0X0010	DMA Source Address Register	**SRC_ADDR**	RW	Source address of DMA transfer	0x0000_0000
0X0014	DMA Destination Address Register	**DEST_ADDR**	RW	The destination address of the DMA transfer	0x0000_0000
0X0018	DMA loop source start address register	**SRC_WRAP_ADDR**	RW	DMA transfer source address in circular mode	0x0000_0000
0X001C	DMA loop destination start address register device	**DEST_WRAP_ADDR**	RW	DMA transfer destination in circular mode site	0x0000_0000
0X0020	DMA Cycle Length Register	**WRAP_SIZE**	RW	DMA loop boundary in loop mode	0x0000_0000
0X0024	DMA Channel Control Register	**CHNL_CTRL**	RW	Current channel DMA start and stop	0x0000_0000
0X0028	DMA Mode Select Register	**DMA_MODE**	RW	DMA Mode Select Register	0x0000_0000
0X002C	DMA Data Flow Control Register	**DMA_CTRL**	RW	Set up DMA transfer data stream	0x0000_0000
0X0030	DMA transfer bytes register	**DMA_STATUS**	RO	Get the current number of bytes transferred	0x0000_0000
0X0034	DMA linked list entry address register	**DESC_ADDR**	RW	DMA linked list address entry address setting	0x0000_0000
0X0038	DMA current destination address register	**CUR_DEST_ADDR**	RO	The address of the current DMA operation	0x0000_0000
DMA CHNL1 registers					
0X0040 - 0X0068	Same as DMA CHNL0 registers				
DMA CHNL2 registers					
0X0070 - 0X0098	Same as DMA CHNL0 registers				
DMA CHNL3 registers					
0X00A0 - 0X00C8					
DMA CHNL4 registers					
0X00D0 - 0X00F8					
DMA CHNL5 registers					
0X0100 - 0X0128					
DMA CHNL6 registers					
0X0130 - 0X0158					
DMA CHNL7 registers					
0X0160 - 0X0188					

6.4.2 Interrupt Mask Register

Table 17 DMA Interrupt Mask Register

Bit	Access	Operating Instructions	reset value
[31:16]		reserved	
[15]	RW	channel7 transfer_done interrupt mask, active high.	1
[14]	RW	channel7 burst_done interrupt mask, active high.	1
[13]	RW	channel6 transfer_done interrupt mask, active high.	1
[12]	RW	channel6 burst_done interrupt mask, active high.	1
[11]	RW	channel5 transfer_done interrupt mask, active high.	1
[10]	RW	channel5 burst_done interrupt mask, active high.	1
[9]	RW	channel4 transfer_done interrupt mask, active high.	1
[8]	RW	channel4 burst_done interrupt mask, active high.	1

[7]	RW	channel3 transfer_done interrupt mask, active high.	1
[6]	RW	channel3 burst_done interrupt mask, active high.	1
[5]	RW	channel2 transfer_done interrupt mask, active high.	1
[4]	RW	channel2 burst_done interrupt mask, active high.	1
[3]	RW	channel1 transfer_done interrupt mask, active high.	1
[2]	RW	channel1 burst_done interrupt mask, active high.	1
[1]	RW	channel0 transfer_done interrupt mask, active high.	1
[0]	RW	channel0 burst_done interrupt mask, active high.	1

6.4.3 Interrupt Status Register

Table 18 DMA Interrupt Status Register

Bit	Access	Operating Instructions	reset value
[31:16]		reserved	
[15]	RW	channel7 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	0
[14]	RW	channel7 burst_done interrupt status, write 1 to clear 0. DMA burst completion generates an interrupt.	0
[13]	RW	channel6 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	0
[12]	RW	channel6 burst_done interrupt status, write 1 to clear 0. DMA burst completion generates an interrupt.	0
[11]	RW	channel5 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	0
[10]	RW	channel5 burst_done interrupt status, write 1 to clear 0. DMA burst completion generates an interrupt.	0
[9]	RW	channel4 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	0
[8]	RW	channel4 burst_done interrupt status, write 1 to clear 0. DMA burst completion generates an interrupt.	0
[7]	RW	channel3 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	0
[6]	RW	channel3 burst_done interrupt status, write 1 to clear 0. DMA burst completion generates an interrupt.	0
[5]	RW	channel2 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	0
[4]	RW	channel2 burst_done interrupt status, write 1 to clear 0. DMA burst completion generates an interrupt.	0
[3]	RW	channel1 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	0
[2]	RW	channel1 burst_done interrupt status, write 1 to clear 0. DMA burst completion generates an interrupt.	0
[1]	RW	channel0 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	0
[0]	RW	channel0 burst_done interrupt status, write 1 to clear 0. DMA burst completion generates an interrupt.	0

6.4.4 UART selection register

Table 19 UART Select Register

Bit	Access	Operating Instructions	reset value
[31:24]		reserved	
[23:8]	RW	dma req clear: Write 1 to each bit to clear the corresponding dma req request. Self-cleaning. For example, writing 1 to bit 23 will clear the 15th corresponding dma request in dma_sel; Writing 1 to bit 8 will clear the 0th dma request in dma_sel - uart_rx_req;	d0

[2:0]	RW	Uart dma channel selection: 3'd0: uart0 module dma channel access dma 3'd1: uart1 module dma channel access dma 3'd2: uart2/7816 module dma channel access dma 3'd3: uart3 module dma channel access dma 3'd4: uart4 module dma channel access dma 3'd5: uart5 module dma channel access dma	0x0

6.4.5 DMA Source Address Register

Table 20 DMA Source Address Register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	In acyclic mode, the destination address, peripheral address or memory address of DMA transfer	0x00

6.4.7 DMA loop source start address register

Table 22 DMA Loop Source Start Address Register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	In circular mode, the starting address of the source address, peripheral address or memory address of the DMA transfer	0x00

6.4.8 DMA loop destination start address register

Table 23 DMA Loop Destination Start Address Register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	In circular mode, the starting address, peripheral address or memory address of the destination address of DMA transfer	0x00

6.4.9 DMA Cycle Length Register

Table 24 DMA Cycle Length Register

Bit	Access	Operating Instructions	reset value
[31:16]	RW	In loop mode, the loop length of the DMA destination address. DMA moves data sequentially from the start address. When the number of bytes of data to be moved reaches this set value, it will jump Go to the cycle start address and continue to move data from the start address	0x00
[15:0]	RW	In loop mode, the DMA source address loop length.	0x00

6.4.10 DMA Channel Control Register

Table 25 DMA Channel Control Register

Bit	Access	Operating Instructions	reset value
[31:2]		reserved	
[1]	RW	dma_stop Stop dma operation, active high. The DMA will stop after completing the current burst operation and clear chnl_on at the same time. software should be based on chnl_on is 0 to determine that the dma has stopped completely	0
[0]	RW	chnl_on Start the current channel DMA conversion, active high.	0

		It is automatically cleared to 0 after the conversion or setting is stopped after Dma is completed.	

6.4.11 DMA Mode Select Register

Table 26 DMA Mode Select Register

Bit	Access	Operating Instructions	reset value
[31:7]		reserved	
[6]	RW	chain_link_en Valid in linked list mode, indicating whether dma continues to read and process subsequent chains after processing the first linked list surface. If it is 1, update next_desc_addr in the linked list and continue reading the next linked list until the linked list where vld is 0; if it is 0, processing stops after completing the current linked list.	0
[5:2]	RW	dma_sel Choice of 16 dma_reqs. 4'd0: uart rx dma req 4'd1: uart tx dma req 4'd2: pwm_cap0_req 4'd3: pwm_cap1_req 4'd4: LS_SPI rx dma req 4'd5: LS_SPI tx dma req 4'd6: SD_ADC chnl0 req 4'd7: SD_ADC chnl1 req 4’d8：SD_ADC chnl2 req 4’d9：SD_ADC chnl3 req 4’d10: I2S RX req 4’d11: I2S TX req 4’d12: SDIO_HOST req	0x00
[1]	RW	chain_mode 1'b0: use normal mode 1'b1: use linked list mode	0
[0]	RW	dma_mode 1'b0: Software mode. 1'b1: Hardware mode.	0

6.4.12 DMA Data Flow Control Register

Table 27 DMA Data Flow Control Register

Bit	Access	Operating Instructions	reset value
[31:24]		reserved	
[23:8]	RW	total_byte The total number of bytes to operate on. It needs to be consistent with the data_size configuration, that is, if it is a word operation, then Should be configured as a multiple of 4; if it is a halfword operation, it should be configured as a multiple of 2.	0x00
[7]	RW	burst_size Set how many units of data the DMA transfers at a time 1'b0: burst is 1 1'b1: burst is 4 When the last burst size exceeds the number of remaining transfers, use the burst size as the size of the remaining data small.	0
[6:5]	RW	data_size Set the handling unit for DMA 2'h0: byte	0x00

		2'h1: half_word 2'h2: word 2'h3: reserved	
[4:3]	RW	dest_addr_inc 2'h0: The destination address remains unchanged after each operation; 2'h1: The destination address is automatically accumulated after each operation. 2'h2: reserved 2'h3: Loop operation, the destination address is automatically accumulated after each operation, and it jumps to the start of the loop when it reaches the defined loop boundary. starting address.	0x00
[2:1]	RW	src_addr_inc 2'h0: The source address remains unchanged after each operation; 2'h1: The source address is automatically accumulated after each operation. 2'h2: reserved 2'h3: Loop operation, the source address is automatically accumulated after each operation, and jumps to the start of the loop when it reaches the defined loop boundary address.	0x00
0	RW	auto_reload When the current DMA transfer is completed, the next DMA transfer will be automatically performed according to the current DMA configuration.	0

6.4.13 DMA transfer bytes register

Table 28 DMA Transfer Bytes Register

Bit	Access	Operating Instructions	reset value
[31:16]		reserved	
[15:0]	RW	transfer_cnt The number of bytes currently transferred. Each time the DMA is turned on again (chnl_on is set to 1), it is cleared to 0 and starts counting again.	0x00

6.4.14 DMA linked list entry address register

Table 29 DMA linked list entry address register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	desc_addr When the linked list is enabled, it is used as the entry address of the linked list. After each transfer of the linked list is completed, the base of the next linked list is address is updated to this register.	0x00

6.4.15 DMA current destination address register

Table 30 DMA current destination address register

Bit	Access	Operating Instructions	reset value
[31:0]	RO	current_dest_addr The current DMA operation destination address. When the software stops the DMA, the destination address that the DMA will operate can be known by looking at this register.	0x00

7 Universal hardware encryption module

7.1 Function overview

The encryption module automatically completes the encryption of the source address space data of the specified length, and automatically writes the encrypted data back to the specified destination address after completion time; supports SHA1/MD5/RC4/DES/3DES/AES/CRC/TRNG.

7.2 Main Features

- Support SHA1/MD5/RC4/DES/3DES/AES/CRC/TRNG encryption algorithm
- DES/3DES supports ECB and CBC modes
- AES supports ECB, CBC and CTR modes
- CRC supports four modes: CRC8, CRC16_MODBUS, CRC16_CCITT and CRC32
- CRC supports input/output reverse
- SHA1/MD5/CRC supports continuous multi-packet encryption
- Built-in true random number generator, also supports seed to generate pseudo-random numbers

7.3 Functional Description

7.3.1 SHA1 encryption

The hardware SHA1 calculation can be performed on consecutive multiple packets of data, the calculation result is stored in the register, and the encryption result of the previous packet can be used as the encryption result of the next packet initial value.

7.3.2 MD5 encryption

Hardware MD5 calculation can be performed on consecutive multiple packets of data, the calculation result is stored in the register, and the encryption result of the previous packet can be used as the initial value of the next packet initial value.

7.3.3 RC4 encryption

Supports RC4 encryption and decryption.

7.3.4 DES encryption

Support DES encryption and decryption, support ECB and CBC two modes.

7.3.5 3DES encryption

Supports 3DES encryption and decryption, and supports both ECB and CBC modes.

7.3.6 AES encryption

Support AES encryption and decryption, support ECB, CBC and CTR three modes.

7.3.7 CRC encryption

The hardware CRC calculation can be performed on consecutive multi-packet data, the calculation result is stored in the register, and the encryption result of the previous packet can be used as the initial value of the next packet.

It supports four modes: CRC8, CRC16_MODBUS, CRC16_CCITT and CRC32, and supports input/output inversion.

The calculation formula of CRC32 is as follows:

1. CRC-32: 0x04C11DB7

X32+X26+X23+X22+X16+X12+X11+X10+X8+X7+X5+X4+X2+X+1

Commonly used in ZIP, RAR, IEEE 802 LAN/FDDI, IEEE 1394, PPP-FCS and other protocols.

2. CRC-16: supports two polynomials

2.1: 0X1021

X16 + X12 + X5 + 1

Commonly used in ISO HDLC, ITU X.25, V.34/V.41/V.42, PPP-FCS CCITT and other protocols.

2.2: 0X8005

X16 + X15 + X2 + 1

Commonly used in USB, ANSI X3.28, SIA DC-07 and other protocols.

3. CRC-8: 0X207

x8+x2+x1+1

7.3.8 TRNG Random Number Generator

A true random number generator module is integrated into the W800 system. Divided into analog module and digital post-processing module. The analog module outputs a random clock ad_trng_clks and random numbers ad_trng_dout, the digital post-processing module is used to eliminate the bias and autocorrelation of random numbers. Related control

The control register is in the GPSEC register list.

The basic operation process is as follows:

1. Enable TRNG_EN and set TRNG_SEL to 1, so that GPSEC register 0x48 displays the output value of TRNG. mode at this time

The pseudo module starts to output random clocks and random signals. The signal sampled by the first 8 clocks is used as the initial state of the LFSR to initialize the LFSR chain, the data sampled by each random clock is post-processed by XOR CHAIN and LFSR register, and then shifted and stored in the register in the server TRNG_RANDOM.

2. Software can read random value through GPSEC register 0x48. Digital postprocessing when register TRNG_DIG_BYPASS is set to 1

The module stops working and directly stores the output value of the analog module into the result register TRNG_RANDOM.

7.4 Register Description

7.4.1 Register List

Table 31 Cryptographic Module Register List

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x0000	source address register	SRC_ADDR	RW	RC4/SHA1/AES/DES/3DES/CRC/MD5 Multiplexing source address	0x0000_0000
0X0004	destination address register	DEST_ADDR	RW	RC4/AES/DES/3DES multiplexing destination address	0x0000_0000
0X0008	configuration register	GPSEC_CFG	RW	Generic Hardware Cryptographic Module Configuration Registers	0x0000_0000
0X000C	control register	GPSEC_CTRL	RW	Generic Hardware Cryptographic Module Control Register	0x0000_0000
0X0010	Key 0 Low Register	KEY00	RW	The first input key in the lower 32 bits of Key0 (RC4/AES/DES/3DES), multiplexed CRC Ci	0x0000_0000
0X0014	Key 0 High Register	KEY01	RW	Key0 high 32-bit first input key (RC4/AES/DES/3DES)	0x0000_0000
0X0018	Key 1 Low Register	KEY10	RW	The second input key of the lower 32 bits of Key1 (RC4/AES//3DES)	0x0000_0000
0X001C	Key 1 High Register	KEY11	RW	Key1 high 32 bits second input key (RC4/AES//3DES)	0x0000_0000
0X0020	Key 2 Low Register	KEY20	RW	Key2 low 32 bit third input key (3DES), multiplex iv1 low 32-bit input initial Vector (AES)	0x0000_0000
0X0024	Key 2 High Register	KEY21	RW	Key2 high 32 bits third input key (3DES), multiplex iv1 high 32-bit input initial Vector (AES)	0x0000_0000
0X0028	Initial vector 0 low register device	IV00	RW	IV0 low 32-bit input initial vector (AES/DES/3DES)	0x0000_0000
0X002C	Initial vector 0 high order register device	IV01	RW	IV0 upper 32-bit input initial vector (AES/DES/3DES)	0x0000_0000
0X0030		GPSEC_STS	RW	Generic Hardware Cryptographic Module Status Register	0x0000_0000
0X0030	Summary 0 Register	SHA1-DIGEST0	RW	sha1-digest0/MD5-digest0	0X6745_2301
0X0038	Summary 1 Register	SHA1-DIGEST1	RW	sha1-digest1/MD5-diges1	0XEFCD_AB89
0X003C	Summary 2 Register	SHA1-DIGEST2	RW	sha1-digest2/MD5-digest	0X98BA_DCFE
0X0040	Summary 3 Register	SHA1-DIGEST3	RW	sha1-digest3/MD5-digest3	0X1032_5476

0X0044	Summary 4 Register	SHA1-DIGEST4	RW	sha1-digest4/MD5-digest4	0XC3D2_E1F0
0X0044	RNG_result	RNG_RESULT	RW	RNG output	0X0000_0000
0X0048	Key 3 Low Register	Key30	RW	The third input key in the lower 32 bits of Key3 (RC4's 256bit mode)	0X0000_0000
0X004C	Key 3 Low Register	Key31	RW	The third input key in the upper 32 bits of Key3 (RC4's 256bit mode)	0X0000_0000
0X0050	TRNG config-uration	TRNG_CR	RW	True random number generator configura-tion options	0X40

========== TO BE TRANSLATED ==========

9 GPIO module

9.1 Function overview

The GPIO controller implements software configuration of GPIO properties, enabling users to conveniently operate GPIO.

Each GPIO can be individually configured by software, set it as input port, output port, set its floating, pull-up, pull-down state, set its rising edge, falling edge, double edge, high level, low level interrupt trigger mode.

9.2 Main Features

- Support GPIO software configuration
- Support GPIO interrupt configuration
- Provides up to 48 GPIOs available

9.3 Functional Description

The GPIO provided in W800 is divided into two groups, one is GPIOA, the other is GPIOB. The starting addresses of GPIOA and GPIOB registers are different, but function the same.

When the user wants to use a specific IO as a software-controlled GPIO, set the corresponding position in the GPIO multiplexing selection register to 0, i.e. Can.

The GPIO direction control register is used to control the direction of the GPIO, 1 means the corresponding GPIO is used as an output pin, 0 means the corresponding GPIO as an input pin.

The GPIO pull-up and pull-down control registers are used to control the pull-up and pull-down functions of the corresponding IO.

The GPIO pull-up control register is **active low**, setting it to 0 means to open the pull-up function of the corresponding IO, and setting it to 1 means to close the pull-up and pull-down function.

For the properties of IO, please refer to the IO multiplexing table.

The GPIO pull-down control register is **active high**. Setting it to 1 means turning on the pull-down function of the corresponding IO, and setting it to 0 means turning off the pull-down function.

For the properties of IO, please refer to the IO multiplexing table.

The GPIO data register represents the level of the input IO when it is set to the input state, and can be specified by writing 1 or 0 when it is set to the output state IO output level. This register is controlled by the GPIO data enable register, only when the GPIO data enable register is set to 1 time, the GPIO data register can be read and written.

The GPIO module provides input signal detection function. High and low level detection and upper and lower edges can be realized by configuring GPIO interrupt related registers Jump detection. When the input signal corresponding to IO meets the preset conditions, such as high-level trigger or rising edge trigger, etc., it will trigger GPIO interrupt is reported to MCU for processing. The MCU needs to clear the corresponding interrupt status to avoid false triggering of the interrupt.

9.4 Register Description

9.4.1 Register List

Table 44. **GPIOA register list**

(aliases of registers as described in CDK file "wm_regs.h", not as in RM!!)

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x0000	GPIO data register	**DATA**	RW	Read and write GPIO current data	0x180B
0x0004	GPIO data enable register	**DATA_B_EN**	RW	Configure the enable bit of GPIO_DATA	0xFFFF
0x0008	GPIO direction control register	**DIR**	RW	Configure GPIO direction	0x0000
0x000C	GPIO Pull-Up Control Register	**PULLUP_EN**	RW	Configure GPIO pull-ups	0xFFFF
0x0010	GPIO multiplexing selection register	**AF_SEL**	RW	Configure GPIO alternate function enable bit	0xFFFF
0x0014	GPIO multiplexing selection register 1	**AF_S1**	RW	GPIO alternate function selection bit high address bit	0x0000
0x0018	GPIO multiplexing selection register 0	**AF_S0**	RW	GPIO alternate function selection bit low address bit	0x0000
0x001C	GPIO pull-down control register	**PULLDOWN_EN**	RW	Configure GPIO pull-down	0x0000
0x0020	GPIO interrupt trigger mode configuration register	**IS**	RW	Configure the interrupt triggering method of GPIO	0x0000
0x0024	GPIO interrupt edge-triggered mode configuration	**IBE**	RW	Configuring GPIO Interrupt Edge Triggered Mode	0x0000
0x0028	GPIO interrupt upper and lower edge trigger configuration register	**IEV**	RW	Configure GPIO interrupt upper and lower edge trigger or high and low level touch	0x0000
0x002C	GPIO Interrupt Enable Configuration Register	**IE**	RW	Configure GPIO Interrupt Enable	0x0000
0x0030	GPIO Bare Interrupt Status Register	**RIS**	RO	Query GPIO raw interrupt status (before MASK)	0x0000
0x0034	GPIO Masked Interrupt Status Register	**MIS**	RO	Query the interrupt status after GPIO masking (MASK Rear)	0x0000
0x0038	GPIO Interrupt Clear Control Register	**IC**	WO	Control GPIO interrupt clearing	0x0000

Table 45 **GPIOB register list**

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x0000	GPIO data register	**DATA**	RW	Read and write GPIO current data	0x0000_7304
0x0004	GPIO data enable register	**DATA_B_EN**	RW	Configure the enable bit of GPIO_DATA	0x7FFF_FFFF
0x0008	GPIO direction control register	**DIR**	RW	Configure GPIO direction	0x0000_0000

0x000C	GPIO Pull-Up Control Register	**PULLUP_EN**	RW	Configure GPIO pull-ups	0xFFFF_FFFF
0x0010	GPIO multiplexing selection register	**AF_SEL**	RW	Configure GPIO alternate function enable bit	0xFFFF_FFFF
0x0014	GPIO multiplexing selection register 1	**AF_S1**	RW	GPIO alternate function selection bit high address bit	0x0000_0000
0x0018	GPIO multiplexing selection register 0	**AF_S0**	RW	GPIO alternate function selection bit low address bit	0x0000_0000
0x001C	GPIO pull-down control register	**PULLDOWN_EN**	RW	Configure GPIO pull-down	0x0000_0000
0x0020	GPIO interrupt trigger mode configuration register	**IS**	RW	Configure the interrupt triggering method of GPIO	0x0000_0000
0x0024	GPIO interrupt edge-triggered mode configuration	**IBE**	RW	Configuring GPIO Interrupt Edge Triggered Mode	0x0000_0000
0x0028	GPIO interrupt upper and lower edge trigger configuration register	**IEV**	RW	Configure GPIO interrupt upper and lower edge trigger or high and low level touch	0x0000_0000
0x002C	GPIO Interrupt Enable Configuration Register	**IE**	RW	Configure GPIO Interrupt Enable	0x0000_0000
0x0030	GPIO Bare Interrupt Status Register	**RIS**	RO	Query GPIO raw interrupt status (before MASK)	0x0000_0000
0x0034	GPIO Masked Interrupt Status Register	**MIS**	RO	Query the interrupt status after GPIO masking (MASK Rear)	0x0000_0000
0x0038	GPIO Interrupt Clear Control Register	**IC**	WO	Control GPIO interrupt clearing	0x0000_0000

9.4.2 **DATA** - GPIO data register

Table 46. **GPIOA** data register

Bit	Access	Operating Instructions	reset value
[15:0]	RW	GPIO current data, each BIT corresponds to the corresponding GPIO line	0x180B

Table 47 **GPIOB** data register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	GPIO current data, each BIT corresponds to the corresponding GPIO line	0x7304

9.4.3 **DATA_B_EN** - GPIO Data Enable Register

Table 48 **GPIOA** Data Enable Register

Bit	Access	Operating Instructions	reset value
[15:0]	RW	Corresponding to the BIT enable bit of GPIO_DATA, only when the corresponding BIT is 1, the operation of the corresponding bit of GPIO_DATA	0xFFFF

		It is valid only after the operation, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO_DATA[x] cannot be read or written [x] = 1, GPIO_DATA[x] can be read and written	

Table 49 GPIOB Data Enable Register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	Corresponding to the BIT enable bit of GPIO_DATA, only when the corresponding BIT is 1, the operation of the corresponding bit of GPIO_DATA It is valid only after the operation, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO_DATA[x] cannot be read or written [x] = 1, GPIO_DATA[x] can be read and written	0xFFFFFFFF

9.4.4 **GPIO_DIR** - GPIO direction control register

Table 50 GPIOA Direction Control Register

Bit	Access	Operating Instructions	reset value
[15:0]	RW	GPIO direction control, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO[x] is input [x] = 1, GPIO[x] is output	0x0000

Table 51 GPIOB direction control register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	GPIO direction control, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO[x] is input [x] = 1, GPIO[x] is output	0x00000000

9.4.5 **PULLUP_EN** - GPIO pull-up and pull-down control register

Table 52 GPIOA Pull-Up Control Register ()

Bit	Access	Operating Instructions	reset value
[15:0]	RW	GPIO pull-up control, each BIT corresponds to the corresponding GPIO line, 1'bx: Note: **This register is active low!** [x] = 0, GPIO[x] has pull-up [x] = 1, no pull-up on GPIO[x]	0xFFFF

PULLDOWN_EN - GPIOA Pull-Down Control Register ()

Bit	Access	Operating Instructions	reset value
[15:0]	RW	GPIO pull-down control, each BIT corresponds to the corresponding GPIO line, 1'bx: Note: **This register is active high!** [x] = 1, GPIO[x] has pull-down [x] = 0, GPIO[x] has no pull-down	0x0000

Table 53 GPIOB pull-up and pull-down control registers

Bit	Access	Operating Instructions	reset value
[31:0]	RW	GPIO pull-up control, each BIT corresponds to the corresponding GPIO line, 1'bx: Note: **This register is active low!** [x] = 0, GPIO[x] has pull-up [x] = 1, no pull-up on GPIO[x]	0xFFFFFFFF

PULLDOWN_EN - GPIOA Pull-Down Control Register ()

Bit	Access	Operating Instructions	reset value
[31:0]	RW	GPIO pull-down control, each BIT corresponds to the corresponding GPIO line, 1'bx: Note: **This register is active high!** [x] = 1, GPIO[x] has pull-down [x] = 0, GPIO[x] has no pull-down	0x00000000

9.4.6 **AF_SEL** - GPIO multiplexing selection register

Table 54 GPIOA multiplexing selection register

Bit	Access	Operating Instructions	reset value
[15:0]	RW	GPIO multiplexing function enable bit, each BIT corresponds to whether the corresponding GPIO multiplexing function is enabled, 1'bx: [x] = 0, GPIO[x] alternate function is disabled [x] = 1, GPIO[x] alternate function is enabled When [x] = 1, the alternate function depends on the corresponding BITs of the two registers **AF_S1** and **AF_S0** status. S1.[x] = 0, S0.[x] = 0, alternate function 1 (opt1) S1.[x] = 0, S0.[x] = 1, alternate function 2 (opt2) S1.[x] = 1, S0.[x] = 0, alternate function 3 (opt3) S1.[x] = 1, S0.[x] = 1, alternate function 4 (opt4) When [x] = 0, if **DIR**[x] = 0 and **PULLUP_EN**[x] = 1, the GPIO multiplexing is opt6 analog IO function For the IO multiplexing function, please refer to the chip pin multiplexing relationship	0xFFFF

Table 55 GPIOB multiplexing selection register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	GPIO multiplexing function enable bit, each BIT corresponds to whether the corresponding GPIO multiplexing function is enabled, 1'bx: [x] = 0, GPIO[x] alternate function is disabled [x] = 1, GPIO[x] alternate function is enabled When [x] = 1, the alternate function depends on the corresponding BITs of the two registers **AF_S1** and **AF_S0** status. S1.[x] = 0, S0.[x] = 0, alternate function 1 (opt1) GPIO multiplexing function enable bit, each BIT corresponds to whether the corresponding GPIO multiplexing function is enabled, 1'bx: [x] = 0, GPIO[x] alternate function is disabled [x] = 1, GPIO[x] alternate function is enabled When [x] = 1, the alternate function depends on the corresponding BITs of the two registers **AF_S1** and **AF_S0** status. S1.[x] = 0, S0.[x] = 0, alternate function 1 (opt1)	0xFFFFFFFF

9.4.7 **AF_S1 -** GPIO multiplexing selection register 1

Table 56 GPIOA multiplexing selection register 1

Bit	Access	Operating Instructions	reset value
[15:0]	RW	The high address bit of the GPIO alternate function selection bit, together with AF_S0 to determine the alternate function	0x0000

		For the IO multiplexing function, please refer to the chip pin multiplexing relationship	

Table 57 GPIOB multiplexing selection register 1

Bit	Access	Operating Instructions	reset value
[31:0]	RW	The high address bit of the GPIO alternate function selection bit, together with GPIO_AF_S0 to determine the alternate function For the IO multiplexing function, please refer to the chip pin multiplexing relationship	0x00000000

9.4.8 **AF_S0 -** GPIO multiplexing selection register 0

Table 58 GPIOA multiplexing selection register 0

Bit	Access	Operating Instructions	reset value
[15:0]	RW	The low address bit of the GPIO alternate function selection bit, and GPIO_AF_S1 together determine the alternate function How to configure see GPIO_AF_SEL register description	0x0000

Table 59 GPIOB multiplexing selection register 0

Bit	Access	Operating Instructions	reset value
[31:0]	RW	The low address bit of the GPIO alternate function selection bit, and GPIO_AF_S1 together determine the alternate function How to configure see GPIO_AF_SEL register description	0x00000000

9.4.9 **IS** - GPIO interrupt trigger mode configuration register

Table 60 GPIOA interrupt trigger mode configuration register

Bit	Access	Operating Instructions	reset value
[15:0]	RW	GPIO interrupt triggering method, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO[x] interrupt is edge-triggered [x] = 1, GPIO[x] interrupt is level sensitive	0x0000

Table 61 GPIOB interrupt trigger mode configuration register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	GPIO interrupt triggering method, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO[x] interrupt is edge-triggered [x] = 1, GPIO[x] interrupt is level sensitive	0x00000000

9.4.10 **IBE** - GPIO Interrupt Edge Triggered Mode Configuration Register

Table 62 GPIOA Interrupt Edge Triggered Mode Configuration Register

Bit	Access	Operating Instructions	reset value
[15:0]	RW	GPIO interrupt edge trigger mode, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO[x] edge-triggered interrupt mode is determined by **IEV** [x] = 1, both edges of GPIO[x] trigger an interrupt	0x0000

Table 63 GPIOB Interrupt Edge Triggered Mode Configuration Register

Bit	Access	Operating Instructions	reset value

[31:0]	RW	GPIO interrupt edge trigger mode, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO[x] edge-triggered interrupt mode is determined by **IEV** [x] = 1, both edges of GPIO[x] trigger an interrupt	0x00000000

9.4.11 **IEV** - GPIO interrupt upper and lower edge trigger configuration register

Table 64 GPIOA interrupt upper and lower edge trigger configuration register

Bit	Access	Operating Instructions	reset value
[15:0]	RW	GPIO interrupt upper and lower edge trigger or high and low level trigger selection, each BIT corresponds to the corresponding GPIO line, 1'BX: [x] = 0, GPIO [X] interrupt is triggered at a low or falling edge [x] = 1, GPIO [x] interrupt is high or rising edge trigger	0x0000

Table 65 GPIOB interrupt upper and lower edge trigger configuration register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	GPIO interrupt upper and lower edge trigger or high and low level trigger selection, each BIT corresponds to the corresponding GPIO line, 1'BX: [x] = 0, GPIO [X] interrupt is triggered at a low or falling edge [x] = 1, GPIO [x] interrupt is high or rising edge trigger	0x00000000

9.4.12 **IE** - GPIO Interrupt Enable Configuration Register

Table 66 GPIOA Interrupt Enable Configuration Register

Bit	Access	Operating Instructions	reset value
[15:0]	RW	GPIO interrupt enable control, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO[x] interrupt disabled [x] = 1, GPIO[x] interrupt enable	0x0000

Table 67 GPIOB Interrupt Enable Configuration Register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	GPIO interrupt enable control, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO[x] interrupt disabled [x] = 1, GPIO[x] interrupt enable	0x00000000

9.4.13 **RIS** - GPIO Raw Interrupt Status Register

Table 68 GPIOA Raw Interrupt Status Register

Bit	Access	Operating Instructions	reset value
[15:0]	RO	GPIO bare interrupt status (before MASK), each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, no interrupt is generated for GPIO[x] [x] = 1, GPIO[x] has an interrupt	0x0000

Table 69 GPIOB Raw Interrupt Status Register

Bit	Access	Operating Instructions	reset value
[31:0]	RO	GPIO bare interrupt status (before MASK), each BIT corresponds to the corresponding GPIO line, 1'bx:	0x00000000

		[x] = 0, no interrupt is generated for GPIO[x] [x] = 1, GPIO[x] has an interrupt	

9.4.14 **MIS** - GPIO Masked Interrupt Status Register

Table 70 GPIOA Masked Interrupt Status Register

Bit	Access	Operating Instructions	reset value
[15:0]	RO	Interrupt status after GPIO masking (after MASK), each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, no interrupt is generated for GPIO[x] (after MASK) [x] = 1, GPIO[x] interrupt is generated (after MASK)	0x0000

Table 71 GPIOB Masked Interrupt Status Register

Bit	Access	Operating Instructions	reset value
[31:0]	RO	Interrupt status after GPIO masking (after MASK), each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, no interrupt is generated for GPIO[x] (after MASK) [x] = 1, GPIO[x] interrupt is generated (after MASK)	0x00000000

9.4.15 **IC** - GPIO Interrupt Clear Control Register

Table 72 GPIOA Interrupt Clear Control Register

Bit	Access	Operating Instructions	reset value
[15:0]	WO	GPIO interrupt clear control, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, no action [x] = 1, clear GPIO[x] interrupt status	0x0000

Table 73 GPIOB Interrupt Clear Control Register

Bit	Access	Operating Instructions	reset value
[31:0]	WO	GPIO interrupt clear control, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, no action [x] = 1, clear GPIO[x] interrupt status	0x00000000

========== TO BE TRANSLATED ==========

9 Timer module

19.1 Functional overview

The timer contains a 32-bit auto-loaded counter driven by a divided system clock. W800 has 6 fully independent timer. Accurate timing and interrupt functions are implemented, which can be used for delayed or periodic event processing.

19.2 Main Features

- 6 fully independent timers
- 32-bit autoload counter
- The timing unit can be configured as ms, us
- It can realize single timing or repeated timing function
- Scheduled interrupt function
- You can query the timer count value at any time;

19.3 Functional Description

The timer module consists of 6 completely independent timers, which do not affect each other, and the 6 channels can work at the same time.

After the system clock is divided by the frequency division factor, the us standard clock is obtained, which is used for the input clock of the counter. Timing unit can be configured as us, ms two kinds of levels.

The timing value is a 32-bit configurable register that can meet the needs of different timing durations. Each timer corresponds to an interrupt, when after the timing time is met, if the interrupt function is enabled, an interrupt request will be generated, which can be used to process periodic events.

19.3.1 Timing function

Timing function means that according to the time set by the user, when the time is up, a hardware interrupt is generated to notify the user to implement a specific function. Timing trigger support ticket there are two types, time and period, one can be used to process single events, and the other can be used to process periodic events.

The user obtains the APB bus clock frequency according to the frequency division factor of the system clock, and sets the base microsecond count configuration register of the timer (TMR_CONFIG), set the timing value, configure the timing unit, work mode, enable the interrupt, and then start the timing function. when timed when the time is up, the program enters the timer interrupt processing function and clears the interrupt.

19.3.2 Delay function

The delay function means that the user can keep the program in a waiting state according to the countdown function of the timer, and the program will continue to run until the timing is completed.

19.4 Register Description

19.4.1 Register List

Table 162 Timer register list

(aliases of registers as described in CDK file “wm_regs.h”, not as in RM!!)

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x0000	Standard us configuration registers	**TMR_CONFIG**	RW	Standard us timing divider value, by bus time divide the frequency to get the standard us timing, this value Equal to APB bus frequency (MHz) minus 1	0X0000_0027
0x0004	Timer Control Register	**CR**	RW	Timer Control Register	0X0631_8C63
0x0008	Timer 1 Timing Value Configuration Register	**TIM0_PRD**	RW	Timer1 timing value configuration register	0X0000_0000
0x000C	Timer2 Timing Value Configuration Register	**TIM1_PRD**	RW	Timer2 timing value configuration register	0X0000_0000
0x0010	Timer 3 Timing Value Configuration Register	**TIM2_PRD**	RW	Timer3 timing value configuration register	0X0000_0000
0x0014	Timer 4 Timing Value Configuration Register	**TIM3_PRD**	RW	Timer4 timing value configuration register	0X0000_0000
0x0018	Timer 5 Timing Value Configuration Register	**TIM4_PRD**	RW	Timer5 timing value configuration register	0X0000_0000

0x001C	Timer 6 Timing Value Configu-ration Register	**TIM5_PRD**	RW	Timer6 timing value configuration register	0X0000_0000
0x0020	Timer 1 current count value	**TMR1_CNT**	RO	Timer1 current count value	0X0000_0000
0x0024	Timer 2 current count value	**TMR2_CNT**	RO	Timer2 current count value	0X0000_0000
0x0028	Timer 3 current count value	**TMR3_CNT**	RO	Timer3 current count value	0X0000_0000
0x002C	Timer 4 current count value	**TMR4_CNT**	RO	Timer4 current count value	0X0000_0000
0x0030	Timer 5 current count value	**TMR5_CNT**	RO	Timer5 current count value	0X0000_0000
0x0034	Timer 6 current count value	**TMR6_CNT**	RO	Timer6 current count value	0X0000_0000

19.4.2 **TMR_CONFIG** - Standard us configuration registers.

Bit	Ac-cess	Operating Instructions	reset value
[31:7]		reserved	
[6:0]	RW	The clock divider configures prescale. For example: apb_clk=40MHz prescale = 40 – 1 = 8'd39	0x27 (dec 39)

19.4.3 **TMR_CR** - Timer Control Register.

Bit	Ac-cess	Operating Instructions	reset value
[31:30]		reserved	0x00
[29:25]	RW	TMR6_CSR, same as TMR1_CSR	0x03
[24:20]	RW	TMR5_CSR, same as TMR1_CSR	0x03
[19:15]	RW	TMR4_CSR, same as TMR1_CSR	0x03
[14:10]	RW	TMR3_CSR, same as TMR1_CSR	0x03
[9:5]	RW	TMR2_CSR, same as TMR1_CSR	0x03
[4:0]	RW	[4:0] is TMR1_CSR, as follows:	
		[4]: Interrupt status flag, **TIM_CR_TIM0_TIF**, write 1 to clear 1'b0: Timer generates no interrupt; 1'b1: Timer generates an interrupt;	0
		[3]: Interrupt Enable bit, **TIM_CR_TIM0_TIE** 1'b0: No interrupt will be generated after the timing is completed; 1'b1: An interrupt is generated after the timing is completed;	0
		[2]: Timer enable bit, **TIM_CR_TIM0_EN** 1'b0: the timer does not work; 1'b1: enable timer	0
		[1]: Timer working mode, **TIM_CR_TIM0_MODE** 1'b0: Timer repeats timing; 1'b1: The timer only timed once, and it will be closed automatically after the time is com-pleted;	1
		[0]: Timer timing unit, **TIM_CR_TIM0_UNIT** 1'b0: the timing unit is us; 1'b1: The timing unit is ms;	1

19.4.4 **TIM0_PRD** - Timer 0 Timing Value Configuration Register

Bit	Ac-cess	Operating Instructions	reset value
[31:0]	RW	Configure the timer value of Timer 0	0x00000000

19.4.5 **TIM1_PRD** - Timer 1 Timing Value Configuration Register

Bit	Ac-cess	Operating Instructions	reset value
[31:0]	RW	Configure the timer value of Timer 1	0x00000000

19.4.6 **TIM2_PRD -** Timer 2 Timing Value Configuration Register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	Configure the timer value of Timer 2	0x00000000

19.4.7 **TIM3_PRD** - Timer 3 Timing Value Configuration Register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	Configure the timer value of Timer 3	0x00000000

19.4.8 **TIM4_PRD** Timer 4 Timing Value Configuration Register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	Configure the timer value of Timer 4	0x00000000

19.4.9 **TIM5_PRD** Timer 5 Timing Value Configuration Register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	Configure the timer value of Timer 5	0x00000000

TIM0_CNT - Timer 0 current count value register

Bit	Access	Operating Instructions	reset value
[31:0]	RO	Current timer value of Timer 0	0x00000000

TIM1_CNT Timer 1 current count value register

Bit	Access	Operating Instructions	reset value
[31:0]	RO	Current timer value of Timer 1	0x00000000

TIM2_CNT - Timer 2 current count value register

Bit	Access	Operating Instructions	reset value
[31:0]	RO	Current timer value of Timer 2	0x00000000

TIM3_CNT Timer 4 current count value register

Bit	Access	Operating Instructions	reset value
[31:0]	RO	Current timer value of Timer 3	0x00000000

TIM4_CNT Timer 5 current count value register

Bit	Access	Operating Instructions	reset value
[31:0]	RO	Current timer value of Timer 4	0x00000000

TIM5_CNT Timer 5 current count value register

Bit	Access	Operating Instructions	reset value
[31:0]	RO	Current timer value of Timer 5	0x00000000

10 High Speed SPI Device Controller

10.1 Function overview

Compatible with the general SPI physical layer protocol, by agreeing on the data format for interaction with the host, the host can access the device at high speed, up to working frequency is 50MHZ.

10.2 Main Features

- Compatible with general SPI protocol
- Selectable level interrupt signal
- Supports up to 50Mbps
- Simple frame format, full hardware parsing and DMA

10.3 Functional Description

10.3.1 Introduction to SPI Protocol

SPI works in a master-slave manner, usually with one master and one or more slaves, requiring at least 4 wires, in fact 3 wires are also possible (single when transferring). They are SDI (data input), SDO (data output), SCLK (clock), CS (chip select).

(1) SDI – Serial Data In, serial data input

(2) SDO – Serial Data Out, serial data output

(3) SCLK – Serial Clock, clock signal, generated by the master device

(4) CS – Chip Select, the slave device enable signal, controlled by the master device.

Among them, CS is the control signal of whether the slave chip is selected by the master chip, that is to say, only when the chip select signal is the predetermined enable signal (high potential or low potential), the operation of the master chip is valid for this slave chip. This makes it possible to connect multiple SPI devices on the same bus.

In addition to the above 4 signal lines, HSPI also adds an INT line, which generates a drop when the slave device has data to upload.

The interruption of the edge realizes the active reporting of data.

SPI communication is accomplished through data exchange, the data is transmitted bit by bit, the clock pulse is provided by SCLK, SDI, SDO are based on

This pulse completes the data transfer. The data output goes through the SDO line, the data changes on the rising or falling edge of the clock, and the data changes on the following falling edge or rising edge is read. To complete one-bit data transmission, the same principle is used for input. Therefore, at least 8 changes of the clock signal (the rising edge and the lower edge is once) to complete the transmission of 8-bit data.

The SCLK signal line is controlled by the master device, and the slave device cannot control the signal line. In an SPI-based device, there is at least one master device.

10.3.2 SPI working process

The HSPI inside the chip works with the wrapper controller. The wrapper controller integrates DMA and is implemented through DMA.

Data exchange between HSPI internal FIFO and chip internal cache. This operation is implemented in hardware, and software does not need to care about data transmission and reception

In the process, you only need to configure the sending and receiving data linked list, and operate the corresponding registers of the wrapper controller.

For a detailed introduction to the wrapper controller, please refer to the relevant chapters.

10.4 Register Description

10.4.1 Register List of Internal Operation of HSPI Chip

Table 74 HSPI Internal Access Registers

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x0000	HSPI FIFO clear register	**CLEAR_FIFO**	RW	Clear the contents of the Tx and Rx FIFOs while A circuit that synchronously resets the system clock domain	0X0000_0000
0x0004	HSPI Configuration Register	**SPI_CFG**		Configure the SPI transfer mode and endianness set	0x0000_0000
0x0008	HSPI Mode Configuration Register	**MODE_CFG**		When configuring the ahb master to access the bus burst length	0x0000_0000
0X000C	HSPI Interrupt Configuration Register	**SPI_INT_CPU_MASK**		Configure whether the interrupt is enabled	0X0000_0003

0X0010	HSPI Interrupt Status Register	**SPI_INT_CPU_STTS**		Get and clear interrupt status	0x0000_0000
	HSPI data upload length register	**RX_DAT_LEN**		Configure the length of data that can be uploaded	0x0000_0000

10.4.1.1 **CLEAR_FIFO** - HSPI FIFO clear register

Table 75 HSPI FIFO clear register

Bit	Access	Operating Instructions	reset value
[31:1]	RO	reserved	
[6:0]	RW	Clear FIFOs, clear the contents of the Tx and Rx FIFOs, and synchronously reset the circuit of the system clock domain (this except for the registers in the list) 0: Do not clear the FIFO 1: Clear valid Set by software, cleared by hardware Note: If you want to reset the whole circuit, you need to use the asynchronous reset leg of this module: rst_n	0

10.4.1.2 **SPI_CFG** - HSPI Configuration Register

Table 76 HSPI configuration registers

Bit	Access	Operating Instructions	reset value
[31:4]	RO	reserved	
[3]	RW	Bigendian, spi interface supports big and small endian selection of data. 0: support small segment data transfer 1: Support big-endian data transmission	0
[2]	RW	spi_tx_always_drive 0: The spi output is only valid when the chip select is valid, and it is high impedance at other times 1: spi output is always valid	0
[1]	RW	SPI CPHA 0: Transmission mode A 1: Transmission mode B	0
[0]	RW	SPI CPOL, SCK polarity at IDLE 0: 0 for SCK IDLE 1: 1 when SCK IDLE	0

10.4.1.3 **MODE_CFG** - HSPI Mode Configuration Register

Table 77 HSPI Mode Configuration Register

Bit	Access	Operating Instructions	reset value
[31:1]	RO	reserved	
[0]	RW	Burst len, the burst length when the ahb master accesses the bus 0: burst len is 1 word 1: burst len is 4 characters It is recommended to set the burst transmission of 4 words, so that when the frequency of the spi interface is high, the continuous flow can be guaranteed.	0

10.4.1.4 **SPI_INT_CPU_MASK** - HSPI Interrupt Configuration Register

Table 78 HSPI Interrupt Configuration Register

Bit	Access	Operating Instructions	reset value
[31:2]	RO	reserved	

[1]	RW	IntEnRxOverrun, RxOverrun interrupt enable 0: Rx FIFO overflow interrupt enable 1: Rx FIFO overflow interrupt disabled	1
[0]	RW	IntEnTxUnderrun, TxUnderrun interrupt enable 0: Tx FIFO underflow interrupt enable 1: Tx FIFO underflow interrupt disabled	1

10.4.1.5 **SPI_INT_CPU_STTS -** HSPI Interrupt Status Register

Table 79 HSPI Interrupt Status Register

Bit	Access	Operating Instructions	reset value
[31:2]	RO	reserved	
[1]	RW	RxOverrun 0：Rx FIFO overflow 1: Rx FIFO overflow Write 1 to clear	0
[0]	RW	TxUnderrun 0: Tx FIFO underflow 1: Tx FIFO underflow Write 1 to clear	0

10.4.1.6 **RX_DAT_LEN-** HSPI data upload length register

Table 80 HSPI data upload length register

Bit	Access	Operating Instructions	reset value
[31:16]	RO	reserved	
[15:0]	RW	Rx_dat_len Indicates the length of data that can be uploaded, in bytes The upload length is an integer multiple of words. If the upload length is less than a whole word, it will be rounded up.	0x0000

10.4.2 Host side access HSPI controller register list

The host side accesses the SPI interface registers through a fixed SPI command format. The command length is fixed to one byte, and the data length is fixed to two byte.

Table 81 HSPI Interface Configuration Register (Master Access)

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x02	Get data length register	**RX_DAT_LEN**	RO	When uploading data, the spi host is used to obtain data from the length of the data read by the device side	0x0000
0x03	Send data flag register	**TX_BUFF_AVAIL**	RO	When the master sends data to the slave, it is used to judge whether it can be download data or commands	0x0000
0x04	reserved		RO		
0x05	Interrupt configuration register	**SPI_INT_HOST_MASK**	RW	Whether to mask the interrupt	0x0000
0x06	Interrupt Status Register		RO	Interrupt status register, the spi host polls this bit check if there is data to upload	0x0000
0x07	reserved		RO		
0x00	Data Port 0	**DAT_PORT0**	RW	The Spi master communicates to the slave device through this register port To send data, the previous data frame is sent using this port	

0x10	Data Port 1	**DAT_PORT1**	RW	The Spi master communicates to the slave device through this register port To send data, send the last data frame to use the port	
0x01	Command port 0	**DN_CMD_PORT0**	WO	The Spi host sends the slave device through this register Command data, use the previous command data the port	
0x11	Command port 1	**DN_CMD_PORT1**	WO	The Spi host sends the slave device through this register Command data, send the last frame of command data to make use this port	

10.4.2.1 HSPI get data length register

Table 82 HSPI Get Data Length Register

Bit	Access	Operating Instructions	reset value
[15:0]	RO	spi host read-only register, when uploading data, it is mainly used to know how much data is read from the device side But in this module, the upload length is an integer multiple of words. If the upload length value is not an integer word, the host will read round up when counting, that is, read some redundant bytes	0x0000

========== TO BE TRANSLATED ==========

20 Power Management Module

20.1 Function Overview

The PMU implements the switching of the chip hardware operating state and the power management during the state switching, and provides timers, real-time clocks, and 32K clocks.

It also provides timer, real time clock and 32K clock.

20.2 Main Features

- Provides chip power control
- Provides timer functionality
- Provides real-time clock control
- Provides 32K RC oscillator calibration
- Provide wake-up function.

20.3 Function Description

20.3.1 Full Chip Power Control

The PMU module controls the power switch of the chip, including 40M start-up circuit, Bandgap, digital PLL, voltage detection circuit, digital circuit, and LDO.

When the chip is powered on, the PMU module guides each module to turn on the power in turn according to the preset power-on sequence.

When the software configuration registers enter sleep mode, the PMU module guides each functional module to turn off the power in turn according to the safe power-down sequence.

turning off the clock, crystal start-up circuitry and associated power supplies in sequence when the software configuration registers enter sleep mode.

Three wake-up modes are provided in the sleep/sleep mode: Timer timer, RTC timer or wake-up by pulling the special WAKEUP pin high.

20.3.2 Low-power modes

2 low-power modes can be selected by configuring the PMU register chip.

Standby mode.

In this mode, the power supply of the digital power domain will be turned off, and only the PMU module will work on the whole chip, providing wake-up and reset functions; at this time, the power consumption of the whole chip is about 15uA.

The power consumption of the chip is about 15uA. After power off, all the data and contents stored in the memory will be lost, and the firmware will be reloaded after waking up, which is equivalent to reboot.

After waking up, the firmware will be reloaded, which is equivalent to reboot.

Sleep mode.

In this mode, the power supply of the digital power domain will be retained, only the DPLL and crystal start circuit will be turned off, and the clock will be cut off.

The power consumption of the whole chip is about 1mA; the stored data and code in the memory will still be retained; the program will continue to run after waking up.

20.3.3 Wake-up modes

PMU supports 3 wake-up modes, Timer wake-up, RTC wake-up and external IO wake-up.

Timer wakeup

Before setting the hibernation/sleep mode in the software, configure the Timer0 module in the PMU and set the hibernation time. When the system enters the hibernate mode, when the Timer0 timer reaches the sleep time, it will wake up the system and generate the corresponding Timer interrupt. When the system resumes operation, it needs to write the corresponding status bits in the interrupt status register.

Otherwise, the system will be woken up by the interrupt immediately after entering the hibernation mode next time.

RTC Wakeup

Before setting the hibernate/sleep mode in software, configure the RTC module in PMU and set the hibernate time. When the system enters the hibernation mode, when the system enters the sleep mode, the system will be woken up when the RTC time reaches the sleep time and the corresponding RTC interrupt will be given. After the system resumes operation, please write '1' to the corresponding status bit in the interrupt register 0x14 to clear the interrupt status, otherwise, the system will be woken up by the interrupt immediately after entering the hibernation mode next time.

External IO Wakeup

After software hibernation/sleep, the PMU detects a specific Wakeup pin and the external controller can wake up the system by pulling this IO high and giving the corresponding IO wakeup interrupt.

The PMU does not detect this IO state after leaving the hibernation mode. After the system resumes operation, please write '1' to the corresponding status bit in the interrupt register 0x14, otherwise, the system will be woken up by the interrupt immediately after the next sleep mode.

20.3.4 Timer0 Timer

The timer enable signal and timing time are configured through the AHB register. First set the timer value, then set the timer enable BIT to start the timer.

When the timing time is reached, an interrupt is generated and the software clears the interrupt flag by writing the BIT0 of the status register.

20.3.5 Real Time Clock Function

Refer to the real time clock module

20.3.6 32K clock source switching and calibration

The W800 chip integrates a 32K RC oscillator as the PMU module clock source.

The output frequency of the 32K RC oscillator may change due to changes in operating environment and temperature, resulting in timing deviations. Therefore, the 32K RC oscillator is introduced in the PMU module.

Therefore, a 32K RC oscillator calibration function and a 32K clock switching function are introduced in the PMU module to correct timing deviations.

1) Switching of 32K clock source

The 32K clock can be switched from 32K RC oscillator to 32K clock obtained from 40M clock division by setting bit3 of PS_CR register to 1.

The 32K clock is obtained from the 40M clock divider. However, when the chip enters sleep mode, bit3 will be automatically cleared to 0 because the 40M clock will be turned off.

If the firmware still wants to use the precision timing function, you need to reset bit3 to 1.

2) Calibration of 32K RC oscillation circuit

First set bit2 of PS_CR register to 0, then set bit2 of PS_CR register to 1.

After the calibration is completed, the 32K RC oscillator will be relatively accurate. However, if you want more accurate timing, it is still recommended to use the 40M clock divider

This will result in an accurate RTC clock, and the deviation will only be related to the crystal frequency bias.

20.4 Register Description

20.4.1 Register List

Table 171 List of PMU registers

Offset Address	Name	Abbreviation	Ac-cess	Description	Reset Value
0x0000	PMU control register	**CR**	RW	For configuring 32K calibration, configuring 32K clock source, set the STANDBY function of the chip	0X0000_0002
0x0004	PMU Timer 0	**TIMER0**	RW	Configure the timing value (in seconds), enable timing device	0X0000_0000
0x0008	reserved				
0x0014	PMU Interrupt Source Register	**IF**	RW	Provides PMU interrupt flags	0X0000_0000

20.4.2 **PMU_CR** - PMU control register

Table 172 PMU control registers

Bit	Ac-cess	Operating Instructions	reset value
[31:11]	RO	reserved	
[10]	RW	Wake-up key interrupt polarity selection. 0: Set the interrupt when the wakeup button is high. 1: Set the interrupt when the wakeup button is high. Valid only for the Sleep interrupt.	1
[9:6]	RW	Minimum hold time for key wake-up. Unit: 128ms.	dec 01
[5]	RW	DLDO_CORE Reference Voltage Source Selection 1: ABG 0: DBG	1
[4]	RW	32K oscillation circuit BYPASS signal High valid, 32K is obtained by 40M clock division after setting to 1. ○2	0
[3]	RW	RC 32K oscillator calibration circuit start switch. 1'b0: reset of the calibration circuit. 1'b1: start calibration circuit. To start the calibration function, this bit needs to be set to 0 first and then to 1.	0
2	RW	Enable key trigger to sleep function 0: no enable. 1: When io_wakeup key is pressed in active mode and reaches the threshold time, it will trigger io_sleep_flag is interrupted and reported to MCU.	0
	RW	Sleep enable signal, high valid. 1'b0: Chip wake-up state 1'b1 1'b1: the chip enters the Sleep state If the WAKEUP pin is invalid and no TIMER0/1 interrupt wake-up is configured, the register is valid, the chip enters the Sleep state. If the wake-up interrupt is generated, the chip will switch from the Sleep state to the wake-up state, and when the wake-up condition is satisfied, this bit is automatically cleared to 0.	1

		If the wake-up interrupt is generated, the chip will switch from the sleep state to the wake-up state. Wake-up source: WAKEUP pin, TIMER0/TIMER1, RTC 1) WAKEUP pin, high active; for the chip to enter Sleep state, WAKEUP pin must be low. To wake up, pull the WAKEUP pin high to generate a wake-up interrupt and make the chip leave the Sleep state. 2) TIMER0, timer wake-up interrupt. When the WAKEUP pin is low, TIMER0 sets the timing time and enables it. Timer0 will generate a wakeup interrupt to make the chip leave the Sleep state. 3) RTC, wake-up at timing time When WAKEUP pin is low, RTC will generate a wakeup interrupt when the timing time is up, which will make the chip leave the Sleep state. state.	
	RW	STANDBY enable signal, high valid. 1'b0: Chip wake-up state 1'b1: chip enters STANDBY state If the WAKEUP pin is invalid and no TIMER0/1 interrupt wake-up is configured, the register is valid, the chip enters the STANDBY state. If the wake-up interrupt is generated, the chip will switch from STANDBY state to wake-up state, and when the wake-up condition is satisfied, this bit will be cleared to 0 automatically. If the wake-up interrupt is generated, the chip will switch from STANDBY state to wake-up state, and this bit will be cleared automatically when the wake-up condition is satisfied.	0

20.4.3 **PMU_TIMER0 -** PMU Timer 0

Table 173 PMU Timer 0 Register

Bit	Access	Operating Instructions	reset value
[31:17]	RO	reserved	0
[16]	RW	Timer0 enable bit 1’b0: Bit enable. 1'b1: enable;	0
[15:0]	RW	Timing value of Timer0, unit: second	0

20.4.4 **PMU_IF** - PMU Interrupt Source flag register

Table 174 PMU Interrupt Source Register

Bit	Access	Operating Instructions	reset value
[31:9]	RO	reserved	0
[8]	RW	Display the current power-on status: 1’b0: Power-on or reset start 1’b1: wake up from sleep, write 1 to clear	0
[7]	RO	reserved	0
[6]	RO	reserved	0
[5]	RW	RTC timer interrupt flag bit: 1'b0: A timed interrupt is generated 1’b1: No timing interrupt is generated, write 1 to clear	0
[4]	RW	reserved	0
[3]	RW	reserved	0
[2]	RW	WAKEUP pin wake-up interrupt flag 1’b0: No WAKEUP wake-up interrupt is generated 1’b1: WAEKUP wake-up interrupt is generated, write 1 to clear	0
[1]	RW		
[0]	RW	Timer0 timer interrupt flag bit: 1’b0: No Timer0 interrupt is generated 1’b1: Timer0 interrupt is generated, write 1 to clear	0

21 Real time clock module

21.1 Function overview

RTC is a BCD counter/timer provided by the PMU module. Two 32-bit registers contain seconds, minutes, hours, days, months, years, and two

The decimal format representation (BCD) of the base code can automatically correct the months of 28, 29 (leap year), 30, and 31 days.

Under the corresponding software configuration, the RTC can not only provide the clock calendar function, but also can be used as a timer, when the timer reaches the set time an RTC interrupt is then generated, which can be used to wake the system from sleep.

The RTC module has two clock sources that can be configured: 40M clock division and internal 32K clock. Which can be used by software configuration during normal work clock source; only 32K clocks can be used in sleep state. If the RTC clock source is divided by 40M clock in normal working state, then it will automatically switch to the 32K clock after entering the sleep state, and the 32K clock will still be used after the system is woken up. So as long as the power supply voltage within the working range, the RTC module will not stop working regardless of whether the module is in a normal working state or a sleep state.

21.2 Main Features

- Provide timing function
- Provide timing function
- Provide timed interrupt
- Interrupt to wake up the system

21.3 Functional Description

21.3.1 Timing function

The initial value of day, hour, minute and second can be configured in RTC configuration register 1, and the initial value of year and month can be configured in RTC configuration register 2.

The timekeeping function is enabled in RTC configuration register 2.

After the RTC timing function is enabled, read the RTC configuration register 1 to get the current day, hour, minute and second values.

Register 2 can get the current year and month value.

21.3.2 Timing function

The day, hour, minute, and second timing values can be configured in RTC configuration register 1, and the year and month timing values can be configured in RTC configuration register 2.

RTC Configuration Register 1 enables the timing function.

When the RTC timer reaches the timing time, an RTC interrupt will be generated. At this time, set the RTC interrupt bit of the PMU interrupt source register to 1 to clear it.

except the interrupted state.

When the system enters sleep mode, the interrupt generated by the RTC timer will wake up the system.

21.4 Register Description

21.4.1 Register List

The RTC module has a total of 2 32-bit dedicated registers. The RTC interrupt status needs to query the PMU interrupt source register.

Table 175 RTC register list

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x000C	RTC Configuration Register 1	**RTCCR0**	RW	Configure RTC day, hour, minute and second value, configure enable timing	0X0000_0000
0x0010	RTC Configuration Register 2	**RTCCR1**	RW	Configure RTC year and month value, configure enable timing	0X0000_0000

21.4.2 RTC Configuration Register 1

Table 176 RTC Configuration Register 1

Bit	Access	Operating Instructions	reset value
[31]	RW	RTC timer interrupt function enable 1'b0: Disable 1'b1: enable	0
[30:29]		reserved	
[28:24]	RW	Day start value/day time value	0
[23:21]		reserved	

[20:16]	RW	Hour initial value/hour timing value	0
[15:14]		reserved	
[13:8]		reserved	
[7:6]		reserved	
[5:0]	RW	Second initial value/second timing value	0

21.4.3 RTC Configuration Register 2

Table 177 RTC Configuration Register 2

Bit	Access	Operating Instructions	reset value
[31:17]		reserved	0
[16]	RW	RTC timing function enable bit 1'b0: Disable 1'b1: enable	0
[15]			
[14:8]	RW	Beginning of the year value/yearly fixed value	0
[7:4]			
[3:0]	RW	Monthly value/monthly fixed value	0

22 Watchdog module

22.1 Function overview

Implement the "watchdog" function. Designed for a global reset in the event of a system crash.

The "watchdog" will generate a periodic interrupt. The system software will clear its interrupt flag after the interrupt is generated. If it exceeds the set time, it will not be cleared.

Otherwise, a hard reset signal will be generated to reset the system.

22.2 Main Features

- Provide timing function
- Provide reset function
- Provide timed interrupt

22.3 Functional Description

22.3.1 Timing function

After setting the timing value to the register WD_LD, set BIT0 of WDG_CTRL to 1 to start the timer, and the WDG module will generate the

When a timed interrupt occurs, the notification program is processed. If the BIT0 of the register WD_CLR is not cleared, a timed interrupt will be generated periodically.

The value of WD_LD is based on the APB clock unit, and the APB clock is divided from the 160M clock.

22.3.2 Reset function

After setting the chip timing value WD_LD, start the timing and reset function (set BIT1/BIT0 of WDG_CTRL), and the WDG module starts the reverse operation timing, when the timing time is up, WDG will generate a timing interrupt. At the same time, if the BIT0 of WD_CLR is not cleared, the chip will the reset signal is generated in the next cycle.

22.4 Register Description

22.4.1 Register List

Table 178 WDG Register List

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x0000	WDG Timing Load Register	**WD_LD**	RW	Configure timing values for repeated loading	0XFFFF_FFFF
0x0004	WDG current value register	**WD_VAL**	RO	Get the value of the current timer	0xFFFF_FFFF
0x0008	WDG Control Register	**WD_CTRL**	RW	control register	0X0000_0000
0x000C	WDG Interrupt Clear Register	**WD_CLR**	WO	interrupt clear register	0X0000_0000
0x0010	WDG Interrupt Source Register	**WD_SRC**	RO	interrupt source register	0X0000_0000
0x0014	WDG Interrupt Output Register	**WD_STATE**	RO	Interrupt Output Status Register	0X0000_0000

22.4.2 **WD_LD** WDG Timing Value Load Register

Table 179 WDG Timing Value Load Register

Bit	Access	Operating Instructions	reset value
[31:0]	RW	Configure timing values for repeated loading The value of this register is counted in APB clocks. For example: when the APB clock is 40MHZ, the maximum duration of the timing value is about 107s, that is, 0FFFFFFFF/40000000	0xFFFF_FFFF

22.4.3 **WD_VAL** - WDG Current Value Register

Table 180 WDG Current Value Register

Bit	Access	Operating Instructions	reset value
[31:0]	RO	Get the value of the current timer To calculate the remaining time, just read the current value. To calculate the elapsed time, simply subtract the value of register WD_VAL from the value of register WD_LD	0xFFFF_FFFF

22.4.4 **WD_CTRL** - WDG Control Register

Table 181 WDG Control Register

Bit	Access	Operating Instructions	reset value
[31:2]		reserved	0
[1]	RW	reset enable bit 1'b0: When the WDG reset condition occurs, no reset signal is generated 1'b1: When the WDG reset condition occurs, the reset signal is generated	0
[0]	RW	timing enable bit 1'b0: Timer not working 1'b1: The timer works and generates periodic interrupts	0

22.4.5 **WD_CLR -** WDG Interrupt Clear Register

Table 182 WDG Interrupt Clear Register

Bit	Access	Operating Instructions	reset value
[31:1]		reserved	0
[0]	WO	Interrupt status clear bit, write any value to clear the current interrupt status	0

22.4.6 **WD_SRC** - WDG Interrupt Source Register

Table 183 WDG Interrupt Source Register

Bit	Access	Operating Instructions	reset value
[31:1]		reserved	0
[0]	RO	The interrupt source register, the timer function is turned on, will generate the interrupt at the same time	0

22.4.7 **WD_STATE** - WDG Interrupt Status Register

Table 184 WDG Interrupt Status Register

Bit	Access	Operating Instructions	reset value
[31:1]		reserved	0
[0]	RO	Interrupt output status register. This interrupt is not generated after the timer is turned off, but WD_SRC may be 1	0

23 PWM controller

23.1 Function overview

PWM is a method of digitally encoding analog signal levels. Through the use of a high resolution counter, the duty cycle of the square wave is modulated with to encode the level of a specific analog signal. The PWM signal is still digital because at any given moment, the full-scale the current supply is either completely present (ON) or completely absent (OFF). A voltage or current source is a repetitive pulse of ON or OFF.

The pulse sequence is added to the simulated load. When it is on, the DC power supply is applied to the load, and when it is off, the power supply is disconnected.

when. Any analog value can be encoded using PWM as long as the bandwidth is sufficient.

23.2 Main Features

- Support 2-channel input signal capture function (two channels of PWM0 and PWM4)
- Input signal capture function supports interrupt interactive mode and DMA transfer mode; DMA mode supports word-by-word operation
- Support 5-channel PWM signal generation function
- 5-channel PWM signal generation supports one-shot generation mode and auto-load mode
- Support 5-channel braking function
- PWM output frequency range: 3Hz~160kHz
- The maximum precision of the duty cycle: 1/256, the width of the counter inserted in the dead zone: 8bit
- Support channel 0 channel 1 synchronization function, support channel 2 channel 3 synchronization function
- Supports complementary and non-complementary modes of channel 0, channel 1, and complementary and non-complementary modes of channel 2 and channel 3
- Support 5-channel sync function

23.3 Functional Description

23.3.1 Input Signal Capture

The PWM controller supports the signal capture function of two channels, and the capture of channel 0 can be activated by setting Bit24 of the PWM_CTL register.

The capture function of channel 4 can be activated by setting Bit1 of the PWM_CAP2CTL register. The level of the captured signal can also be to set whether to flip the function. After the channel captures the corresponding signal, the capture number is updated to the corresponding capture register PWM_CAPDAT (pass channel 0 capture number) and PWM_CAP2DAT (channel 4 capture number).

23.3.2 DMA Transfer Captures

After channel 0 or channel 4 enables the capture function, the count of the capture register can be quickly transferred to the memory through the DMA channel to speed up the user process.

23.3.3 Support for single-shot and autoload modes

The five output channels of the PWM controller all support one-shot output mode and auto-load mode. In single-load mode, the channel outputs the specified cycle after the waveform, the PWM wave will no longer be output; in the automatic loading mode, after the channel outputs the specified cycle waveform, the cycle will be automatically reloaded number, so as to continue to generate PWM waves.

23.3.4 Multiple Output Modes

The PWM controller supports independent output mode, that is, each channel outputs independently and does not interfere with each other; supports dual-channel synchronous mode, that is, one channel

The output is exactly the same as the output of another channel; supports five-channel sync mode, the output of channel 1 to channel 4 is exactly the same as the output of channel 0

It supports dual-channel complementary output, that is, the waveform output by one channel is completely opposite to the waveform output by the other channel; supports complementary mode

Commonly used dead zone settings, the dead zone length can be set up to 256 clock cycles; support braking mode, when the braking port detects the specified power

After leveling, the output channel will output the set braking level.

A variety of output modes are flexible and configurable to meet users' various application scenarios related to PWM.

23.4 Register Description

23.4.1 PWM Register List

Table 185 PWM Register List (**register abbr. as in *wm_regs.h***)

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x0000	Clock divider register_01	**PWM_CLKDIV01**	RW	Divide the clock for channel 0 and channel 1	0x00000000
0x0004	Clock divider register_02	**PWM_CLKDIV23**	RW	Divide the clock for channel 2 and channel 3	0x00000000
0x0008	control register	**PWM_CTL**	RW	Used to configure or control some configurable items	0x00000000
0x000C	Period register	**PWM_PERIOD**	RW	Used to set the period of channel 0 to channel 4	0x00000000
0x0010	Period Number Register	**PWM_PNUM**	RW	is used to set the signal generation of channel 0 to channel 4 into cycles	0x00000000
0x0014	Compare Register	**PWM_CMPDAT**	RW	is used to store the comparison value of channel 0 to channel 4 to produce different duty cycles	0x00000000
0x0018	Dead Time Control Register	**PWM_DTCTL**	RW	is used to configure or control the configurable set item	0x00000000
0x001C	Interrupt Control Register	**PWM_IE**	RW	Used to enable and control related interrupts	0x00000000
0x0020	Interrupt Status Register	**PWM_IF**	RW	Used to query the status of related interrupts	0x00000000
0x0024	Channel 0 Capture Register	**PWM_CH0CAPDAT**	RO	Used to capture and count rises to channel 0 edge and falling edge	0x00000000
0x0028	brake control register	**PWM_BKCR**	RW	Used to control the braking mode	0x00000000
0x002C	Clock divider register_4	**PWM_CH4CR1**	RW	Divide the clock for channel 4	0x00000000
0x0030	Channel 4 Control Register_2	**PWM_CH4CR2**	RW	Set related configuration items of channel 4	0x00000000
0x0034	Channel 4 Capture Register	**PWM_CH4CAPDAT**	RO	Used to capture and count the rise to channel 4 edge and falling edge	0x00000000
0x00338	Channel 4 Control Register_3	**PWM_CH4CR3**	RW	Set related configuration items of channel 4	0x00000000

23.4.2 **PWM_CLKDIV01** - Clock divider register_01

Table 186 PWM clock divider register_01

Bit	Access	Operating Instructions	reset value
[31:16]	RW	CLKDIV1 CH1 frequency division counter The frequency division is determined by the counter value Note: The frequency division range is (0~65535). If frequency division is not required, enter 0 or 1.	0x0000
[15:0]	RW	CLKDIV0 CH0 frequency division counter Same as CH1	0x0000

23.4.3 2 **PWM_CLKDIV012** - Clock divider register_23

Table 187 PWM Clock Divider Register_23

Bit	Access	Operating Instructions	reset value
[31:16]	RW	CLKDIV3 CH3 frequency division counter Same as CH1	0x0000
[15:0]	RW	CLKDIV2 CH2 frequency division counter Same as CH1	0x0000

23.4.4 **PWM_CTL** Control Register

Table 188 PWM Control Register

Bit	Access	Operating Instructions	reset value
[31:27]	RW	CNTEN Counter count enable 1'b0: stop counting 1'b1: start counting Note: Each bit controls each channel separately, and controls CH4, CH3, CH2, CH1 and CH0 in turn from high to low	0

[26]		reserved	0
[25]	RW	CAPINV Capture reverse enable flag 1’b0: Inverse of input signal in capture mode is invalid 1'b1: The input signal in the capture mode is valid in reverse, and the input signal is reversed	0
[24]	RW	CPEN Capture function enable flag 1’b0: The capture function of CH0 is invalid, and the RCAPDAT and FCAPDAT values will not be updated; 1’b1: CH0 capture function is valid, capture and latch the PWM counter, respectively store in RCAPDAT (rising edge latch) and FCAPDAT (falling edge latch)	
[23:22]	RW	CNTTYPE3 2'b0 CH3 Counter counting method 2’b00: edge-aligned mode (counter counting is incremented, only for capture mode) 2’b01: edge-aligned mode (counter counts down, only for PWM mode) 2’b10: Center-aligned mode (PWM mode only) Note: In PWM mode, when the counter is set to edge-aligned mode, you need to set the counting method to decrement way.	00
[21:20]	RW	CNTTYPE2 CH2 Counter counting method Same as CH3	00
[19:18]	RW	CNTTYPE1 CH1 Counter counting method Same as CH3	00
[17:16]	RW	CNTTYPE0 CH0 Counter counting method Same as CH3	00
[15:14]	RW	TWOSYNCEN 2-channel sync mode enable signal 1’b0: 2-channel sync is not allowed 1’b1: Enable 2-channel synchronization, PWM_CH0 and PWM_CH1 have the same phase, and the phase is determined by PWM_CH0; PWM_CH2 has the same phase as PWM_CH3, and the phase is determined by PWM_CH2 15bit control CH3 and CH2 14bit control CH1 and CH0	00
[13]		reserved	0
[12]	RW	POEN PWM pin output enable bit 1’b0: PWM pin is set to output state 1’b1: PWM pin is set to tri-state Note: only for CH0	0
[11:8]	RW	CNTMODE PWM generation loop method 1'b0: One shot mode 1'b1: Autoload mode Note: During the change of CNTMODE, PWM_CMPDAT returns to zero; each bit controls each channel separately, from high controls PW3, PW2, PW1 and PW0 in sequence to low	0
[7]		reserved	0
[6]	RW	ALLSYNCEN All-channel sync mode enable signal 1'b0: All channels are not allowed to synchronize 1’b1: All channels are allowed to synchronize, PWM_CH0, PWM_CH1, PWM_CH2 and PWM_CH3 have the same phase, and the phase is determined by PWM_CH0	0
[5:2]	RW	PINV PWM output signal polarity enable 1’b0: PWM output polarity reversal disabled 1’b1: PWM output polarity reversal enable Note: Each bit controls each channel separately, control PWM3, PWM2, PWM1 and PWM0 in sequence from high to low	0
[1:0]	RW	OUTMODE output mode 1'b0: Non-complementary mode for every two channels 1'b1: Complementary mode for every two channels BIT1 controls CH2 and CH3 BIT0 controls CH0 and CH1	0

23.4.5 **PWM_PERIOD** - Period Register

Table 189 PWM Period Register

Bit	Access	Operating Instructions	reset value
[31:24]	RW	PERIOD3 CH3 period register value (Note: period cannot be greater than 255) "Edge-aligned mode (counter counts down)": ➢ PERIOD register value, period value is (PERIOD + 1)	0x00

		➢ Duty cycle = (CMP+1)/(PERIOD + 1) ➢ CMP>=PERIOD: PWM output is fixed high ➢ CMP<PERIOD: PWM low level width is (PERIOD-CMP), high level width is (CMP+1) ➢ CMP=0: PWM low level width is PERIOD, high level width is 1; "Center Alignment Mode": ➢ PERIOD register value: period is 2*(PERIOD+1) ➢ Duty cycle=(2*CMP+1)/(2*(PERIOD+1)) ➢ CMP>PERIOD: PWM is continuously high ➢ CMP<=PERIOD: PWM low level=2*(PERIOD-CMP)+1, High level=(2*CMP)+1 ➢ CMP=0: PWM low level width is 2*PERIOD+1, high level width is 1. Note: The number of cycles should not be 255 in "Center Aligned Mode". No matter which alignment mode is selected, the channel period is determined by the division number (N) and the number of periods (P), That is: the input clock is 40MHz, the clock frequency f_div after frequency division is: f_div = 40MHz/N, N is the division Frequency (16bit). The output frequency f_output is: f_output = f_div / P, where P is the number of cycles. Note: In PWM mode, when the counter is set to edge-aligned mode, you need to set the counting method to decrement way.	
[23:16]	RW	PERIOD2 CH2 period register value (Note: period cannot be greater than 255) Same as PERIOD3	0x00
[15:8]	RW	PERIOD1 CH1 period register value (Note: period cannot be greater than 255) Same as PERIOD3	0x00
[7:0]	RW	PERIOD0 CH0 period register value (Note: period cannot be greater than 255) Same as PERIOD3	0x00

23.4.6 **PWM_PNUM** - Cycle Number Register

Table 190 PWM Period Number Register

Bit	Access	Operating Instructions	reset value
[31:24]	RW	PNUM3 PWM3 generation cycle number Set the number of PWM3 cycles PNUM3, when the PWM generates PNUM3 PWM signals, stop generating the signal. number, trigger the interrupt and set the interrupt status word at the same time	0x00
[23:16]	RW	PNUM2 Number of PWM2 generation cycles Same as PNUM3	0x00
[15:8]	RW	PNUM1 Number of PWM1 generation cycles Same as PNUM3	0x00
[7:0]	RW	PNUM0 Number of PWM0 generation cycles Same as PNUM3	0x00

23.4.7 **PWM_CMPDAT** - Compare Register

Table 191 PWM Compare Register

Bit	Access	Operating Instructions	reset value
[31:24]	RW	"Edge-aligned mode (counter counts down)": ➢ PERIOD register value, period value is (PERIOD + 1) ➢ Duty cycle = (CMP+1)/(PERIOD + 1) ➢ CMP>=PERIOD: PWM output fixed bit high ➢ CMP<PERIOD: PWM low level width is (PERIOD-CMP), high level width is (CMP+1) ➢ CMP=0: PWM low level width is PERIOD, high level width is 1; "Center Alignment Mode": ➢ PERIOD register value: period is 2*(PERIOD+1) ➢ Duty ratio=(2*CMP+1)/2*(PERIOD+1) ➢ CMP>PERIOD: PWM is continuously high ➢ CMP<=PERIOD: PWM low level=2*(PERIOD-CMP)+1, high level=(2*CMP)+1 ➢ CMP=0: PWM low level width is 2*PERIOD+1, high level width is 1.	0x00

		Regardless of which alignment mode is selected, the channel period is determined by the division number (N) and the number of periods (P), namely: The input clock is 40MHz, the clock frequency f_div after frequency division is: f_div = 40MHz/N, N is the frequency division number (16bit). The output frequency f_output is: f_output = f_div / P, where P is the number of cycles. Note: In PWM mode, when the counter is set to edge-aligned mode, you need to set the counting method to decrement way.	
[23:16]	RW	CMP2 PWM2 compare register value Same as CMP3	0x00
[15:8]	RW	CMP2 PWM2 compare register value Same as CMP3	0x00
[7:0]	RW	CMP2 PWM2 compare register value Same as CMP3	0x00

23.4.8 **PWM_DTCTL** - Dead Time Control Register

Table 192 PWM Dead Time Control Register

Bit	Ac-cess	Operating Instructions	reset value
[31:22]		reserved	0x00
[21]	RW	DTEN23 Can channel 2 and channel 3 insert deadband valid flags? Insert deadband valid signal is valid only after the complementary mode of the channel is turned on. And, if a valid signal is inserted If it is 0, the complementary signal output by the two channels has no dead zone insertion. 1'b0: Invalid insertion dead zone 1'b1: Insert dead zone valid	0
[20]	RW	DTEN01 Can channel 0 and channel 1 insert deadband valid flags? Same as DTEN23	0
[19:18]		reserved	0
[17:16]	RW	DTDIV Dead time clock divider control 2'b00: Dead time clock equal to base clock (40MHz) 2'b01: The dead-time clock is equal to the reference clock (40MHz) divided by two 2'b10: Dead-band clock equal to base clock (40MHz) divided by four 2'b11: Dead-band clock equal to base clock (40MHz) divided by eight	0
[15:8]	RW	DTCNT23 Dead time interval for channel 3 and channel 2 8bit determines the dead time interval value, and the dead time clock is determined by DTDIV	0
[7:0]	RW	DTCNT01 Dead time interval for channel 1 and channel 0 8bit determines the dead time interval value, and the dead time clock is determined by DTDIV	0

23.4.9 **PWM_IE** - Interrupt Control Register

Table 193 PWM Interrupt Control Register

Bit	Ac-cess	Operating Instructions	reset value
[31:8]		reserved	0
[21]	RW	DMA_request_EN DMA_request enable 1'b0: DMA_request is invalid 1'b1: DMA_req	0
[6]	RW	FLIEN Falling edge buffer interrupt enable bit 1'b0: Falling edge buffer interrupt invalid 1'b1: Falling edge buffer interrupt is valid Note: For CH0	0
[5]	RW	RLIEN Rising edge buffer interrupt enable bit 1'b0: Rising buffer interrupt invalid 1'b1: Rising edge buffer interrupt is valid Note: For CH0	0

[4:0]	RW	PIEN PWM Period Interrupt Enable Bit 1'b0: Periodic interrupt is invalid 1'b1: Periodic interrupt is valid Note: When the counter counts to 0 and the number of PWM cycles meets PWM_PNUM, an interrupt is triggered.	0

23.4.10 **PWM_IF** - Interrupt Status Register

Table 194 PWM Interrupt Status Register

Bit	Access	Operating Instructions	reset value
[31:10]		reserved	0
[9]	RW	OVERFL Counter overflow flag 1'b0: Capture mode, the counter does not overflow during the counting process 1'b1: Capture mode, counter overflows during counter counting Note: When the user clears CFLIF or CRLIF, this bit is also cleared at the same time	0
[8]	RW	FLIFOV Falling edge delay interrupt flags overrun status 1'b0: When CFILF is 1, no falling edge delay interrupt is generated 1'b1: When CFILF is 1, another falling edge delay interrupt occurs Note: When the user clears CFILF, this bit is also cleared at the same time	0
[7]	RW	RLIFOV Rising edge delay interrupt flag overrun status 1'b0: When CRILF is 1, no rising edge delay interrupt is generated 1'b1: When CRILF is 1, another rising edge delay interrupt occurs Note: When the user clears CRILF, this bit is also cleared at the same time	0
[6]	RW	CFLIF Capture falling edge interrupt flag 1'b0: No falling edge captured 1'b1: When a falling edge is captured, this bit is set to 1 Note: By writing 1, clear this flag; Note: For CH0	0
[5]	RW	CRLIF Capture rising edge interrupt flag 1'b0: No rising edge captured 1'b1: When a rising edge is captured, this bit is set to 1 Note: By writing 1, clear this flag; Note: For CH0	0
[4:0]	RW	PIF PWM Period Interrupt Flag When the PWM generates a PWM signal with a specified period, the flag is set to 1; write 1 through software to clear the flag Note: Each bit identifies each channel, and controls PW4, PW3, PW2, PW1 and PW0 in sequence from high to low	0

23.4.11 **PWM_CH0CAPDAT** - Channel 0 Capture Register

Table 195 PWM Channel 0 Capture Register

Bit	Access	Operating Instructions	reset value
[31:16]	RO	PWM_FCAPDAT Capture falling edge register When there is a falling edge of the input signal, the current counter value is stored	0x00
[15:0]	RO	PWM_RCAPDAT Capture rising edge register When there is a rising edge of the input signal, the current counter value is stored	0x00

23.4.12 **PWM_BKCR** - Brake Control Register

Table 196 PWM Brake Control Register

Bit	Access	Operating Instructions	reset value
[31:16]		reserved	0x00
[15:11]	RW	BRKCTL	0x00

		Brake Mode Enable 1’b0: Braking mode disabled 1’b1: Brake mode activated [15:11] correspond to CH4, CH3, CH2, CH1 and CH0 respectively	
[10:8]		reserved	0
[7:3]	RW	BKOD Brake output control register 1’b0: When the braking mode is valid, the PWM output is low level 1'b1: When the braking mode is valid, the PWM output is high level [7:3] correspond to CH4, CH3, CH2, CH1 and CH0 respectively	0
[2:0]		reserved	0

23.4.13 **PWM_CH4CR1** - Channel 4 Control Register_1

Table 197 PWM Channel 4 Control Register_1

Bit	Access	Operating Instructions	reset value
[31:16]	RW	CLKDIV4 CH4 Frequency division counter The frequency division is determined by the counter value Note: The frequency division range is (0~65535). If frequency division is not required, enter 0 or 1.	0x0000
[15:8]	RW	PERIOD4 CH4 period register value (Note: period cannot be greater than 255) "Edge-aligned mode (counter counts down)": ➢ PERIOD register value, period value is (PERIOD + 1) ➢ Duty cycle = (CMP+1)/(PERIOD + 1) ➢ CMP>=PERIOD: PWM output fixed bit high ➢ CMP<PERIOD: PWM low level width is (PERIOD-CMP), high level width is (CMP+1) ➢ CMP=0: PWM low level width is PERIOD, high level width is 1; "Center Alignment Mode": ➢ PERIOD register value: period is 2*(PERIOD+1) ➢ Duty cycle=(2*CMP+1)/(2*(PERIOD+1)) ➢ CMP>PERIOD: PWM is continuously high ➢ CMP<=PERIOD: PWM low level=2*(PERIOD-CMP)+1, high level=(2*CMP)+1 ➢ CMP=0: PWM low level width is 2*PERIOD+1, high level width is 1. Note: The number of cycles should not be 255 in "Center Aligned Mode". No matter which alignment mode is selected, the channel period is determined by the division number (N) and the number of periods (P), That is: the input clock is 40MHz, the clock frequency f_div after frequency division is: f_div = 40MHz/N, N is the division Frequency (16bit). The output frequency f_output is: f_output = f_div / P, where P is the number of cycles. Note: In PWM mode, when the counter is set to edge-aligned mode, you need to set the counting method to decrement way.	0x00
[7:0]	RW	CH4 Generation Cycles Set the number of PWM4 cycles to PNUM4, after the PWM generates PNUM4 PWM signals, stop generating a signal while triggering an interrupt and setting the interrupt status word	0x00

23.4.14 **PWM_CH4CR2** - Channel 4 Control Register_2

Table 198 PWM Channel 4 Control Register_2

Bit	Access	Operating Instructions	reset value
[31:16]		reserved	0x0000
[15:8]	RW	CMP4 CH4 period register value "Edge-aligned mode (counter counts down)": ➢ PERIOD register value, period value is (PERIOD + 1) ➢ Duty cycle = (CMP+1)/(PERIOD + 1) ➢ CMP>=PERIOD: PWM output fixed bit high ➢ CMP<PERIOD: PWM low level width is (PERIOD-CMP), high level width is (CMP+1) ➢ CMP=0: PWM low level width is PERIOD, high level width is 1; "Center Alignment Mode": ➢ PERIOD register value: period is 2*(PERIOD+1) ➢ Duty ratio=(2*CMP+1)/2*(PERIOD+1)	0

		➢ CMP>PERIOD: PWM is continuously high ➢ CMP<=PERIOD: PWM low level=2*(PERIOD-CMP)+1, high level=(2*CMP)+1 ➢ CMP=0: PWM low level width is 2*PERIOD+1, high level width is 1. Regardless of which alignment mode is selected, the channel period is determined by the division number (N) and the number of periods (P), namely: The input clock is 40MHz, the clock frequency f_div after frequency division is: f_div = 40MHz/N, N is the frequency division number (16bit). The output frequency f_output is: f_output = f_div / P, where P is the number of cycles. Note: In PWM mode, when the counter is set to edge-aligned mode, you need to set the counting method to decrement way.	
[7:5]		reserved	
[4:3]		CNTTYPE4 CH4 Counter counting method 2'b00: edge-aligned mode (counter counting is incremented, only for capture mode) 2'b01: edge-aligned mode (counter counts down, only for PWM mode) 2'b10: Center-aligned mode (PWM mode only) Note: In PWM mode, when the counter is set to edge-aligned mode, you need to set the counting method to decrement way.	0
[2]		reserved	
[1]		CNTMODE4 CH4 generation cycle method 1'b0: One shot mode 1'b1: Autoload mode Note: During CNTMODE changing, PWM_CMPDAT returns to zero	0
[0]		PINV4 CH4 output signal polarity enable 1'b0: PWM output polarity reversal disabled 1'b1: PWM output polarity reversal enable	0

23.4.15 **PWM_CH4CAPDAT** - Channel 4 Capture Register

Table 199 PWM Channel 4 Capture Register

Bit	Ac-cess	Operating Instructions	reset value
[31:16]	RO	PWM_FCAP2DAT Capture falling edge register When there is a falling edge of the input signal, the current counter value is stored	0x0000
[15:0]	RO	PWM_RCAP2DAT Capture rising edge register When there is a rising edge of the input signal, the current counter value is stored	0x0000

23.4.16 **PWM_CH4CR3** - Channel 4 Control Register 3

Table 200 PWM Channel 4 Control Register_3

Bit	Ac-cess	Operating Instructions	reset value
[31:11]		reserved	0x0000
[10]	RW	DMA_request2_mask DMA_request2 enable 1'b0: DMA_request2 is invalid 1'b1: DMA_request2 is valid Note: only for CH4	0
[9]	RW	FLIEN2 Falling edge buffer interrupt enable bit 1'b0: Falling edge buffer interrupt invalid 1'b1: Falling edge buffer interrupt is valid Note: only for CH4	0
[8]	RW	RLIEN2 Rising edge buffer interrupt enable bit 1'b0: Rising buffer interrupt invalid 1'b1: Rising edge buffer interrupt is valid Note: only for CH4	0
[7]	RW	OVERFL2 Counter overflow flag 1'b0: Capture mode, the counter does not overflow during the counting process 1'b1: Capture mode, counter overflows during counter counting	0

		Note: When the user clears CFLIF or CRLIF, this bit is also cleared at the same time Note: only for CH4	
[6]	RW	FLIFOV2 Falling edge delay interrupt flags overrun status 1'b0: When CFILF is 1, no falling edge delay interrupt is generated 1'b1: When CFILF is 1, another falling edge delay interrupt occurs Note: When the user clears CFILF, this bit is also cleared at the same time Note: only for CH4	0
[5]	RW	RLIFOV2 Rising edge delay interrupt flag overrun status 1'b0: When CRILF is 1, no rising edge delay interrupt is generated 1'b1: When CRILF is 1, another rising edge delay interrupt occurs Note: When the user clears CRILF, this bit is also cleared at the same time Note: only for CH4	0
[4]	RW	CFLIF2 Capture falling edge interrupt flag 1'b0: No falling edge captured 1'b1: When a falling edge is captured, this bit is set to 1 Note: Clear this flag by writing a 1 Note: only for CH4	0
[3]	RW	CRLIF2 Capture rising edge interrupt flag 1'b0: No rising edge captured 1'b1: When a rising edge is captured, this bit is set to 1 Note: Clear this flag by writing a 1 Note: only for CH4	
[2]	RW	POEN2 PWM pin output enable bit 1'b0: PWM pin is set to output state 1'b1: PWM pin is set to tri-state Note: only for CH4	
[1]	RW	CPEN2 Capture function enable flag 1'b0: CH4 capture function is invalid, RCAPDAT and FCAPDAT values will not be updated; 1'b1: CH4 capture function is valid, capture and latch the PWM counter, respectively store in RCAPDAT (rising edge latch) and FCAPDAT (falling edge latch) Note: only for CH4	
[0]	RW	CAPINV2 Capture reverse enable flag 1'b0: Inverse of input signal in capture mode is invalid 1'b1: The input signal in the capture mode is valid in reverse, and the input signal is reversed Note: only for CH4	

14 SPI Controller

14.1 Function overview

SPI is an abbreviation for Serial Peripheral Interface. SPI is a high-speed, full-duplex, synchronous communication protocol

String. The communication principle of SPI is very simple. It works in a master-slave mode. This mode usually has a master device and one or more slave devices.

At least 4 wires, in fact 3 wires are also possible (for unidirectional transmission), including SDI (data input), SDO (data output), SCLK (time clock), CS (chip select).

14.2 Main Features

- Can be used as both SPI master and SPI slave
- Transmit and receive paths each have 8-word deep FIFOs
- master supports 4 formats of motorola spi (CPOL, CPHA), TI timing, macrowire timing
- slave supports 4 formats of motorola spi (CPOL, CPHA)
- Supports full duplex and half duplex
- The master device supports bit transmission, up to 65535bit transmission

- The slave device supports transmission modes of various length bytes
- The maximum clock frequency of spi_clk input from the device is 1/6 of the system APB clock

14.3 Functional Description

14.3.1 Master and slave can be configured

The SPI controller supports both a device as a SPI communication master and a device as a SPI communication slave. By setting the SPI_CFG register Bit2 of the device can switch the master-slave role of the device back and forth.

14.3.2 Multiple Mode Support

As the master device, by setting Bit1(CPHA) and Bit0(CPOL) of the SPI_CFG register, it can be set to MOTOROLA respectively.

The four formats of SPI transmit data. CPOL is used to determine the idle level of the SCK clock signal, CPOL=0, the idle level is low,

When CPOL=1, the idle level is high. CPHA is used to determine the sampling time, CPHA=0, on the first clock edge of each cycle

Sampling, CPHA=1, is sampled on the second clock edge of each cycle. The master can also be set by setting the TRANS_MODE register

The data is transmitted in the TI timing or the microwire timing, and the transmission data length under both timings can be adjusted.

As a slave device, only four formats of MOTOROLA SPI are supported, and the format selection is also done by setting the same signal as that of the master device register to achieve.

14.3.3 Efficient data transfer

The FIFO memory is a first-in, first-out dual-port buffer, that is, the first data entered into it is the first to be shifted out, and one of the memory's the input port, and the other port is the output port of the memory. The SPI controller integrates two (one for each transceiver) FIFO storage with a depth of 8 words. It can increase the data transfer rate, handle a large number of data streams, and match systems with different transfer rates, thereby improving system performance. able to pass Setting Bit[8:6] and Bit[4:2] of the MODE_CFG register can set the trigger level of RXFIFO and TXFIFO to meet different performance requirements at the same transmission rate. After the trigger level of the FIFO is triggered, an interrupt or DMA can be triggered to transfer data from memory move to TXFIFO or move data from RXFIFO to memory.

14.4 Register Description

14.4.1 Register List

Table 109 SPI register list

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x0000	Channel Configuration Register	**CH_CFG**	RW	is used to perform some configure	0x00000000
0x0004	SPI Configuration Register	**SPI_CFG**	RW	Configure SPI communication related items	0x00000004
0x0008	Clock Configuration Register	**CLK_CFG**	RW	Used to set the clock frequency division factor	0x00000000
0x000C	Mode Configuration Register	**MODE_CFG**	RW	Configure transfer mode	0x00000000
0x0010	Interrupt Control Register	**INT_MASK**	RW	Mask or enable related interrupts	0x000000FF
0x0014	Interrupt Status Register	**INT_SRC**	RW	Used to query interrupt sources	0x00000000
0x0018	SPI Status Register	**STATUS**	RO	List relevant states in SPI communication	0x00000000
0x001C	SPI Timeout Register	**TIMEOUT**	RW	Set SPI communication timeout	0x00000000
0x0020	data transmission register	**TXDATA**	RW	TX FIFO, used to store the data to be sent	0x00000000
0x0024	transfer mode register	**TRANS_MODE**	RW	Set transfer mode	0x00000000
0x0028	data length register	**SLV_LEN**	RO	Used as a slave device for storage when sending out or the length of the data received by the	0x00000000
0x002C		**RSV**		Reserved	0x00000000
0x0030	data receive register	**RXDATA**	RW/RO	RX FIFO for storing received data	0x00000000

14.4.2 **CH_CFG** - Channel Configuration Register

Table 110 SPI Channel Configuration Register

Bit	Access	Operating Instructions	reset value
[31]		reserved	0
[30:23]	RW	RX_INVALID_BIT, Indicates how many first bits are invalid data when the receiving channel starts to receive, and these invalid data need to be thrown directly off, do not enter the Rx FIFO. Only subsequent data goes into the Rx FIFO This register is used in conjunction with Tx/Rx length. The final amount of data actually stored in the Rx FIFO is Tx/Rx length - RX_INVALID_BIT Note: master mode is valid Motorola/TI mode active	0
[22]	RW	Clear FIFOs, clear the contents of Tx and Rx FIFO, and reset all circuits of master and slave synchronously (except configuration registers) 1'b0: Do not clear FIFO 1'b1: Clear valid Set to 1 by software, cleared to 0 by hardware Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[21]	RW	Clear FIFOs, clear the contents of Tx and Rx FIFO, and reset all circuits of master and slave synchronously (except configuration registers) 1'b0: Do not clear FIFO 1'b1: Clear valid Set to 1 by software, cleared to 0 by hardware Note: Both master/slave are valid Motorola/TI/microwire mode is valid When this mode is enabled, if there is no data in the tx fifo, invalid data may be sent first. so please after filling in the data, start the spi master Motorola/TI/microwire mode is valid	0
[20]	RW	RxChOn, whether the receive channel is turned on 1'b0: Rx channel off 1'b1: Rx channel on Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[19]	RW	TxChOn, whether the transmission channel is turned on 1'b0: Tx channel off 1'b1: Tx channel on Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[18:3]	RW	Tx/Rx length When Spi is transmitting, the number of valid SCKs It also indirectly reflects the length of the data sent or received. When (TxChOn=1, RxChOn=1), it indicates the number of bits sent and the maximum number of bits received (with how much the body receives is related to RX_INVALID_BIT) When (TxChOn=1, RxChOn=0), Indicates the number of bits sent, When (TxChOn=0, RxChOn=1), Indicates the maximum number of bits received (the specific number of received bits is related to RX_INVALID_BIT, the actual number of received bits is Tx/Rx length - RX_INVALID_BIT) When (TxChOn=0, RxChOn=0), meaningless. Note: master is valid Motorola/TI/microwire mode is valid	0
[2]	RW	Chip selects 1'b0: SPI_CS valid signal is 0 1'b1: SPI_CS valid signal is 1 Note: master is valid Motorola/TI/microwire mode is valid	0
[1]	RW	Force CS out 1'b0: spi_cs signal output is controlled by hardware 1'b1: The spi_cs signal output is controlled by software, and the specific output value is Chip selects This signal cooperates with Chip selects to realize the programmable output csn signal, that is, when the signal is 1, spi_cs = Chip selects Note: master is valid Motorola/TI/microwire mode is valid	0
[0]	RW	SPI start, Command SPI to start receiving or sending, write 1 for spi to start working, after that, it will automatically return to zero 1'b0: stop spi working 1'b1: start a send or receive of spi, automatically return to zero Note: master is valid Motorola/TI/microwire mode is valid	0

14.4.3 **SPI_CFG** - SPI Configuration Register

Table 111 SPI Configuration Registers

Bit	Access	Operating Instructions	reset value

[31:19]		reserved	0
[18:17]	RW	FRAM FORMAT 2'b00: motorola 2’b01:TI 2'b10: microwire 2’b11: reserved Choose which manufacturer's protocol the master supports Note: master is valid	0
[16]	RW	SPI_TX pin always driven 1’b0: The spi output is driven only when spi_cs is valid, and is tri-stated at other times 1'b1: The spi output is always driven, even if there is no data transfer Note: Both master/slave are valid Motorola/TI/microwire mode is valid	
[15]		reserved	
[14:12]	RW	cs hold, the time that spi_cs is valid after data transmission is completed, that is, the hold time of spi_cs 3’b000 >=1 APB bus CLK 3’b001 >=2 APB bus CLK 3’b010 >=4 APB bus CLK 3’b011 >=8 APB bus CLK 3’b100 >=16 APB bus CLKs 3’b101 >=32 APB bus CLK 3’b110 >=64 APB bus CLKs 3’b111 >=127 APB bus CLK Note: master is valid Motorola mode is valid	0
[11:9]	RW	cs setup, the time that spi_cs is valid in advance before data transmission, that is, the setup time of spi_cs 3’b000 >=1 APB bus CLK 3’b001 >=2 APB bus CLK 3’b010 >=4 APB bus CLK 3’b011 >=8 APB bus CLK 3’b100 >=16 APB bus CLKs 3’b101 >=32 APB bus CLK 3’b110 >=64 APB bus CLKs 3’b111 >=127 APB bus CLK Note: master is valid Motorola mode is valid	
[8:7]	RW	SPI-out delay, the delay of SPI data output relative to SCK, mainly for hold time consideration. [8:7] Number of system clock cycles (APB clock) 2’b00 0 2’b01 1 2’b10 2 2’b11 3 Note: Both master/slave are valid Motorola mode is valid	0
[6:4]	RW	Frame delay, the default interval between the end of a frame (spi_cs is valid) and the beginning of the next frame is SCK Half of the clock period, which is the SPI_CS inactive time. But for compatibility, it can be configured here. Default at least 0.5SCK [6:4] SCK clock 3’b000 0 3’b001 2 3’b010 4 3’b011 8 3’b100 16 3’b101 32 3’b110 64 3’b111 127 For example, if 128byte data is transmitted in block mode, after the data transmission is completed, the set delay will be added. late time. Note: master is valid	0
[3]	RW	Bigendian 1'b0: The data format adopts the little endian mode, that is, during the transmission process, the low byte is sent first 1'b1: The data format adopts the big endian mode, that is, during the transmission process, the high byte is sent first	0
[2]	RW	MASTER/SLAVE 1'b0: slave, the device is slave 1'b1: master, the device is the master Note: Both master/slave are valid	1
[1]	RW	SPI CPHA 1’b0: Transmission mode A 1’b1: Transmission mode B Note: Both master/slave are valid	0

		Motorola mode is valid	
[0]	RW	SPI CPOL, SCK polarity at IDLE 1'b0: 0 when SCK IDLE 1'b1: 1 for SCK IDLE Note: Both master/slave are valid Motorola mode is valid	0

14.4.4 **CLK_CFG** - Clock Configuration Register

Table 112 SPI Clock Configuration Register

Bit	Ac-cess	Operating Instructions	reset value
[31:16]		reserved	0
[15:0]	RW	Divider FSCK = FAPB_CLK/ (2 x (Divider +1)) Note: master is valid Motorola/TI/microwire mode is valid	0

14.4.5 **MODE_CFG** - Mode Configuration Register

Table 113 SPI Mode Configuration Register

Bit	Ac-cess	Operating Instructions	reset value
[31:9]		reserved	0
[8:6]	RW	RxTrigger level Data stored in RX FIFO triggers interrupt or DMA request threshold: 0~7word Only when the data in the rxbuffer is greater than the RxTrigger level will trigger an interrupt or request a DMA transfer Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[5]		reserved	
[4:2]	RW	TxTrigger level Data stored in TX FIFO triggers interrupt or DMA request threshold: 0~7word Only when the data in the txbuffer is greater than or equal to the TxTrigger level will an interrupt be triggered or a DMA transfer request will be made. shift Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[1]	RW	RxDMA On, use DMA to move data enable 1'b0: Do not use DMA, 1'b1: DMA Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[0]	RW	TxDMA On, use DMA to move data enable 1'b0: Do not use DMA, 1'b1: DMA Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0

14.4.6 **INT_MASK** - Interrupt Control Register

Table 114 SPI Interrupt Control Register

Bit	Ac-cess	Operating Instructions	reset value
[31:8]		reserved	0
[7]	RW	IntEn_spi_timeout 1'b0: enable spi_timeout interrupt generation 1'b1: spi_timeout interrupt is not allowed Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1
[6]	RW	IntEn_spi_done 1'b0: spi send or receive completed, enable interrupt generation 1'b1: The spi send or receive is completed, and interrupts are not allowed Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1
[5]	RW	IntEnRxOverrun 1'b0: Rx FIFO overflow interrupt enable 1'b1: Rx FIFO overflow interrupt disabled Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1
[4]	RW	IntEnRxUnderrun 1'b0: Rx FIFO underflow interrupt disabled	1

		1’b1: Rx FIFO underflow interrupt enable Note: Both master/slave are valid Motorola/TI/microwire mode is valid	
[3]	RW	IntEnTxOverrun 1’b0: Tx FIFO overflow interrupt enable 1’b1: Tx FIFO overflow interrupt disabled Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1
[2]	RW	IntEnTxUnderrun 1’b0: Tx FIFO underflow interrupt enable 1’b1: Tx FIFO underflow interrupt disabled Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1
[1]	RW	IntEnRxFifoRdy 1’b0: Rx FIFO has data upload interrupt enable 1’b1: Rx FIFO has data upload interrupt disabled Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1
[0]	RW	IntEnTxFifoRdy 1’b0: Tx FIFO can write data to TX FIFO interrupt enable 1'b1: Tx FIFO can write data to TX FIFO interrupt disabled Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1

14.4.7 **INT_SRC** - Interrupt Status Register

Table 115 SPI Interrupt Status Register

Bit	Ac-cess	Operating Instructions	reset value
[31:8]		reserved	0
[7]	RW	spi_timeout 1'b0: There is no end data in rxfifo that needs to be taken by the CPU 1'b1: There is end data in rxfifo that needs to be taken by the CPU Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[6]	RW	spi_done 1’b0: SPI transmission or reception is not completed 1'b1: SPI transmission or reception completed Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[5]	RW	RxOverrun 1'b0: Rx FIFO overflow 1'b1: Rx FIFO overflow Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[4]	RW	RxUnderrun 1'b0: Rx FIFO underflow 1'b1: Rx FIFO underflow Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[3]	RW	TxOverrun 1'b0: Tx FIFO overflow 1'b1: Tx FIFO overflow Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[2]	RW	TxUnderrun 1'b0: Tx FIFO underflow 1'b1: Tx FIFO underflow Write 1 to clear In the case of continue mode = 1, the interrupt is never generated. Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[1]	RW	RxFifoRdy 1’b0: Rx FIFO data volume <= RxTrigger level, no need to upload 1’b1: Rx FIFO data volume > RxTrigger level, request to upload Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[0]	RW	TxFifoRdy 1’b0: Tx FIFO data volume > TxTrigger level, cannot write data to TX FIFO 1’b1: Tx FIFO data volume <= TxTrigger level, data can be written to TX FIFO Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0

14.4.8 **STATUS** - SPI Status Register

Table 116 SPI Status Register

Bit	Ac-cess	Operating Instructions	reset value
[31:13]		reserved	0

[12]	RO	SPI Busy 1’b0: SPI has no transmit and receive tasks 1'b1: SPI is in the process of sending or receiving Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[11:6]	RO	RX FIFO fill level The amount of data in the Rx FIFO, in bytes Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[5:0]	RO	Tx FIFO fill level The amount of data in the Tx FIFO, in bytes Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0

14.4.9 **TIMEOUT** - SPI Timeout Register

Table 117 SPI Timeout Register

Bit	Ac-cess	Operating Instructions	reset value
[31]	RW	spi_timer_en 1’b0: timer is not allowed 1'b1: allow timer to count Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0
[30:0]	RW	SPI_TIME_OUT When a transmission is completed, in the receive channel rxfifo, if the data at the end cannot trigger the receive interrupt When RxFifoRdy or DMA request, a timing mechanism needs to be used to notify the CPU to remove the end data. Specific method: When rxfifo is in idle state (no read and write operations, no dma request, cs is invalid, There is data in rxfifo, and the amount of data is less than or equal to the RxTrigger level), start counting and reach the setting of this register. If the set value is set, the timeout interrupt is triggered, and the CPU is requested to remove the end data. Any read or write to rxfifo will clear the timeout timer. The time represented is: T = SPI_TIME_OUT/FAPB_CLK Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0

14.4.10 **TXDATA** - Data Transmission Register

Table 118 SPI Data Transmit Register

Bit	Ac-cess	Operating Instructions	reset value
[31:0]	RW	Window address for writing data to Tx FIFO Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0

14.4.11 **TRANS_MODE** - Transfer Mode Register

Table 119 SPI Transfer Mode Register

Bit	Ac-cess	Operating Instructions	reset value
[31:30]		reserved	0
[29:24]	RW	TI_BLK_LEN In the timing mode of TI, the length of each block transmission, that is, the transmission data length after each CS is valid. Support 4~32bit 6’h4: 4bit long data 6’h5: 5bit long data 6’h6: 6bit long data …….. 6’h20: 32bit long data Note: master is valid TI mode active	0
[23:17]			
[16]		MICRO_BURST 1b’1: In Microwire mode, burst transmission is used, that is, Tx sends control words, Rx receives data, and then Alternately, MICRO_CONTROL_LEN indicates the length of the control word, MICRO_DAT_LEN indicates is the length of the sent or received word, Tx/Rx length indicates the valid sck and burst during the entire transmis-sion process In mode, the number of times of sending and receiving alternately is (Tx/Rx length)/(MICRO_CON-TROL_LEN+ MICRO_DAT_LEN+1) 1’b0: In Microwire mode, burst transmission is not used In this mode, there are two cases 1) tx_ch_on = 1, rx_ch_on = 0, at this time, only sending, MICRO_CONTROL_LEN means Control word length, Tx/Rx length indicates the valid sck during the entire transmission process, the length of the data sent at this time	0

		The degree is m*MICRO_DAT_LEN = Tx/Rx length - MICRO_CONTROL_LEN, where m represents How many (MICRO_DAT_LEN) length words to send 2) tx_ch_on = 1, rx_ch_on = 1, at this time, Tx sends control word, Rx receives data, MICRO_CONTROL_LEN indicates the length of the control word, and Tx/Rx length indicates the entire transmission Valid sck in the process, the length of the received data is m*MICRO_DAT_LEN = Tx/Rx lengthMICRO_CONTROL_LEN-1, where m indicates how many (MICRO_DAT_LEN) words are received Note: master is valid microwire mode works	
[15:14]			
[13:8]		MICRO_DAT_LEN In Microwire mode, in burst mode, the length of data transmitted in each burst From 1 to 32: 6'h1: 1bit long data 6'h2: 2bit long data 6'h3: 3bit long data …….. 6'h20 32bit long data Note: master is valid microwire mode works	
[7:6]			
[5:0]		MICRO_CONTROL_LEN In Microwire mode, the length of the command word From 1 to 32: 6'h1: 1bit long command 6'h2: 2bit long command 6'h3: 3bit long command …….. 6'h20 32bit long command Note: master is valid microwire mode works	

14.4.12 **SLV_LEN** - Data Length Register

Table 120 SPI Data Length Register

Bit	Access	Operating Instructions	Reset
[31:16]	RO	When acting as a slave, the length of the data sent out during the valid period of cs, in bits Note: slave is valid Motorola mode is valid	0
[15:0]	RO	When acting as a slave, during the valid period of cs, the length of the received data, in bits Note: slave is valid Motorola mode is valid	0

14.4.13 **RXDATA** - Data Receive Register

Table 121 SPI Data Receive Register

Bit	Access	Operating Instructions	Reset
[31:0]	RO	Window address to read data from Rx FIFO Note: Both master/slave are valid Motorola/TI/microwire mode is valid	0

15 I2C Controller.

15.1 Function overview.

The I2C bus is a simple, bidirectional two-wire synchronous serial bus. It requires only two wires to transmit information between devices connected to the bus interest. The master device is used to start the bus to transmit data and generate a clock to open the device for transmission. At this time, any addressed device is considered a slave. piece. The relationship between master and slave, sending and receiving on the bus is not constant, but depends on the direction of data transfer at this time. If the master wants to send data to the slave device, the host first addresses the slave device, then actively sends data to the slave device, and finally terminates the data transfer by the host; if the host wants to receive The data of the slave device is first addressed by the master device. Then the host receives the data sent from the device, and finally the host terminates the receiving process. under these circumstances. The host is responsible for generating the timing clock and terminating the data transfer.

15.2 Main Features

- APB bus protocol standard interface
- Can **only be used as a master** device controller
- I2C working rate can be configured, **100KHz~400KHz**
- Multiple GPIOs can be multiplexed into I2C communication interface
- Quickly output and detect timing signals

15.3 Functional Description

15.3.1 Transmission rate selection.

The data transfer rate on the I2C bus can be configured from 100KHz to any bus frequency integer divide value between 400KHz.

15.3.2 Interrupt and start-stop control.

Enable or disable the I2C controller to generate interrupts by setting Bit6 of register CTR, and can also start at any time by setting Bit7 or stop the work of the I2C controller.

15.3.3 Fast output and detection signal.

By setting the corresponding bits of the register CR_SR, the controller can quickly output or detect the bus START signal, bus STOP signal, total Line ACK signal, bus NACK signal. In master mode, the I2C interface initiates data transfers and generates clock signals. a serial data transfer The output always starts with a start signal and ends with a stop signal. Once the start signal is generated on the bus, the master mode is selected.

15.4 Register Description

15.4.1 Register List

Table 122: I2C register list

Offset Address	Name	Abbreviation	Access	Description	Reset Value
0x0000	Clock divider register_1	**PRESCALE_L**	RW	Stores the low-order 8-bit frequency division value for APB The bus clock is divided	0x0000_00FF
0x0004	Clock divider register_2	**PRESCALE_H**	RW	Stores the high-order 8-bit frequency division value for APB The bus clock is divided	0x0000_00FF
0x0008	Control register	**EN**	RW	Is used to control interrupt enable and I2S control enable	0x0000_0040
0x000C	Data register	**DATA**	RW	Is used to store the data to be sent or receive	0x0000_0000
0x0010	Transceiver control register	**CR_SR**	RW	Is used to control some data read and write related operations	0x0000_0000
0x0014	TXR read register	**TXR**	RO	Read TXR register value when I2C is sent	0x0000_0000
0x0018	CR Read register	CR	RO	Read the set value of I2C control register CR	0x0000_0000

15.4.2 PRESCALE_L - Clock divider register_1

Table 123 I2C Clock Divider Register_1

Bit	Access	Description	Reset Value
[31: 8]		reserved	
[7 : 0]	RW	The clock divider configures the higher 8 bits of prescale. For example: apb_clk=40MHz, SCL=100KHz prescale = (40*1000)/(5*100) – 1 = 16'd79 apb_clk = 40M, SCL=400K prescale=(40*1000)/(5*400) – 1 = 16'd19	0xFF

15.4.3 PRESCALE_H - Clock divider register_2

Table 124 I2C Clock Divider Register_2

Bit	Access	Description	Reset Value
[31: 8]		reserved	
[7 : 0]	RW	The clock divider configures the lower 8 bits of prescale. For example: apb_clk=40MHz, SCL=100KHz prescale = (40*1000)/(5*100) – 1 = 16'd79 apb_clk = 40M, SCL=400K prescale=(40*1000)/(5*400) – 1 = 16'd19	8'hff

15.4.4 EN - Control Register

Table 125 I2C Control Register

Bit	Access	Description	Reset value
[31:8]		reserved	
[7]	RW	I2C enable control, 1'b0: Disable 1'b1: enable	0
[6]	RW	interrupt MASK, 1'b0: Enable interrupt generation 1'b1: Interrupt generation is not allowed	1
[5:0]	RW	reserved	

15.4.5 DATA – I2C Data Register

Table 126 I2C Data Register

Bit	Access	Description	Reset Value
[31: 8]		reserved	
[7 : 0]	WR	When writing this register, it is the transmit register TXR, which indicates the next byte to be transmitted, when it is a device address, [0]: 1 means read, 0 means write. When reading this register, it is the receive register RXR, is the latest byte received from I2C.	0x00

15.4.6 CR_SR - Transceiver Control Register

Table 127 I2C Transceiver Control Register

Bit	Access	Description	Reset Value
[31: 8]		reserved	
[7 : 0]	WR	When writing this register, it is CR, and the function is as follows: [7]: STA, control to generate START timing; 1'b0: invalid 1'b1: Generate START timing [6]: STO, control generates STOP sequence; 1'b0: invalid 1'b1: Generate STOP sequence [5]: RD, read from SLAVE; 1'b0: invalid 1'b1: read from SLAVE [4]: WR, write to SLAVE; 1'b0: invalid 1'b1: write to SLAVE [3]: Control sends ACK/NACK back to SLAVE; 1'b0: return ACK 1'b1: return NAK [2:1]: reserved; [0]: IACK, clear the interrupt status, 1 is valid; 1'b0: invalid 1'b1: clear interrupt flag When reading this register, it is SR, and its function is as follows: [7]: RxACK, ACK/NACK status received from SLAVE; 1'b0: ACK received from SLAVE 1'b1: NAK received from SLAVE [6]: BUSY; 1'b0: STO rear 1'b1: STA is followed by 1 [5]: AL, Arbitration Lost, this bit is reserved; [4:2]: reserved; [1]: TIP; 1'b0: No transfer in progress 1'b1: there is a transfer in progress [0]: IF, interrupt status bit; 1'b0: No interrupt is generated 1'b1: Set to 1 when transfer is complete or AL	8'h0

15.4.7 TXR Readout Register

Table 128 I2C TXR Readout Register

Bit	Ac-cess	Description	Reset Value
[31: 8]		reserved	
[7 : 0]	RO	Read only, read value of TXR register, See TXR_RXR register for function description;	0x00

15.4.8 CR read register

Table 129 I2C CR readout register

Bit	Ac-cess	Description	Reset Value
[31: 8]		reserved	
[7 : 0]	RO	Read only, read value of CR register, See CR_SR register for function description;	0x00

бит	доступ	описание	по умолчанию
[31: 8]		зарезервировано	0
[7]	запись	I2C_CR_START, уст. в «1» генерирует START на шине, уст. «0» не разрешена	0
	чтение	I2C_SR_RXACK, если «1» - слейв ответил NACK, если «0» - слейв ответил ACK	0
[6]	запись	I2C_CR_STOP, уст. в «1» генерирует STOP на шине, уст. «0» не разрешена	0
	чтение	I2C_SR_BUSY, если «1» - обнаружен START на шине, сброс в «0» после обнаружения STOP	0
[5]	запись	I2C_CR_RD, уст. в «1» читает байт от слейва, уст. «0» не разрешена	0
	чтение	AL, arbitration lost, зарезервирован	0
[4]	запись	I2C_CR_WR, уст. в «1» отправляет байт слейву, уст. «0» не разрешена	0
	чтение	зарезервировано	0
[3]	запись	I2C_CR_ACK, уст. «1» запрещает отправку ACK слейву при приеме, уст. «0» разрешает ACK	0
	чтение	зарезервировано	0
[2]	запись	зарезервировано	0
	чтение	зарезервировано	0
[1]	запись	зарезервировано	0
	чтение	I2C_SR_TIP, transfer_in_progress, стоит в «1» пока идет передача или прием байта	0
[0]	запись	I2C_CR_IF, уст. «1» сбрасывает флаг прерывания от I2C, уст. «0» не разрешена	0
	чтение	I2C_SR_IF, флаг окончания передачи байта (или AL), флаг прерывания	0

17 UART module

17.1 Function overview

UART is a universal serial data bus used for asynchronous communication. The bus supports bidirectional communication and can realize full-duplex transmission and reception.

W800 has a total of 6 groups of common UART ports, and can realize various baud rate settings through fine clock frequency division combination, and can support up to 2Mbps

communication rate. W800 UART can be used in conjunction with hardware DMA to achieve efficient asynchronous transfer of data.

17.2 Main Features

- Compliant with APB bus interface protocol, full-duplex asynchronous communication mode
- Support interrupt or polling working mode
- Support DMA Byte transfer mode, send and receive each 32-byte FIFO
- Programmable baud rate, up to 2Mbps
- 5-8bit data length, and parity polarity can be configured
- 1 or 2 stop bits configurable
- Support RTS/CTS flow control
- Support Break frame sending and receiving
- Support Overrun, parity error, frame error, rx break frame interrupt indication

17.3 Functional Description

17.3.1 UART Baud Rate

Asynchronous communication requires both parties to send and receive data according to the negotiated baud rate because the two sides do not have the same clock source for reference. W800 fine-grained baud rate control can be achieved through the baud rate setting register, the BAUD_RATE_CTRL register.

BAUD_RATE_CTRL[15:0] is named ubdiv, BAUD_RATE_CTRL[19:16] is named ubdiv_frac, the wave to be set Bit rate baudrate, the formula is as follows:

ubdiv = apbclk / (16 * baudrate) – 1 //Integer

ubdiv_frac = (apbclk % (baudrate * 16)) / baudrate //Integer

Take the APB clock of 40MHz and the baud rate of 19200bps as an example:

ubdiv = 40000000 / (16 * 19200) – 1 = 129

ubdiv_frac = (40000000 % (19200* 16)) / 19200 = 3

According to the above formula, when the APB clock is 40MHz and the baud rate is 19200bps, the baud rate register should be set as:

BAUD_RATE _CTRL = (3<<16) | 129 = 0x0003_0081.

17.3.2 UART Data Format

● Data length

The UART of W800 supports configurable data length of 5bit, 6bit, 7bit and 8bit. The definition of data length is as follows:

Figure 28 UART data length

Normal UART communication consists of 1bit start bit, 1bit stop bit plus the middle data bit, and the middle data bit can be configured,

W800 supports 4 configurable data bits of 5bit, 6bit, 7bit, 8bit, and the data bit length can be selected according to the actual application.

Normal UART communication consists of 1bit start bit, 1bit stop bit plus the middle data bit, and the middle data bit can be configured,

图 28 UART 数据长度

W800 supports 4 configurable data bits of 5bit, 6bit, 7bit, 8bit, and the data bit length can be selected according to the actual application.