BX2400

Bluetooth Low Energy 5.0 SoC

Datasheet

DS-2400-01

Version: 1.7

Released Date: 2020/04/09

Table of Contents

1 General description

The BX2400 is a System-on-Chip combining an application processor, memories, cryptography engine, power management unit, digital and analog peripherals and a MAC engine complied with Bluetooth® Low Energy 5.0 and radio transceiver.

A 32-bit ARM Cortex-M0+ MCU is integrated in BX2400 for BLE link layer management and system operation. Rich digital and analog peripheral interfaces are optimized for external control, including GPIO, SPI, UART, IIC, PWM, ADC and LDOs. Power Management Unit with on-chip DCDC buck converter, Battery Charger and various Regulators provides ultra-low current consumption for BX2400.

2 Feature list

- Complies with Bluetooth BLE 5.0 with 1Mbps/2Mbps data rates
- Supports SIG Mesh
- Can meet BQB/SRRC/FCC/CE standards
- Radio Transceiver:
  - ISM band 2.4~2.5GHz operation
  - -93 dBm RX sensitivity at 1Mbps mode
  - -90 dBm RX sensitivity at 2Mbps mode
  - RF output power levels: -20dBm, 0dBm, 3dBm and 8dBm
  - 50dB RSSI dynamic range
  - 3.4 mA in RX and 3.5 mA in TX with ideal DCDC Converter at 4.3V
  - 4.3 mA in RX and 4.4 mA in TX with on chip DCDC Converter at 4.3V
  - 5.5 mA in RX and 5.6 mA in TX with on chip DCDC Converter at 3.3V
- Clock and PLL:
  - 32MHz Crystal and RC oscillators
  - 32KHz Crystal and RC oscillators
  - 96MHz/80MHz/64MHz/48MHz/32MHz/16MHz PLL
- Analog Interfaces:
  - 10-bit ADC with average capability (Oversampling up to 64 steps providing effectively up to 12 bits resolution)
  - Touch-Key Function
  - Battery monitoring function from 5.5V to 2.0V
  - Temperature sensor from -40℃ to 85℃
- Digital Interfaces:
  - 27 GPIO pins
  - 5 PWM outputs
  - IO State retention in deep sleep mode
  - Quad-SPI Flash interface
  - 2 SPI: up to 24Mbps and each with two chip select signal output
  - 1 SPI slave interface up to 8Mbps
  - 2 UART: flow control up to 1Mbps and supports all the baud rate under 1Mbps, IRDA is supported
  - 2 I2C: master/slave programmable and speed up to 1Mbps
  - 2 timers and 1 watch dog timer
- Integrated ARM Cortex-M0+
  - Clock frequency: 16MHz, 32MHz, 48MHz, 64MHz, 80MHz and 96MHz
  - 4-way associative cache
  - SWD debug interface
  - AHB/APB bus matrix with speed up to 96MHz
- Ultralow Current Mode

  - Sleep current 2.5uA ~ 6uA : SRAM (16 KB ~ 208 KB) retention
  - Average current 20uA during 1.28 sec active(broadcasting ADV)/sleep (208 KB SRAM retention) cycle time
- Memories
  - 208 KB SRAM with retention capability, each 32KB can be set into retention state separately
  - 128 KB ROM (boot ROM and BLE stack)
  - 16 KB 4 way cache controller for external SPI flash which enable CPU run on the external SPI flash, the 16kB cache can be used as system SRAM when cache is disabled.
  - 32KB retention exchange memory for BLE connection data
- Power Management
  - Integrated DCDC buck converter
  - Battery Charger
  - 2.3-5.0V power input
  - One 1.8V LDO 40mA output
  - Two 3.3V LDO 50mA and 30mA output each
- Cryptographic engine:
  - ECC
  - AES-128
- Packages:
  - QFN 52 pins 6mm X 6mm

3 Package

BX2400 provides QFN 52 pins 6 mm x 6 mm packages.

Pin	Symbol	Type	Description
1	P02	DIO	spim0_cs1/FUNC_IO00/GPIO02
2	P03	DIO	spim0_cs0/SPIS_CS/FUNC_IO01/GPIO03
3	P04	DIO	spim0_clk/SPIS_CLK/FUNC_IO02/GPIO04
4	P05	DIO	spim0_miso/SPIS_MISO/FUNC_IO03/GPIO05
5	P06	DIO	spim0_mosi/SPIS_MOSI/FUNC_IO04/GPIO06
6	P15	DIO	FUNC_IO13/GPIO15
7	P16	DIO	FUNC_IO14/GPIO16
8	P17	DIO	FUNC_IO15/GPIO17
9	P07	DIO	spim1_cs1/FUNC_IO05/GPIO07
10	P09	DIO	spim1_clk/FUNC_IO07/GPIO09
11	P10	DIO	spim1_miso/FUNC_IO08/GPIO10
12	P11	DIO	spim1_mosi/FU:NC_IO09/GPIO11
13	P12	DIO	FUNC_IO10/GPIO12
14	P13	DIO	FUNC_IO11/GPIO13
15	VDD_SRAM	PO	VDD_SRAM output
16	VDD_3V_1	PO	Supply to external 3.3V
17	VDD_1V8	PO	Supply to external 1.8V
18	VDD_DIG	PI	Digital circuit power supply
19	GND_D	GND	Ground for digital circuit
20	VDD_1V2	PO	DC/DC Converter output
21	VDD_BAT	PI	Battery supply voltage
22	Ext Reset	DI	Pull low internally. High active.
23	P00	DIO	swck/GPIO00
24	P01	DIO	swd/GPIO01
25	VDD_CPU	PO	VDD_CPU output
26	VDD_AWO	PO	VDD_AWO output
27	P20	DIO	FUNC_IO18/GPIO20

Pin	Symbol	Type	Description
28	P21	DIO	FUNC_IO19/GPIO21
29	P22	DIO	FUNC_IO20/GPIO22
30	P23	DIO	FUNC_IO21/GPIO23
31	P26	DIO	qspi_dat0/GPIO26
32	P25	DIO	qspi_clk/GPIO25
33	P29	DIO	qspi_dat3/GPIO29
34	P28	DIO	qspi_dat2/GPIO28
35	P27	DIO	qspi_dat1/GPIO27
36	P24	DIO	qspi_cs_n/GPIO24
37	XTAL32K_P	AI	32.768 kHz Crystal input (+)
38	XTAL32K_N	AI	32.768 kHz Crystal input (-)
39	VDD_3V_2	PO	Supply to external 3.3V
40	VDD_BAT_2	PI	Guard ring power supply
41	VDD_VCO	PI	VCO power supply
42	LOOP_C	AIO	PLL loop filter external capacitor.
43	VDD_CP	PI	PLL power supply
44	VDD_RF1	PI	RF power supply
45	RF_P	AIO	RF input/output
46	RF_N	AIO	RF input/output
47	VDD_A	PI	Power supply for an analog circuit
48	VDD_BAT_1	PI	ADC power supply
49	P30	AI	ADC Input Channel 0
50	P08	DIO	spim1_cs0/FUNC_IO06/GPIO08
51	XTAL32M_P	AI	32 MHz Crystal input (+)
52	XTAL32M_N	AI	32 MHz Crystal input (-)
IC Ground pad		GND	Backside GND plane. Must be connected to the GND.

NOTE: AI : analog input AO : analog output AIO : analog input/output DI : digital input DIO : digital input/output PI : power input PO : power output

4 System overview

4.1 Electronical Characteristics

4.1.1 Absolute maximum ratings

Parameters	Min	Max	Unit
Storage temperature	-40	+120	°C
Voltage applied to inputs	-0.5	+5.5	V

4.1.2 Recommended Operating Conditions

Parameters	Min	Typ	Max	Units
Ambient Operating Temperature	-40		+105	°C
Supply Voltage for VDD_3V (*)	1.9	4.3	5	V
Logical high input voltage (for DI type pins)	0.85 x VDD_IO		VDD_IO	V
Logical low input voltage (for DI type pins)	0		0.2 x VDD_IO	V

Note : VDD_IO is programmable as 3.3V or 1.8V individually.

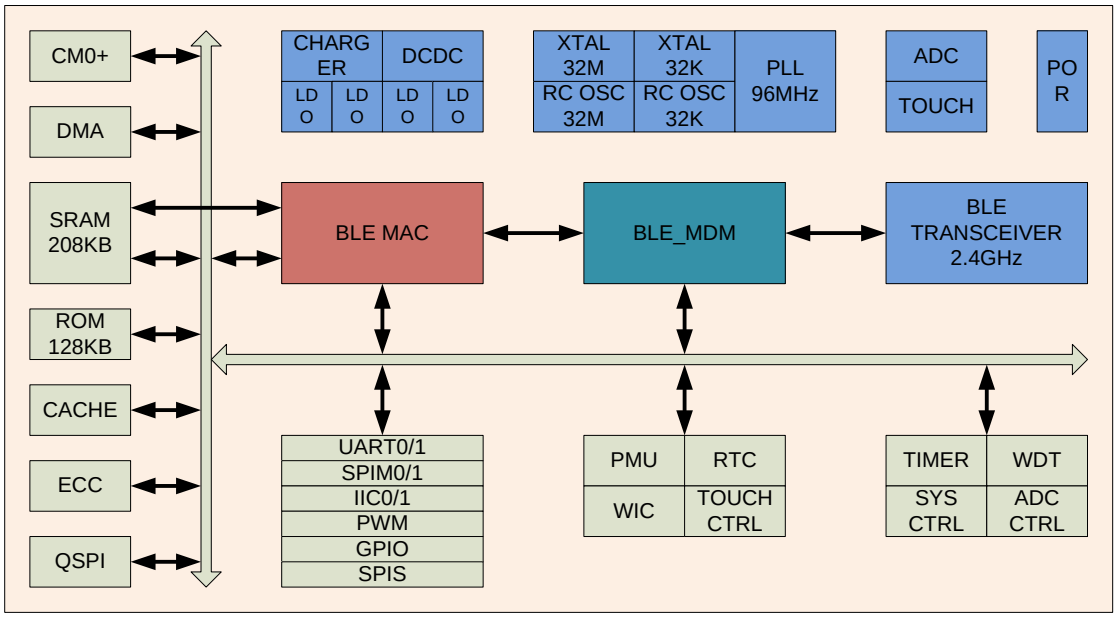
4.1.3 Radio Frequency Characteristics

TA = 25oC, VDD = 4.3 V, Frequency=2.440 GHz

Parameter	Condition	Min	Typ	Max	Unit
Receiver Sensitivity	At 1 Mbps RX Mode, 37 Byte payload length, BER<0.1%.	-93			dBm
	At 2 Mbps RX Mode, 37 Byte payload length, BER<0.1%	-90			dBm
Transmitter Output Power	Maximum RF Output Power at 0 dBm setting (Comply with Power Class 3 in Bluetooth Low Energy 5.0)	-4	-2	0	dBm
	Maximum RF Output Power at 8 dBm setting (Comply with Power Class 1.5 in Bluetooth Low Energy 5.0)	5	6.5	8	dBm
Current Consumption	Receiver with integrated DC/DC Converter		4.3		mA
	0 dBm Transmitter with integrated DC/DC Converter		4.4		mA
	8 dBm Transmitter with integrated DC/DC Converter		26		mA
	Sleep with SRAM (16KB ~ 208KB) retention, 32.768KHz Crystal Clock	2.5		6	uA
	Average current during 1.28 sec active (broadcasting ADV) /sleep (SRAM 208 KB retention) cycle time		18		uA
	CPU Standby with 16MHz Clock running		1.5		mA

4.2 Block diagram

The block diagram of BX2400 is shown below.



4.3 System blocks

ARM Cortex M0+ CPU: The Cortex-M0+ processor is an entry-level 32-bit ARM Cortex processor designed for a broad range of embedded applications. It offers significant benefits to developers, including:

- A simple architecture that is easy to learn and program.
- Ultra-low power, energy-efficient operation.
- Excellent code density.
- Deterministic, high-performance interrupt handling.
- Upward compatibility with Cortex-M processor family.
- Platform security robustness, with integrated Memory Protection Unit (MPU).

BLE 5.0 MAC and PHY: This is the BLE MAC and PHY which is compatible with the BLE 5.0 protocol.

BLE RF module: This is the low power RF module which is compatible with the BLE 5.0 protocol.

Cache Controller: This is the 32bit 4 way read only cache controller which makes the CPU can run on the external flash with quard-SPI interface. The cache controller offers one cycle latency read access when the read from the CPU hit the cache. If cache miss, the cache controller is responsible to generate the SPI access command and fetch data from the external flash through quard-SPI interface.

AHB/APB bus matrix: This is the bus matrix which provides the data access channel between multiple AHB masters and multiple AHB/APB slaves. The access of different master slave pair does not affect each other which improve the data throughput of the system.

RAM controller: This is the SRAM controller which converts the AHB bus access to the 208KB SRAM read/write command. Because one of the SRAM is shared with the exchange memory used by the BLE MAC, the SRAM controller is also responsible for the arbitration between the AHB bus and the BLE MAC.

ROM controller: This is the ROM controller which converts the AHB bus access to the 128KB ROM read command. The boot code and the BLE link layer stack are stored in the ROM.

ROM patch controller: This is the data patch controller for ROM data. It provides data patching for up to 16 ROM addresses.

System controller(CPU): This is the system controller which controls the clock generator, reset generator and the pin share logic of the system. It is also responsible for frequency calibration of the 32KHz clock from RC oscillator.

DMA controller: This is the DMA controller which provides directly data access channel between the SRAM and peripheral interfaces. There are 6 DMA channels implemented in the DMA controller.

Quard-SPI controller: This is the SPI master interface with 4 data IOs. It provides the data channel between the AHB interface and the external flash through quard-SPI interface. It supports DMA access by the DMA controller. The maximum data bandwidth is 96Mbps.

ADC controller: This is the ADC controller which provides the data channel between the ADC and the APB bus. It supports both one time ADC sample and continuous ADC sample. It also supports DMA access by the DMA controller in the continuous sampling mode.

ECC engine: This is the ECC codec for the encrypt transmission of BLE. It provides hardware fast ECC calculation which costs less than 50ms for single ECC calculation.

Pin share controller: This is the pin share logic which provides flexible pin share scheme for different customers. The pin share logic is controlled by the system controller(CPU).

Clock generator(CPU): This is the clock generator which provides clock for all of the modules in the CPU power domain. It implements the clock divider, clock mux/switch and architecture clock gating for the CPU power domain.

Reset generator(CPU): This is the reset generator which provides the reset for all of the

modules in the CPU power domain. It implements the reset synchronizer and the software reset logic for the CPU power domain.

UART2AHB: This is the UART interface module which provides the access channel to all of the system address space for the external UART controller. It can provide the access channel without the help of the CPU which means that even if the CPU does not work correctly, the external UART controller can access all the registers and memory space. It is mainly for debug.

Timer: This is the timer counter which is counting in a frequency programmable clock. There are two independent timer counters implemented.

Watch dog: This is the watch dog controller for the system, which can work in two modes: system reset mode and interrupt followed by system reset mode. The watch dog can prevent system from entering some dead status by interrupt and reset the system if the system does not kick the watch dog for a long time.

System controller(PER): This is the system controller which controls the clock generator, reset generator of the PER power domain.

Clock generator(PER): This is the clock generator which provides the clock for all of the modules in the PER power domain. It implements the clock divider, clock mux/switch and architecture clock gating for the PER power domain.

Reset generator(PER): This is the reset generator which provides the reset for all of the modules in the PER power domain. It implements the reset synchronizer and the software reset logic for the PER power domain.

UART interface controller(UART0/UART1): Asynchronous serial interface controller with throughput up to 2Mbps. UART0 supports flow control and UART1 does not. Both of the UART interface controller support DMA access by DMA controller.

IIC interface controller(IIC0/IIC1): This is the IIC interface controller which can be programmed to be master or slave. It supports DMA access by DMA controller.

SPI master interface controller(SPIM0/SPIM1): This is the SPI master interface with one bit data input and one bit data output and two bit chip select. The maximum throughput is 24Mbps. It supports DMA access by DMA controller.

SPI slave interface controller(SPIS): This is the SPI slave interface with one bit data input and one bit data output. The maximum throughput is 6Mbps. It supports DMA access by DMA controller.

PWM controller: This is the PWM waveform generator which generates 5 independent PWM output signal. The frequency and the duty cycle of the PWM signal are programmable.

GPIO controller: This is the general purpose IO controller which implements 30 GPIOs. The direction and output value are both programmable. And also the interrupt mode is programmable to edge and level.

Power management unit(PMU): This is the power management controller of the system which controls the power up and power down flow for each power domain. All of the DCDC/CHARGER/LDO are controlled by PMU according to the internal FSM of the PMU.

Touch controller: This is the ADC controller for touch function. The sampled data from ADC is processed and compared with a programmable threshold and an interrupt is triggered if necessary.

Wakeup interrupt controller(WIC): The wakeup interrupt controller monitor the wakeup interrupts and inform the PMU to power up the system if necessary. The wakeup controller also wakeup CPU after the system has been powered up.

Mode controller: The mode controller monitor the boot select IO value during the system power up reset active period and inform the CPU from which interface the CPU can get the boot loader and the IO voltage.

Pad ring: All the digital IO cell is implemented in this module.

Power PWM controller: This controller is used to control the power output. The power output can be set to on or off. And also the power output can be set to on for some time and off for some time just like a PWM waveform.

Real time controller: This is the real time timer for the system.

System controller(AWO): This is the system controller which controls the clock generator, reset generator, touch controller, power PWM, PMU, pad ring of the AWO power domain. The IO retention function is also implemented in this module.

Clock generator(AWO): This is the clock generator which provides all the clock for the AWO power domain and all the other power domain. It implements the clock divider, clock mux/switch and architecture clock gating for the AWO power domain.

Reset generator(AWO): This is the reset generator which provides all the reset for the AWO power domain and all the other power domain. It implements the reset synchronizer and the software reset logic for the AWO power domain.

4.4 Function mode

BX2400 has two functional modes of operation: Mirrored mode and Cached mode.

In Mirrored mode, the system code is mirrored from the external device to the system SRAM, and CPU is running on the system SRAM. In this mode cache controller is disabled. And the 16KB SRAM used by the cache controller can be used as the normal system SRAM.

In Cached mode, the system code is stored in the external flash, and CPU is running on the external flash. In this mode cache controller is enabled and the 16KB SRAM used by the cache controller can not be used as the normal system SRAM.

Cache mode is used only when the system SRAM is not big enough for system running. Cache mode will cost more power than the mirrored mode and the performance of the CPU in cache mode is much worse than it is in the mirrored mode.

4.5 System boot sequence

After power up, power up reset will hold low for some time. Then after the power up reset is released, CPU will start to execute code from address 0x0. No address remapping is implemented in BX2400 and ROM is always addressed at address 0x0. So after power up the boot sequence in the ROM is executed by CPU.

By default, all of the power sources are powered up after power on reset, and the 32MHz crystal and 32KHz RC oscillator are active. PLL is off by default. 32MHz clock is the main clock source of the system. CPU is working on 32MHz by default.

BX2400 can boot from QSPI interface or UART interface. The IO voltage of the boot interface can be 1.8V or 3.3V. This is decided by two boot select pins. The two boot select pin shall be pulled up or pulled down to predefined value during power on reset period. The value of the boot select pins are latched to the internal boot registers during the power on reset period and be read by the CPU at the beginning of the boot sequence. Then CPU can decide which interface shall boot from and how much is the IO voltage level of the interface by reading the boot registers implemented in the system controller. The clock frequency of QSPI interface during boot period is 8MHz and the data rate of the UART interface during boot period is 115200bps. If boot from UART then P12 is used as TX data of UART interface and P13 is used as the RX data of UART interface.

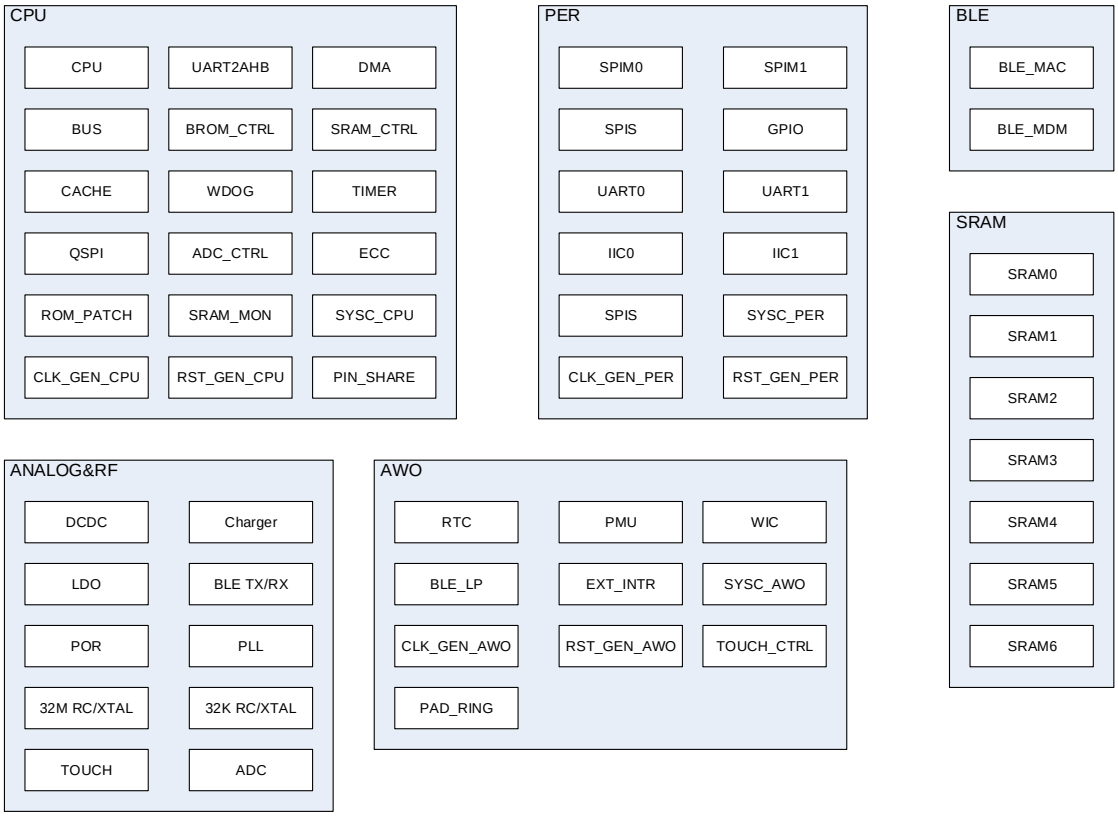
The decode logic of the boot select pin is shown below:
P16 is pulled down means boot from QSPI, P16 is pulled up means boot from

UART0(P12/P13). P23 is pulled down means IO voltage is 1.8V, P23 is pulled up means IO voltage is 3.3V. The default IO voltage is 1.8V.

The boot sequence in the ROM is shown below:

4.6 Power domain

The block diagram of the power domain of BX2400 is shown below:



BX2400 is composed of 6 power domains which are CPU, PER, BLE, AWO, SRAM and ANALOG.

ANALOG power domain includes all of the analog and RF submodules and each analog and RF submodule is a dedicated sub power domain and can be powered on/off separately.

The SRAM power domain includes all of the 208kB system SRAM and is divided into 7 SRAM blocks. The size of each SRAM block is 32kB except the last block which is 16kB. Each of the 7 SRAM blocks is a dedicated sub power domain and can be powered on/off separately. The address mapping for the 7 SRAM blocks is shown below:

	start address	end address
SRAM0	0x100000	0x107FFF
SRAM1	0x108000	0x10FFFF

SRAM2	0x110000	0x107FFF
SRAM3	0x118000	0x11FFFF
SRAM4	0x120000	0x127FFF
SRAM5	0x128000	0x12FFFF
SRAM6	0x130000	0x133FFF

The PER power domain includes all of the digital peripheral interface modules except the QSPI interface controller. The PER power domain can work under 0.9V or 1.1V. The PER power domain can be powered off.

The BLE power domain includes BLE MAC and BLE PHY. The BLE power domain can work under 0.9V or 1.1V. The BLE power domain can be powered off.

The CPU power domain includes the core system which includes CPU, DMA, SRAM controller, bus matrix, cache controller, ECC module, system timer and system watch dog. CPU power domain can work under 0.9V or 1.1V. The CPU power domain can be powered off.

The AWO power domain includes the power management module, the IO ring, and the logic which can wake up system from sleep status which includes RTC, BLE low power counter, external interrupt controller and touch key controller. The AWO power domain can not be powered off. AWO power can only work under 0.9V.

All of the other power domains except the AWO and ANALOG power domain work under the same voltage in active mode. Under inactive mode (retention mode) the voltage of SRAM can be programmed to be less than 0.9V to save leakage power of SRAM. Each of the BLE/PER/SRAM* power domains can be powered off independently. And if CPU is powered off, BLE and PER must be powered off and SRAM* must be powered off or be set to retention status. The power on and power off sequence is controlled by hardware FSM and triggered by software.

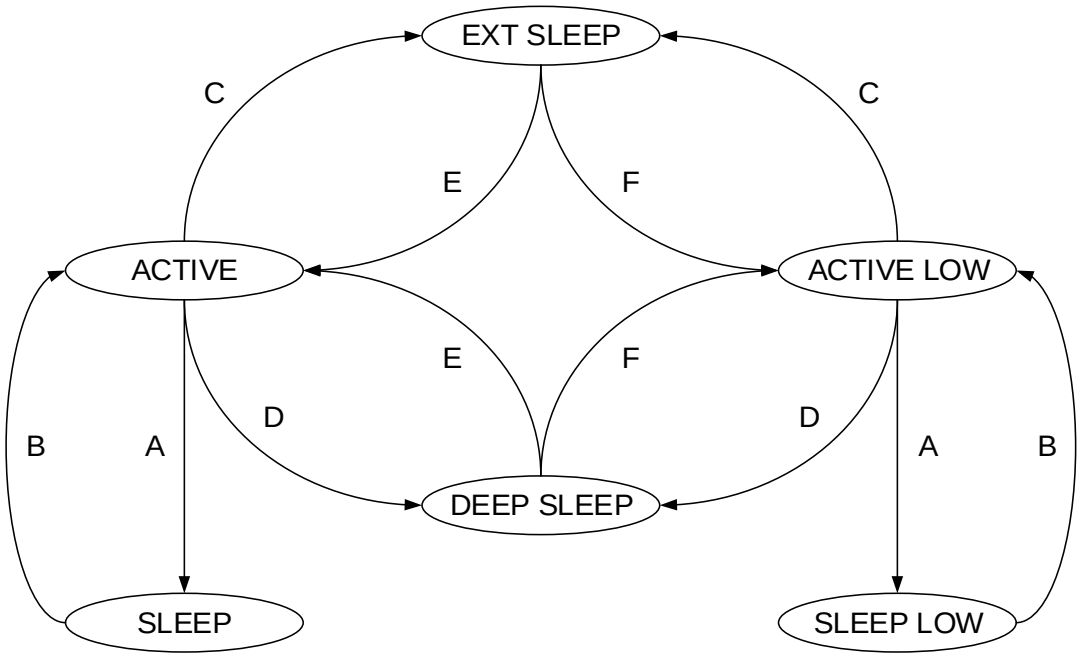
BX2400 has 6 power modes which are described in the table below:

power mode	description	AWO	CPU	PER	BLE	SRAM	ANALOG
ACTIVE	AWO is working CPU is working. The working frequency of CPU is more than 32MHz, and CPU is working under 1.1V at least one of the SRAM is working(power on) all of the other power domain is on/off programmable by CPU	ON	ON	P	P	P	P

power mode	description	AWO	CPU	PER	BLE	SRAM	ANALOG
ACTIVE LOW	AWO is working CPU is working. The working frequency of CPU is equal to or less than 32MHz, and CPU is working under 0.9V at least one of the SRAM is working(power on) all of the other power domain is on/off programmable by CPU	ON	ON	P	P	P	P
SLEEP	AWO is working CPU is clock gated and the working voltage of CPU is 1.1V at least one of the SRAM is working(power on) all the other power domain is on/off programmable by CPU	ON	ON	P	P	P	P
SLEEP LOW	AWO is working CPU is clock gated and the working voltage of CPU is 0.9V at least one of the SRAM is working(power on) all the other power domain is on/off programmable by CPU	ON	ON	P	P	P	P
EXTENDED SLEEP	AWO is working under 32KHz CPU, PER and BLE are powered off At least one of the SRAM is in retention state All of the other power domain is power off	ON	OFF	OFF	OFF	ON/OFF	OFF
DEEP SLEEP	PD_AWO is working under 32KHz CPU, PER and BLE are powered off All of the SRAM are off All of the other power domain is power off	ON	OFF	OFF	OFF	OFF	OFF

Note:
ON: power on
OFF: power off
P: ON/OFF programmable

The power state machine of the power states is shown in the block diagram below:

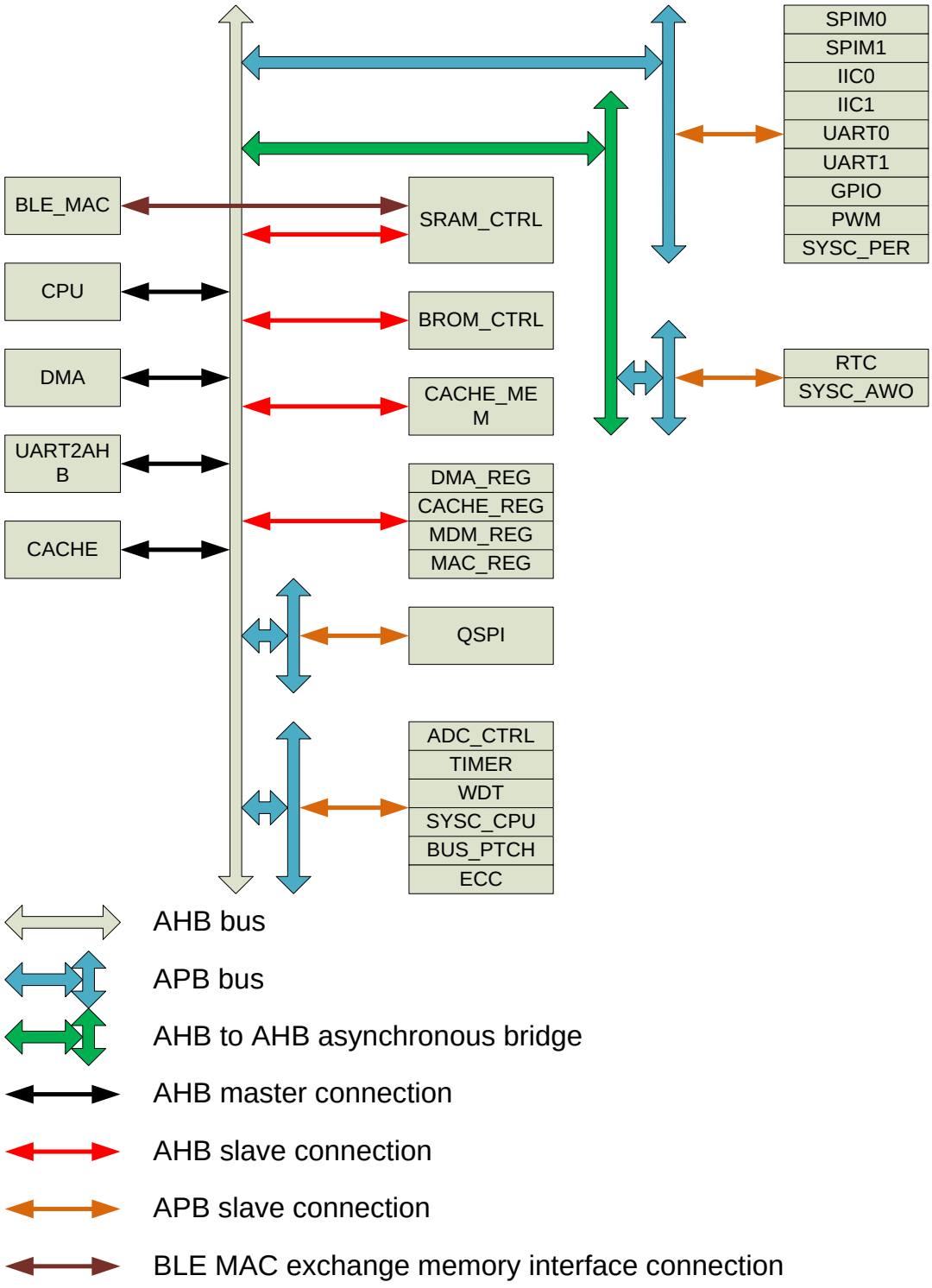


The power state transfer condition in the above diagram is shown in the table below:

A	CPU execute WFI with DEEP_SLEEP bit set to low
B	any non-masked interrupt is active
C	CPU execute WFI with DEEP_SLEEP bit set to high, and not all of the active SRAM is powered off together with CPU
D	CPU execute WFI with DEEP_SLEEP bit set to high, and all of the active SRAM are powered off together with CPU
E	Any of the external interrupt, BLE low power interrupt, RTC interrupt and touch interrupt is active and CPU is waked up with the register VDD_VOLTAGE is set to low which means CPU will work under 1.1V after wake up
F	Any of the external interrupt, BLE low power interrupt, RTC interrupt and touch interrupt is active and CPU is waked up with the register VDD_VOLTAGE is set to high which means CPU will work under 0.9V after wake up
note: register VDD_VOLTAGE is a programmable register in AWO power domain which address is 0x20201048[0]	

5 Bus architecture

The bus architecture of BX2400 is shown below.



The system bus is based on AHB and APB. The data width of the bus is 32 bits. The AHB bus works under the same frequency with CPU. The maximum frequency of AHB bus is 96MHz

and programmable with 16MHz step. The maximum frequency of APB bus is one half of the frequency of AHB bus which is 48MHz. The frequency of APB bus is the integer division of the frequency of the AHB bus. There are altogether 4 AHB masters which are CPU, DMA, UART2AHB and CACHE. The exchange memory(EM/32KB) required by the BLE MAC is 32KB and shared with system SRAM. So the BLE MAC can access the SRAM directly through an internal interface. BLE MAC has the highest priority when any other AHB master accesses the same SRAM with BLE MAC at the same time. The accessibility and the priority of the AHB masters to the slaves is shown in the table below:

	BROM_CTRL	CACHE_MEM	SRAM_CTRL	OTHER	AWO_REG	QSPI
CPU	0	0	1	0	0	1
UART2AHB	X	X	2	1	1	2
DMA	X	X	3	2	X	3
CACHE	X	X	X	X	X	0
BLE_MAC	X	X	0	X	X	X

In this table, X means the master can not access the slave. Number means the master can access the slave and 0 has the highest priority. The cache size in the bus diagram is 16kB which is shared with the system SRAM. When cache is enable, the last 16kB system SRAM address space must not be accessed by CPU or other bus master. The last 16kB SRAM is used as the cache SRAM at that time.

The size of the system SRAM is 208kB and the address starts with 0x100000. The system SRAM is composed of 7 sub SRAM blocks as described in chapter 4.5. And if one of the SRAM sub block is not needed any more in the system it can be powered of to save power. BLE_MAC can only access the fifth SRAM sub block which address starts with 0x128000. When the CPU is running on the external flash, the flash controller must be enabled and the last SRAM sub block is used as the cache and can not accessed by the other AHB master.

6 Address mapping

The address mapping of BX2400 is shown below:

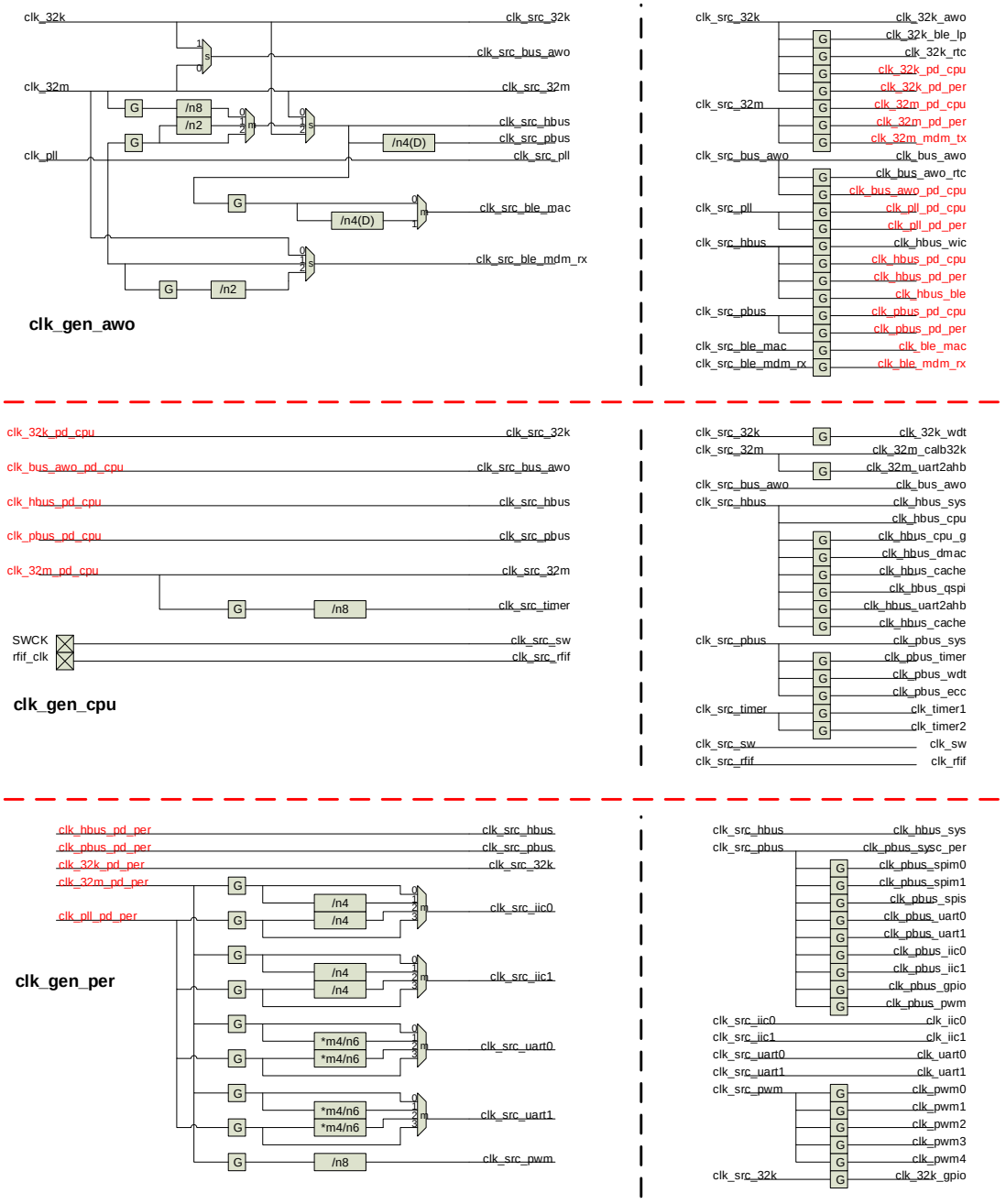
addr zone			start	end	space
brom			0x00000000	0x0001FFFF	128kB
reserved			0x00020000	0x000FFFFF	896kB
sram			0x00100000	0x00133FFF	208kB
reserved			0x00134000	0x001FFFFF	816kB
reserved			0x00200000	0x007FFFFF	6MB
cache			0x00800000	0x00FFFFFF	8MB
reserved			0x01000000	0x1FFFFFFF	496MB
reserved			0x20000000	0x200FFFFF	1MB
peripheral	ahb	ble_mac_reg	0x20100000	0x2010FFFF	64kB
		ble_mdm_reg	0x20110000	0x2011FFFF	64kB
		dma_reg	0x20120000	0x20120FFF	4kB
		cache_reg	0x20121000	0x20121FFF	4kB
		reserved	0x20122000	0x2012FFFF	56kB
	apb0	timer	0x20130000	0x20130FFF	4kB
		wdt	0x20131000	0x20131FFF	4kB
		sysc_bus	0x20132000	0x20132FFF	4kB
		bus_ptch	0x20133000	0x20133FFF	4kB
		ecc	0x20134000	0x20134FFF	4kB
		reserved	0x20135000	0x20135FFF	4kB
		adc_ctrl	0x20136000	0x20136FFF	4kB
		reserved	0x20137000	0x2013FFFF	36kB
	apb3	spim0	0x20140000	0x20140FFF	4kB
		spim1	0x20141000	0x20141FFF	4kB
		spis	0x20142000	0x20142FFF	4kB
		uart0	0x20143000	0x20143FFF	4kB
		uart1	0x20144000	0x20144FFF	4kB
		iic0	0x20145000	0x20145FFF	4kB
		iic1	0x20146000	0x20146FFF	4kB
		pwm	0x20147000	0x20147FFF	4kB
		gpio	0x20148000	0x20148FFF	4kB
		sysc_per	0x20149000	0x20149FFF	4kB
		reserved	0x2014A000	0x2014FFFF	24kB
reserved			0x20150000	0x200FFFFF	704kB
awo	apb1	rtc	0x20200000	0x20200FFF	4kB
		sysc_awo	0x20201000	0x20201FFF	4kB

addr zone			start	end	space
qspi	apb2	qspi	0x20300000	0x20300FFF	4kB

7 Clock system

7.1 General description

The block diagram of the clock system is shown below:



The clock system is comprised of three subsystems which are clk_gen_awo, clk_gen_cpu and clk_gen_per. Clk_gen_awo is in the AWO power domain and

generates clocks for the submodules of AWO and the system bus clock for the other power domains. Clk_gen_per is in the PER power domain and generates clocks for the submodules of PER power domain. Clk_gen_cpu is in the CPU power domain and generates clocks for the submodules of CPU power domain.

There are three main clock sources in the clk_gen_awo which are clk_32k, clk_32m and clk_pll. These three clocks are from the ANALOG power domain and are the main clock sources of the most of the blocks in the digital part. The clocks of the blocks in the digital part are either directly from these three clocks or the division version of these three clocks. SWCK and rfif_clk in the clk_gen_cpu is from the system IO. clk_32k is the 32KHz clock of the system.

clk_32k has two sources which are 32KHz crystal oscillator and the 32KHz RC oscillator. The 32KHz crystal oscillator is composed of the external crystal and the internal oscillating logic, the frequency of the 32KHz crystal oscillator depends on the external crystal. The 32KHz RC oscillator does not need external components and output a clock which maximum frequency is 100KHz. The frequency of the RC oscillator is programmable and can be calibrated by the calibration logic in the CPU power domain.

clk_32M has two sources which are 32MHz crystal oscillator and the 32MHz RC oscillator. The 32MHz crystal oscillator is composed of the external crystal and the internal oscillating logic, the frequency of the 32MHz crystal oscillator depends on the external crystal. The 32MHz RC oscillator does not need external components and output a clock which maximum frequency is 32MHz. The frequency of the RC oscillator is programmable and can be calibrated by the calibration logic in the CPU power domain.

Clk_pll is from the PLL module in the ANALOG power domain. The maximum frequency of the clk_pll is 96MHz and can be programmed at 16MHz step.

Clk_32k, clk_32m and clk_pll can be enabled or disabled by programmable registers. If the clock is not needed, it can be disabled to save power.

The clock signals on the right side of the above diagram is the clock output of the current clk_gen. The clock signals marked in red are the clocks for the other power domain. The clock signals marked in black are the clocks for the sub blocks in the same power domain. The clock signals on the left side of the above diagram is the clock input of the current clk_gen. The clocks marked in red are the clocks from the other power domain. The clock signals marked in black are clocks from the same power domain.

Block with “s” represents clock switch which is a glitch free clock mux. The select

signal of the clock switch can be changed any time and no glitches will be found on the output clock.

The block with “m” represents a clock mux. The clock mux is not glitch free. So before change the selection of the mux, the gating before the clock mux must be closed.

The block with “G” represents clock gating. There is synchronization logic for the control signal of the clock gating. When the input control signal is low, the clock gate is closed and the output clock is tied to ground. When the input control signal is high, the clock gate is opened and the output clock is the same as the input clock.

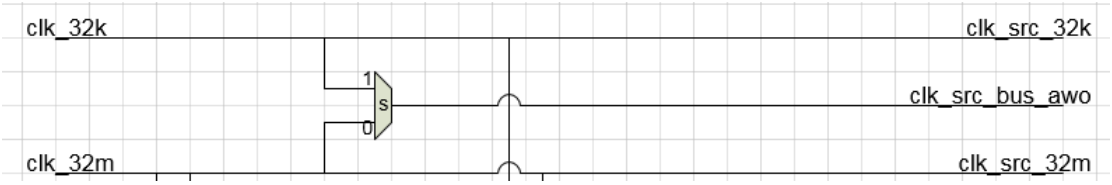
The block with “/nx”(x is a number) represents an x bit width integer static clock divider. The output clock frequency is 1/(n+1) of the input clock frequency with n is the value of the x bit divide parameter from the control register. Static clock divider means when the x bit divide parameter changes, there may be glitches on the output clock. So to avoid the glitch on the output clock, the input clock must be gated before the divide parameter changes. And after the divide parameter has changed, open the input clock again.

The block with “*my/nx” (x and y are number) represents an x bit width fractional static clock divider. This kind of clock divider has 4 divide parameters which are num0, num1, len0 and len1. Parameter num0 and num1 are x bits wide and parameter len0 and len1 are y bits wide. The output clock is comprised of a circulation of (num0+1) clock cycle with (len0+1) divide parameter followed by (num1+1) clock cycle with (len1+1) divide parameter. The output clock average frequency is (num0+1+num1+1)/((num0+1)*(len0+1)+(num1+1)*(len1+1)) of the input clock frequency with num0 and num1 is x bit divide parameter and len0 and len1 is the x bit divide parameter. Static clock divider means when the x bit divide parameter changes, there may be glitches on the output clock. So to avoid the glitch on the output clock, the input clock must be gated before the divide parameter changes. And after the divide parameter has changed, open the input clock again.

The block with “/nx(D)”(x is a number) represents an x bit width integer dynamic clock divider. This kind of clock divider is dynamic clock divider which means that the clock divide parameter can be changed at any time. This clock divider has a clock divide parameter input and together with a parameter change indication input signal. When the value of the parameter change indication input signal changes, the value divide parameter input will be updated to the output clock. So when change the output clock frequency, just change the divide parameter input first and then toggle the indication input, then the output clock frequency changes and no glitches will be found on the output clock.

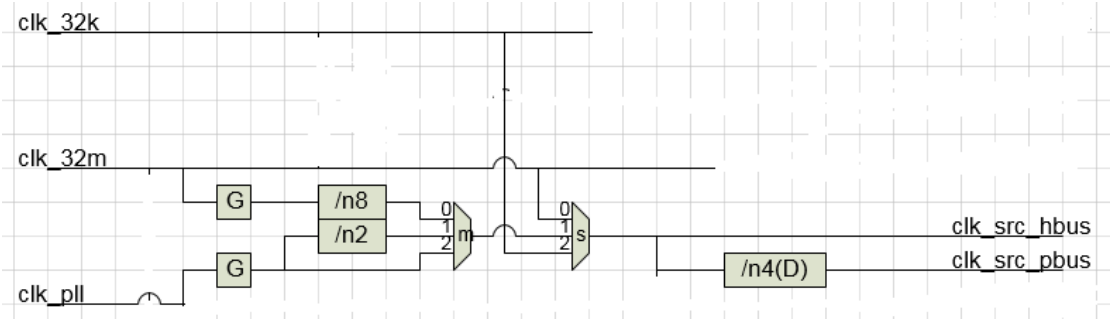
7.2 Control for clk_src_bus_awo

Clk_bus_awo is the bus clock for the AWO power domain. This clock comes from clk_32k and clk_32m. The selection between the two clocks is not controlled by the programmable register. It is controlled by the FSM of the system PMU block. When the system goes into extended sleep status or deep sleep status, clk_bus_awo comes from clk_32k to save power during sleep status. When system wakes up, clk_bus_awo comes from clk_32m to improve the system performance.



7.3 Control for clk_src_hbus

Clk_src_hbus is the clock of the AHB bus of the system. It comes from clk_32k, clk_32m and clk_pll.



The clock gating on the path of clk_32m is controlled by the programmable registers 0x20201004[7:6]. The name of the registers are clkg_hbus_div_32m_clr and clkg_hbus_div_32m_set. Writing 1 to 0x20201004[7] will close the clock gate(no clock will be output). Writing 1 to 0x20201004[6] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

The clock gating on the path of clk_pll is controlled by the programmable registers 0x20201004[5:4]. The name of the registers are clkg_hbus_div_pll_clr and clkg_hbus_div_pll_set. Writing 1 to 0x20201004[5] will close the clock gate(no clock will be output). Writing 1 to 0x20201004[4] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

The divider parameter of the static integer clock divider “/n8” and “/n2” are from the same programmable control register 0x20201000[15:8] which is named as clk_div_pbus_para_m1. The clock divider on the path of clk_pll is a two bit width clock divider, and the divider parameter is the low 2 bits of the clk_div_pbus_para_m1. If the register clk_div_pbus_para_m1 changes then the frequency of the output clocks of both clock dividers will change. Both of the two clock dividers are static dividers which means before change the clock divide parameter, the clock gate before the clock divider must be close first to avoid clock glitches on the output clock. The frequency of the output clock of the clk_32m clock divider is 1/(clk_div_pbus_para_m1+1) of the clk_32m. The frequency of the output clock of the clk_pll clock divider is 1/(clk_div_pbus_para_m1[1:0]+1) of the clk_pll.

The clock mux “m” is controlled by the programmable register 0x20201000[4:3] which is named as clk_sel_hbus1. When the register is 0, the clock from clk_32m divider is selected. When the register is 1, then the clock from the clk_pll divider is selected. When the register is 2, then clk_pll is selected.

The glitch free clock switch “s” is controlled by the programmable register 0x20201000[2:0] which is named as clk_sel_hbus0. This register is a one hot register which means at any time only one bit of this register can be high. When the register is 001, clk_32m is selected. When the register is 010, the clock from the clock mux is selected. When the register is 100, clk_32k is selected. Because the clock switch is glitch free, so the control register can be programmed at any time. But when the control register is programmed, the two clocks which are selected and unseleted must be active.

The programming flow is as below:

The target is to select clk_32m

- Writing 001 to 0x20201000[2:0] to select clk_32m
- Writing 1 to 0x20201004[7] to close the clock gate on the clk_32m path
- Writing 1 to 0x20201004[5] to close the clock gate on the clk_pll path

The target is to select clk_32k

- Writing 100 to 0x20201000[2:0] to select clk_32k
- Writing 1 to 0x20201004[7] to close the clock gate on the clk_32m path
- Writing 1 to 0x20201004[5] to close the clock gate on the clk_pll path

The target is to select the output from the clock mux

- Writing 001 to 0x20201000[2:0] to select clk_32m. If current clock is clk_32m, then this step can be omitted.
- Writing 1 to 0x20201004[7] to close the clock gate on the clk_32m path
- Writing 1 to 0x20201004[5] to close the clock gate on the clk_pll path
- Writing register 0x20201000[15:8] to set the clock divide parameter
- Writing register 0x20201000[4:3] to select the expected clock
- Writing 1 to 0x20201004[6] or 0x20201004[4] to open the clock gate on the path of

clk_32m or clk_pll.

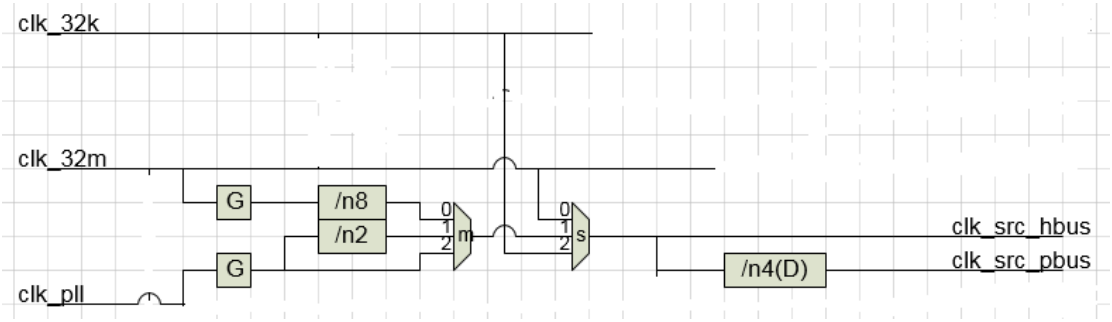
- Writing 010 to 0x20201000[2:0] to select the clock from the clock mux.

Note: If the clock clk_pll is needed, then before the above steps, the control registers for the analog module PLL must be programmed first and then wait for the stable of the PLL. If after the above step, the clk_pll is not needed any more, then the analog module PLL can be powered off to save power.

Clk_src_hbus can also be controlled by the system power management logic. When the system goes into sleep state, the glitch free clock switch “s” will be set by PMU to select clk_32k to save power of the sleep status. When the system wakes up from the sleep state, the glitch free clock switch “s” will be set by PMU to select clk_32m to speed up the system. After that the software can program the clk_src_hbus to expected frequency through the steps above.

7.4 Control for clk_src_pbus

Clk_src_pbus is the clock of the APB bus of the system. It comes from clk_src_hbus.



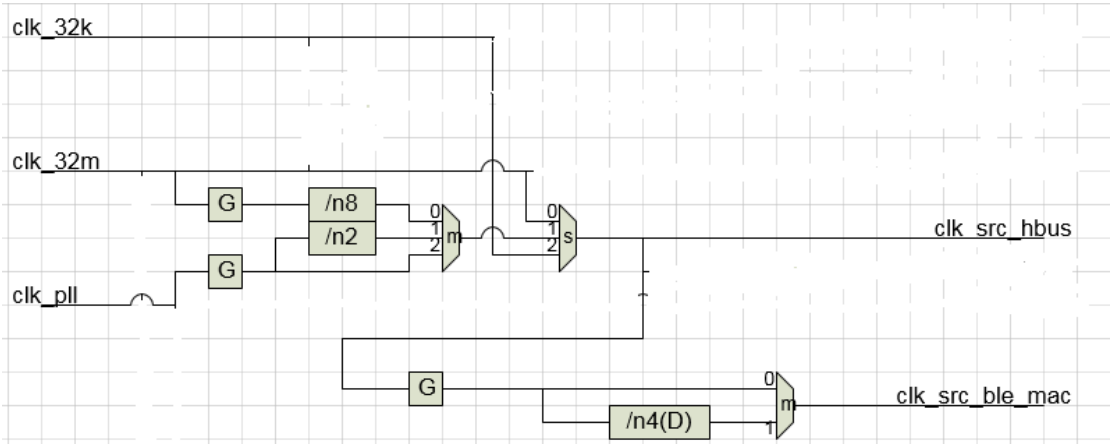
The clock divider “/n4(D)” on the path of clk_src_pbus is a dynamic integer clock divider which means the clock divide parameter can be changed at any time. This dynamic clock divider has two control signals which are both from the programmable registers. The first one is divide parameter register 0x20201000[19:16] which is named as clk_div_pbus_para_m1. The frequency clk_src_pbus is 1/(clk_div_pbus_para_m1+1) of the frequency of clk_src_hbus. The other is the parameter update control register 0x20201038[0] which is named as clk_div_pbus_para_up. Writing 1 to this register will update the divide parameter to the clock divider and then the frequency of the output clock will be changed according to clk_div_pbus_para_m1.

The programming flow for clk_src_pbus is as below:

- Program the registers for clk_src_hbus as described in chapter 7.3
- Set the divide parameter register 0x20201000[19:16]
- Writing 1 to 0x20201038[0] to update the divide parameter to the clock divider.

7.5 Control for clk_src_ble_mac

Clk_src_ble_mac is the clock of the BLE MAC. It comes from the clk_src_hbus.



Clk_src_ble_mac is the same as clk_src_hbus or the divide clock of clk_src_hbus. The clock gate on the path of clk_src_ble_mac is controlled by register 0x20201004[13:12] which is named as clkg_ble_mac_div_clk and clkg_ble_mac_div_set. Writing 1 to 0x20201004[13] will close the clock gate(no clock will be output). Writing 1 to 0x20201004[12] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

The clock divider "/n4(D)" on the path of clk_src_ble_mac is a dynamic integer clock divider which means the clock divide parameter can be changed at any time. This dynamic clock divider has two control signals which are both from the programmable registers. The first one is divide parameter register 0x2020103c[11:8] which is named as clk_div_ble_mac_para0_m1. The frequency of clk_src_ble_mac is 1/(clk_div_ble_mac_para0_m1+1) of the frequency of clk_src_hbus. The other is the parameter update control register 0x20201038[1] which is named as clk_div_ble_mac_para_up. Writing 1 to this register will update the divide parameter to the clock divider and then the frequency of the output clock will be changed according to clk_div_ble_mac_para0_m1.

The clock mux "m" before clk_src_ble_mac is controlled by programmable register 0x2020103c[0]. When the register is 0, clk_src_hbus is selected. When the register is 1, the divide clock is selected. The clock mux is not glitch free, so before change the select control register, the clock gate must be closed to avoid glitch on the output clock.

The programming flow for clk_src_ble_mac is as below:

- Program the registers for clk_src_hbus as described in chapter 7.3

- Writing 1 to 0x20201004[13] to close the clock gate
- Writing to 0x2020103c[0] to select expected clock
- Writing 1 to 0x20201004[12] to open the clock gate
- If needed, writing to 0x2020103c[11:8] to set the output clock frequency
- If needed, Writing 1 to this register to update the divide parameter to the clock divider

Note that the frequency of clk_src_hbus is a multiple of 16MHz and equal to or less than 96MHz. The frequency of clk_src_ble_mac must be a multiple of 1MHz and equal to or larger than 8MHz and equal to or less than 32MHz. The configuration for the clk_src_ble_mac must satisfy those rules. There is a register named ble_freq0 in the system controller which address is 0x10232040[13:8] to indicate the frequency of clk_src_ble_mac in MHz. After programming clk_src_ble_mac, the frequency of clk_src_ble_mac must be written into this register correctly. The configuration for clk_src_ble_mac must be finished before the BLE MAC starts to work.

Anti-harmonics logic is implemented in BX2400. To avoid the noise from the working clock of the digital module on particularly frequency, BX2400 adjust the frequency of the clock of BLE MAC and BLE modem. For example, if current BLE RX channel is 2432MHz, then if the anti-harmonics logic is enabled, BX2400 will adjust the frequency of clock of BLE MAC from 32MHz to 24MHz to avoid noise to the RF module. The adjustment on the BLE MAC clock is to control the clock divide parameter of the dynamic integer clock divider. Related control registers is listed below:

- Register 0x2020103c[11:8] which is named as clk_div_ble_mac_para0_m1 defines the first clock divide parameter.
- Register 0x2020103c[15:12] which is named as clk_div_ble_mac_para1_m1 defines the first clock divide parameter.
- Register 0x2020103c[16] which is named as clk_sel_ble_mac_by_hw defines the control source of the dynamic integer clock divider. If this register is low, then the clock divider is controlled by the software(by register clk_div_ble_mac_para0_m1 and clk_div_ble_mac_para_up). If this register is high, then the clock divider is controlled by the anti-harmonics logic
- Register 0x20110028[31:0] and register 0x2011002c[7:0] which are altogether 40 bits define the BLE frequency channels which are enabled with anti-harmonics logic and which are not enabled with anti-harmonics logic. Each bit of these two registers represents a BLE frequency channel. If the bit is high, then when the BLE MAC is sending or receiving data on the frequency channel, the frequency of the clock of BLE MAC is defined by 0x2020103c[15:12]. If the bit is low, then when the BLE MAC is sending or receiving data on the frequency channel, the frequency of the clock of BLE MAC is defined by 0x2020103c[11:8].

The programming flow for anti-harmonics logic is as below:

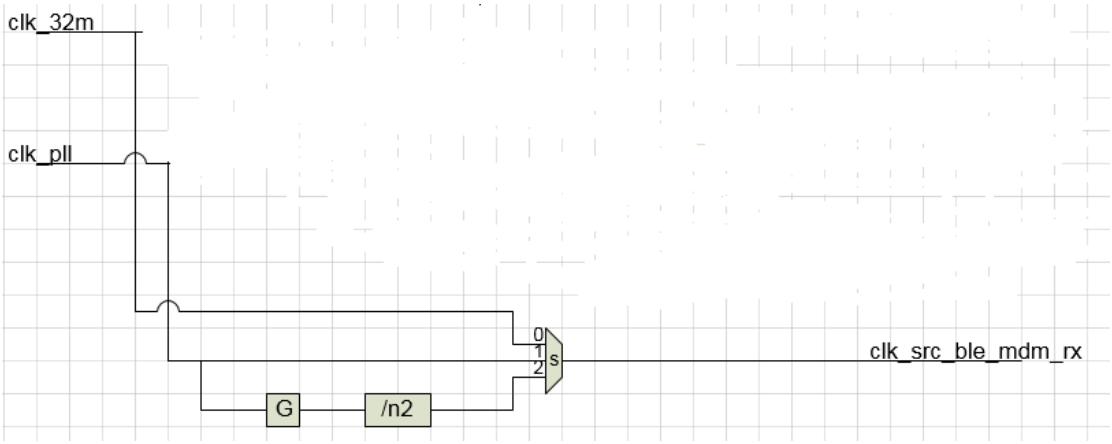
- Config clk_src_hbus

- Writing 1 to 0x20201004[13] to close the clock gate
- Writing 1 to 0x2020103c[0] to select clock divider
- Writing 1 to 0x20201004[12] to open the clock gate
- Writing 1 to 0x2020103c[16] to enable anti-harmonics logic
- Writing predefined clock divide parameter to 0x2020103c[11:8] and 0x2020103c[15:12]
- Writing predefined channel select value to 0x20110028[31:0] and 0x2011002c[7:0] to define the channels on which the BLE MAC works on the clock defined by 0x2020103c[11:8] and on which the BLE MAC works on the clock defined by 0x2020103c[15:12].

Note that because the frequency of the clock of BLE MAC changes dynamically between two predefined values, so there must be two the frequency number indication registers. The first register is 0x10232040[13:8] which is named as ble_freq0 and indicates the frequency number defined by 0x2020103c[11:8] in MHz. The second register is 0x10232040[21:16] which is named as ble_freq1 and indicates the frequency number defined by 0x2020103c[15:12] in MHz.

7.6 Control for clk_src_ble_mdm_rx

Clk_src_ble_mdm_rx is the receiving clock for the BLE modem. It comes from clk_32m and clk_pll.



The clock gate is controlled by register 0x20201004[15:14] which is named as clkg_ble_mdm_rx_div_clr and clkg_ble_mdm_rx_div_set. Writing 1 to 0x20201004[15] will close the clock gate(no clock will be output). Writing 1 to 0x20201004[14] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

The clock divider "/n2" is a static integer clock divider which is controlled by register 0x2020103c[6:5] which is named as clk_div_ble_mdm_rx_para_m1. It is a static clock divider, so before change the clock divide parameter, the clock gate before the

divider must be closed. The frequency of the output clock of the clk_32m clock divider is 1/(clk_div_ble_mdm_rx_para_m1+1) of the clk_32m.

The glitch free clock switch “s” is controlled by the programmable register 0x2020103c[3:1] which is named as clk_sel_ble_mdm_rx_sw. This register is a one hot register which means at any time only one bit of this register can be high. When the register is 001, clk_32m is selected. When the register is 010, the clk_pll is selected. When the register is 100, the divider clock is selected. Because the clock switch is glitch free, so the control register can be programmed at any time. But when the control register is programmed, the two clocks which are selected and unseleted must be active.

The programming flow for clk_src_ble_mdm_rx is as below:

- Writing 1 to 0x20201004[15] to close the clock gate
- Writing to 0x2020103c[6:5] to set the output clock frequency
- Writing 1 to 0x20201004[14] to open the clock gate
- Writing to 0x2020103c[3:1] to select the expected clock

Just like clk_src_ble_mac, there is anti-harmonics logic implemented for clk_src_ble_mdm_rx too. The related control registers is listed below:

- Register 0x2020103c[4] which is named as clk_sel_ble_mdm_rx_by_hw defines the control source of the glitch free clock switch. If this register is low, then the clock switch is controlled by the software(by register clk_sel_ble_mdm_rx_sw). If this register is high, then the clock switch is controlled by the anti-harmonics logic.
- Register 0x20110020[10:8] is the first predefined clock select configuration. It must be set to 001.
- Register 0x20110020[13:11] is the second predefined clock select configuration. It can be set to 010 or 100. It depends the frequency of clk_pll. If frequency of clk_pll is 48MHz, then it must be set to 010 to select clk_pll. If the frequency of clk_pll is 96MHz, then it must be set to 100 to select the divider clock and the clock divide parameter must be set to 1 to make the output clock frequency to be 48MHz.
- Register 0x2011001c[31:0] and register 0x20110020[7:0] which are altogether 40 bits define the BLE frequency channels which are enabled with anti-harmonics logic and which are not enabled with anti-harmonics logic. Each bit of these two registers represents a BLE frequency channel. If the bit is high, then when the BLE modem is receiving data on the frequency channel, the frequency of the clock of BLE modem is selected by 0x20110020[13:11]. If the bit is low, then when the BLE modem is receiving data on the frequency channel, the frequency of the clock of BLE modem is selected by 0x20110020[10:8].

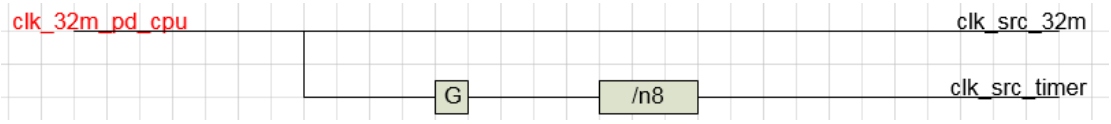
The programming flow for anti-harmonics logic is as below:

- Writing 1 to 0x20201004[15] to close the clock gate
- Writing to 0x2020103c[6:5] to set the output clock frequency

- Writing 1 to 0x20201004[14] to open the clock gate
- Writing 0x20110020[13:11] and 0x20110020[10:8] to set the predefined clock configuration.
- Writing 1 to 0x2020103c[4] to enable anti-harmonics logic

7.7 Control for clk_src_timer

CLk_src_timer is the clock for the counter of the system timer. It is from clk_32m.



The clock gate is controlled by register 0x20132010[7:6] which is named as clkg_clr_timer_div and clkg_set_timer_div. Writing 1 to 0x20132010[7] will close the clock gate(no clock will be output). Writing 1 to 0x20132010[6] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

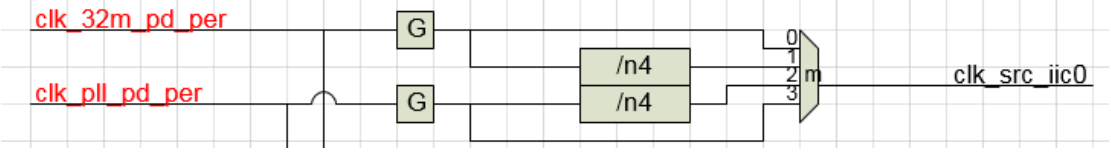
The clock divider "/n8" is a static integer clock divider which is controlled by register 0x20201008[23:16] which is named as clk_div_timer_para_m1. It is a static clock divider, so before change the clock divide parameter, the clock gate before the divider must be closed. The frequency of the output clock of the clk_32m clock divider is 1/(clk_div_timer_para_m1+1) of the clk_32m.

The programming flow for clk_src_timer is as below:

- Writing 1 to 0x20132010[7] to close the clock gate
- Writing to 0x20201008[23:16] to set the output clock frequency
- Writing 1 to 0x20132010[6] to open the clock gate

7.8 Control for clk_src_iic0

Clk_src_iic0 is the clock for IIC0. It is from clk_32m and clk_pll.



The clock gate on the path of clk_32m is controlled by register 0x20149010[1:0] which is named as clkg0_clr_iic0 and clkg0_set_iic0. Writing 1 to 0x20149010[1] will close the clock gate(no clock will be output). Writing 1 to 0x20149010[0] will open

the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

The clock gate on the path of clk_pll is controlled by register 0x20149010[3:2] which is named as clkg1_clr_iic0 and clkg1_set_iic0. Writing 1 to 0x20149010[3] will close the clock gate(no clock will be output). Writing 1 to 0x20149010[2] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

The two clock dividers "/n4" are both static integer clock dividers which are controlled by the same register 0x20149004[3:0] which is named as clk_div_iic0_para_m1. They are both static clock dividers, so before change the clock divide parameter, the clock gates before the dividers must be closed. The frequency of the output clock of the clk_32m clock divider is 1/(clk_div_iic0_para_m1+1) of the clk_32m. The frequency of the output clock of the clk_pll clock divider is 1/(clk_div_iic0_para_m1+1) of the clk_pll.

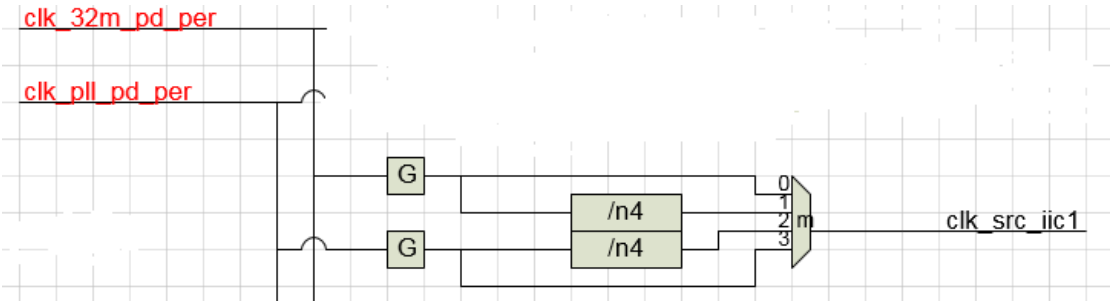
The clock mux "m" above is controlled by programmable register 0x20149000[1:0]. When the register is 0, clk_32m is selected. When the register is 1, the divide clock from clk_32m is selected. When the register is 2, clk_pll is selected. When the register is 3, the divide clock from clk_pll is selected. The clock mux is not glitch free, so before change the select control register, the clock gate before the clock divider must be closed to avoid glitch on the output clock.

The programming flow for clk_src_iic0 is as below:

- Writing 1 to 0x20132010[1] and 0x20132010[3] to close the clock gate
- Writing to 0x20149004[3:0] to set the output clock frequency
- Writing to 0x20149000[1:0] to select the expected clcok
- Writing 1 to 0x20132010[1] or 0x20132010[3] to open the clock gate

7.9 Control for clk_src_iic1

Clk_src_iic1 is the clock for IIC1. It is from clk_32m and clk_pll.



The clock gate on the path of clk_32m is controlled by register 0x20149010[5:4] which is named as clkg0_clr_iic1 and clkg0_set_iic1. Writing 1 to 0x20149010[5] will close the clock gate(no clock will be output). Writing 1 to 0x20149010[4] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

The clock gate on the path of clk_pll is controlled by register 0x20149010[7:6] which is named as clkg1_clr_iic1 and clkg1_set_iic1. Writing 1 to 0x20149010[7] will close the clock gate(no clock will be output). Writing 1 to 0x20149010[6] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

The two clock dividers "/n4" are both static integer clock dividers which are controlled by the same register 0x20149004[7:4] which is named as clk_div_iic1_para_m1. They are both static clock dividers, so before change the clock divide parameter, the clock gates before the dividers must be closed. The frequency of the output clock of the clk_32m clock divider is 1/(clk_div_iic1_para_m1+1) of the clk_32m. The frequency of the output clock of the clk_pll clock divider is 1/(clk_div_iic1_para_m1+1) of the clk_pll.

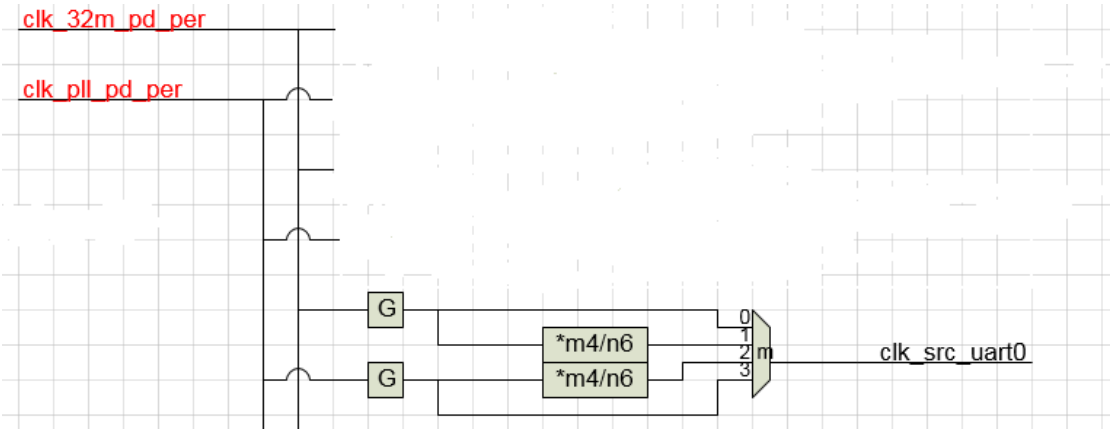
The clock mux "m" above is controlled by programmable register 0x20149000[5:4]. When the register is 0, clk_32m is selected. When the register is 1, the divide clock from clk_32m is selected. When the register is 2, clk_pll is selected. When the register is 3, the divide clock from clk_pll is selected. The clock mux is not glitch free, so before change the select control register, the clock gate before the clock divider must be closed to avoid glitch on the output clock.

The programming flow for clk_src_iic1 logic is as below:

- Writing 1 to 0x20132010[5] and 0x20132010[7] to close the clock gate
- Writing to 0x20149004[7:4] to set the output clock frequency
- Writing to 0x20149000[5:4] to select the expected clcok
- Writing 1 to 0x20132010[4] or 0x20132010[6] to open the clock gate

7.10Control for clk_src_uart0

Clk_src_uart0 is the clock for UART0. It is from clk_32m and clk_pll.



The clock gate on the path of clk_32m is controlled by register 0x20149010[9:8] which is named as clkg0_clr_uart0 and clkg0_set_uart0. Writing 1 to 0x20149010[9] will close the clock gate(no clock will be output). Writing 1 to 0x20149010[8] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

The clock gate on the path of clk_pll is controlled by register 0x20149010[11:10] which is named as clkg1_clr_uart0 and clkg1_set_uart0. Writing 1 to 0x20149010[11] will close the clock gate(no clock will be output). Writing 1 to 0x20149010[10] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

The two clock dividers “*m4/n6” are both static integer clock dividers which are

controlled by the same register 0x20149008[5:0]，0x20149008[13:8]，

0x20149008[19:16]，0x20149008[27:24] which are named as

clk_div_uart0_para_len0_m1，clk_div_uart0_para_len1_m1，

clk_div_uart0_para_num0_m1 and clk_div_uart0_para_num1_m1. They are both static clock dividers, so before change the clock divide parameter, the clock gates before the dividers must be closed. The output clock is comprised of a circulation of (num0+1) clock cycle with (len0+1) divide parameter followed by (num1+1) clock cycle with (len1+1) divide parameter. The output clock average frequency is (num0+1+num1+1)/((num0+1)*(len0+1)+(num1+1)*(len1+1)) of the input clock frequency.

The clock mux “m” above is controlled by programmable register 0x20149000[9:8]. When the register is 0, clk_32m is selected. When the register is 1, the divide clock from clk_32m is selected. When the register is 2, clk_pll is selected. When the register is 3, the divide clock from clk_pll is selected. The clock mux is not glitch free,

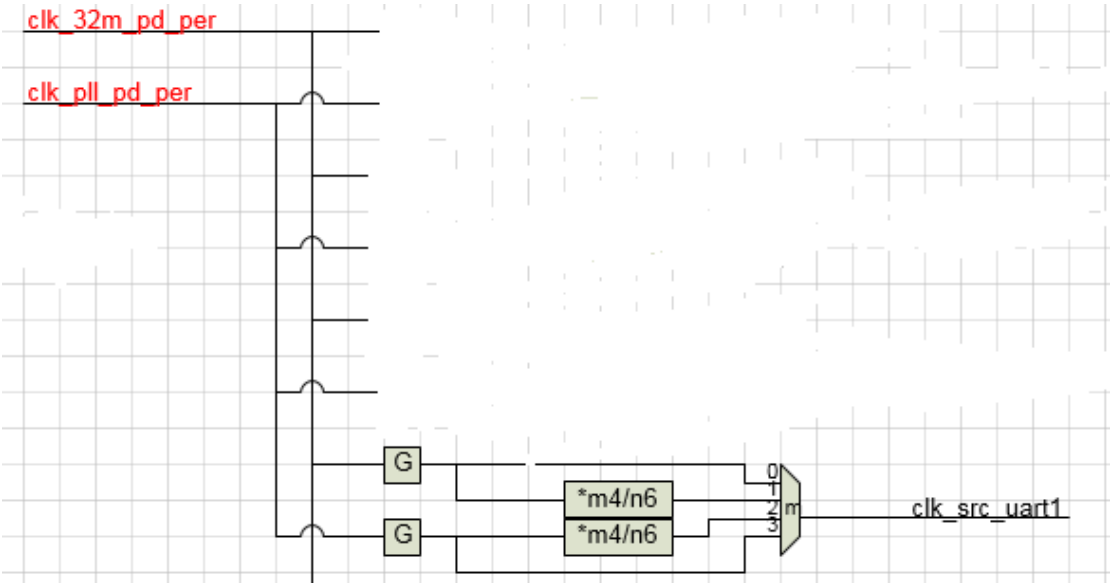
so before change the select control register, the clock gate before the clock divider must be closed to avoid glitch on the output clock.

The programming flow for clk_src_uart0 is as below:

- Writing 1 to 0x20132010[9] and 0x20132010[11] to close the clock gate
- Writing to 0x20149008[5:0]，0x20149008[13:8]，0x20149008[19:16]，0x20149008[27:24] to set the output clock frequency
- Writing to 0x20149000[9:8] to select the expected clcok
- Writing 1 to 0x20132010[8] or 0x20132010[10] to open the clock gate

7.11Control for clk_src_uart1

Clk_src_uart1 is the clock for UART1. It is from clk_32m and clk_pll.



The clock gate on the path of clk_32m is controlled by register 0x20149010[13:12] which is named as clkg0_clr_uart1 and clkg0_set_uart1. Writing 1 to 0x20149010[13] will close the clock gate(no clock will be output). Writing 1 to 0x20149010[12] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

The clock gate on the path of clk_pll is controlled by register 0x20149010[15:14] which is named as clkg1_clr_uart1 and clkg1_set_uart1. Writing 1 to 0x20149010[15] will close the clock gate(no clock will be output). Writing 1 to 0x20149010[14] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

The two clock dividers “*m4/n6” are both static integer clock dividers which are

controlled by the same register 0x2014900c[5:0] , 0x2014900c[13:8] ,

0x2014900c[19:16] , 0x2014900c[27:24] which are named as

clk_div_uart1_para_len0_m1 , clk_div_uart1_para_len1_m1 ,

clk_div_uart1_para_num0_m1 and clk_div_uart1_para_num1_m1. They are both static clock dividers, so before change the clock divide parameter, the clock gates before the dividers must be closed. The output clock is comprised of a circulation of (num0+1) clock cycle with (len0+1) divide parameter followed by (num1+1) clock cycle with (len1+1) divide parameter. The output clock average frequency is (num0+1+num1+1)/((num0+1)*(len0+1)+(num1+1)*(len1+1)) of the input clock frequency.

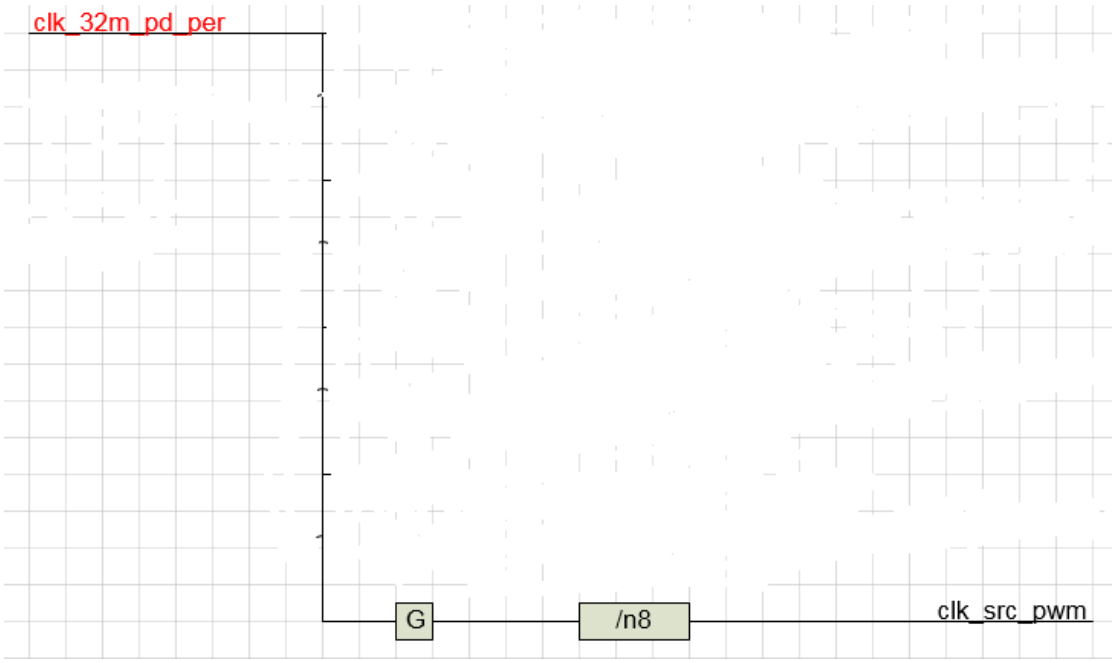
The clock mux “m” above is controlled by programmable register 0x20149000[13:12]. When the register is 0, clk_32m is selected. When the register is 1, the divide clock from clk_32m is selected. When the register is 2, clk_pll is selected. When the register is 3, the divide clock from clk_pll is selected. The clock mux is not glitch free, so before change the select control register, the clock gate before the clock divider must be closed to avoid glitch on the output clock.

The programming flow for clk_src_uart1 is as below:

- Writing 1 to 0x20132010[13] and 0x20132010[15] to close the clock gate
- Writing to 0x2014900c[5:0]，0x2014900c[13:8]，0x2014900c[19:16]，0x2014900c[27:24] to set the output clock frequency
- Writing to 0x20149000[13:12] to select the expected clcok
- Writing 1 to 0x20132010[12] or 0x20132010[14] to open the clock gate

7.12Control for clk_src_pwm

Clk_src_pwm is the clock of the counter of the PWM. It is from clk_32m.



The clock gate is controlled by register 0x20149014[1:0] which is named as clkg_clr_pwm_div and clkg_set_pwm_div. Writing 1 to 0x20149014[1] will close the clock gate(no clock will be output). Writing 1 to 0x20149014[0] will open the clock gate(clock will be output). If the clock divider is not needed by the current system, the clock gate can be closed to save the power.

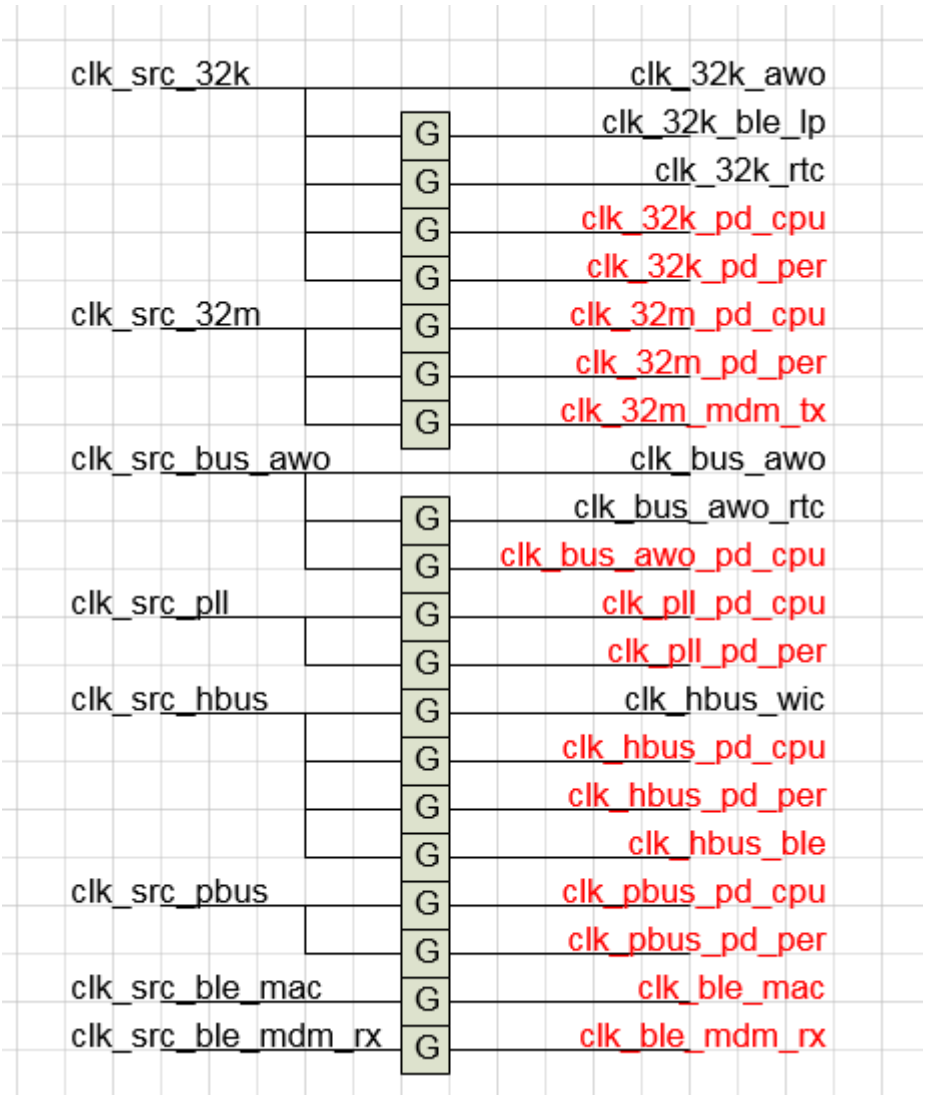
The clock divider "/n8" is a static integer clock divider which is controlled by register 0x20201004[31:24] which is named as clk_div_pwm_para_m1. It is a static clock divider, so before change the clock divide parameter, the clock gate before the divider must be closed. The frequency of the output clock of the clk_32m clock divider is 1/(clk_div_pwm_para_m1+1) of the clk_32m.

The programming flow for clk_src_pwm is as below:

- Writing 1 to 0x20149014[1] to close the clock gate
- Writing to 0x20201004[31:24] to set the output clock frequency
- Writing 1 to 0x20149014[0] to open the clock gate

7.13Control for clock gate(AWO power domain)

The clocks for the modules in AWO power domain and out of AWO power domain is shown below. Some of these clocks have clock gates and some don't have. The control signals of these clock gates comes from the programmable control register or from the PMU.



The table below shows the control signal for all of the clocks above.

name	souce	control reg	PMU signal
clk_32k_awo	clk_src_32k		
clk_32k_ble_lp	clk_src_32k	clkg_ble_lp	
clk_32k_rtc	clk_src_32k	clkg_rtc	
clk_32k_pd_cpu	clk_src_32k		clkg_pd_cpu
clk_32k_pd_per	clk_src_32k		clkg_pd_per
clk_32m_pd_cpu	clk_src_32m		clkg_pd_cpu
clk_32m_pd_per	clk_src_32m		clkg_pd_per
clk_32m_mdm_tx	clk_src_32m	clkg_ble	clkg_pd_ble
clk_bus_awo	clk_src_bus_awo		
clk_bus_awo_rtc	clk_src_bus_awo	clkg_rtc	
clk_bus_awo_pd_cpu	clk_src_bus_awo		clkg_pd_cpu
clk_pll_pd_cpu	clk_src_pll		clkg_pd_cpu
clk_pll_pd_per	clk_src_pll		clkg_pd_per

name	souce	control reg	PMU signal
clk_hbus_wic	clk_src_hbus	clkg_wic	
clk_hbus_pd_cpu	clk_src_hbus		clkg_pd_cpu
clk_hbus_pd_per	clk_src_hbus		clkg_pd_per
clk_hbus_ble	clk_src_hbus	clkg_ble	clkg_pd_ble
clk_pbus_pd_cpu	clk_src_pbus		clkg_pd_cpu
clk_pbus_pd_per	clk_src_pbus		clkg_pd_per
clk_ble_mac	clk_src_ble_mac	clkg_ble	clkg_pd_ble
clk_ble_mdm_rx	clk_src_ble_mdm_rx	clkg_ble	clkg_pd_ble

Signal clkg_pd_cpu comes from the system PMU. When the CPU power domain is powered off, this signal will be set low to close the related clock gates. When the CPU power domain is powered on, this signal will be set high to open the related clock gates.

Signal clkg_pd_per comes from the system PMU. When the PER power domain is powered off, this signal will be set low to close the related clock gates. When the PER power domain is powered on, this signal will be set high to open the related clock gates.

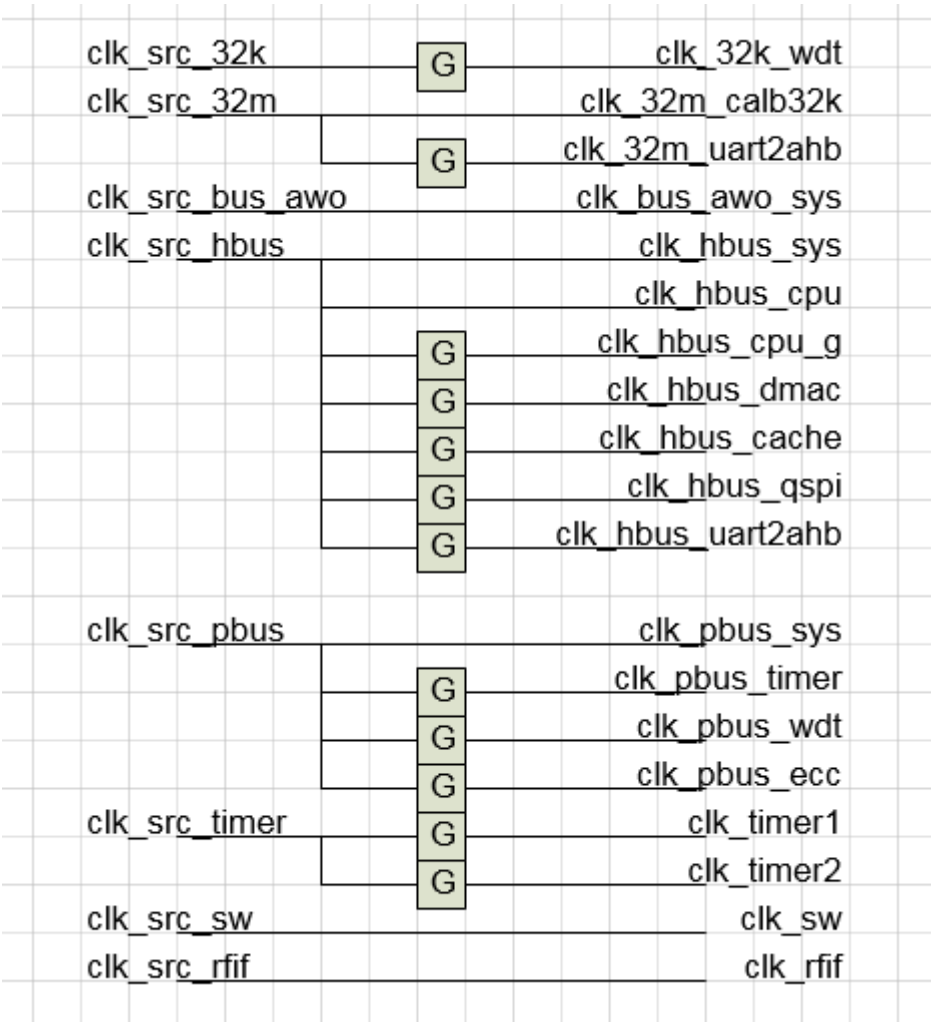
Signal clkg_pd_ble comes from the system PMU. When the BLE power domain is powered off, this signal will be set low to close the related clock gates. When the BLE power domain is powered on, this signal will be set high to open the related clock gates.

All of the other control signals are from the programmable registers. Each of the control signals represents a two bits control register in the system controller of AWO power domain which are named as clkg_*_clr and clkg_*_set. Writing 1 to clkg_*_clr, the control signal will be set low and the clock gate is closed, no clock will be output. Writing 1 to clkg_*_set, the control signal will be set high and the clock gate is opened, output clock is the same as the input clock. The control registers are listed below:

0x20201004 [11]	clkg_ble_clr
0x20201004 [10]	clkg_ble_set
0x20201004 [9]	clkg_wic_clr
0x20201004 [8]	clkg_wic_set
0x20201004 [3]	clkg_rtc_clr
0x20201004 [2]	clkg_rtc_set
0x20201004 [1]	clkg_ble_lp_clr
0x20201004 [0]	clkg_ble_lp_set

7.14Control for clock gate(CPU power domain)

The clocks for the modules in CPU power domain is shown below. Some of these clocks have clock gates and some don't have. The control signals of these clock gates comes from the programmable control registers.



The table below shows the control signal for all of the clocks above.

name	souce	control reg
clk_32k_wdt	clk_src_32k	clkg_wdt
clk_32m_calb32k	clk_src_32m	
clk_32m_uart2ahb	clk_src_32m	clkg_uart2ahb
clk_bus_awo	clk_src_bus_awo	
clk_hbus_sys	clk_src_hbus	
clk_hbus_cpu	clk_src_hbus	
clk_hbus_cpu_g	clk_src_hbus	hclk_gate
clk_hbus_dma	clk_src_hbus	clkg_dma
clk_hbus_cache	clk_src_hbus	clkg_cache

name	souce	control reg
clk_hbus_qspi	clk_src_hbus	clkg_qspi
clk_hbus_uart2ahb	clk_src_hbus	clkg_uart2ahb
clk_pbus_sys	clk_src_pbus	
clk_pbus_timer	clk_src_pbus	clkg_timer1 ，clkg_timer2
clk_pbus_wdt	clk_src_pbus	clkg_wdt
clk_pbus_ecc	clk_src_pbus	clkg_ecc
clk_timer1	clk_src_timer	clkg_timer1
clk_timer2	clk_src_timer	clkg_timer2
clk_sw	clk_src_sw	
clk_rfif	clk_src_rfif	

Only the control signal named hclk_gate for clock clk_hbus_cpu_g does not comes from the programmable register. It comes from the FSM inside the CPU. When CPU enters sleep mode, this signal will be set low to gate the clock for CPU to save power. When CPU wakes up from sleep mode, this signal will be set high.

There are two control signals for clock clk_pbus_timer. When both of these two control signals are low, the clock is gated. Any of these two control signals is high, the clock is opened.

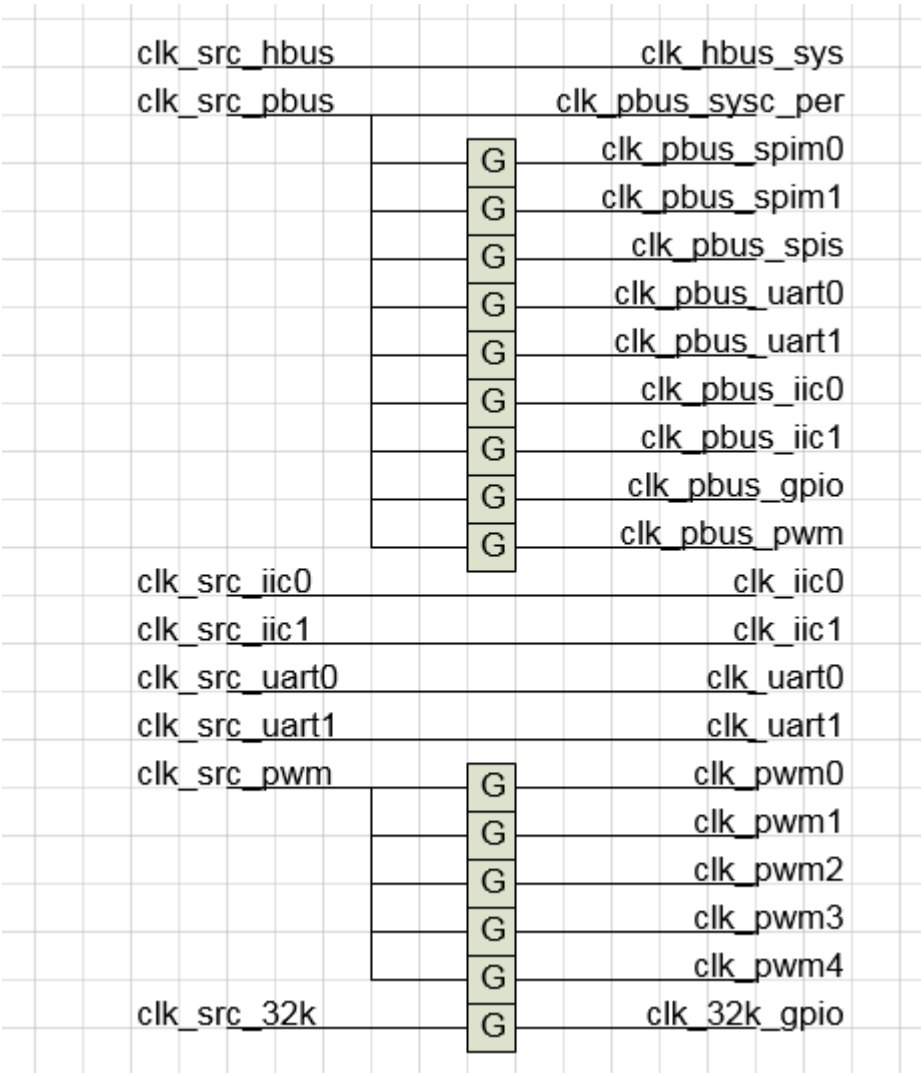
All of the other control signals are from the programmable registers. Each of the control signals represents a two bits control register in the system controller of CPU power domain which are named as clkg_clr_* and clkg_set_*. Writing 1 to clkg_clr_*, the control signal will be set low and the clock gate is closed, no clock will be output. Writing 1 to clkg_set_*, the control signal will be set high and the clock gate is opened, output clock is the same as the input clock. The control registers are listed below:

0x20132014[19]	clkg_clr_ecc
0x20132014 [18]	clkg_set_ecc
0x20132014 [17]	clkg_clr_uart2ahb
0x20132014 [16]	clkg_set_uart2ahb
0x20132014 [15]	clkg_clr_timer1
0x20132014 [14]	clkg_set_timer1
0x20132014 [13]	clkg_clr_timer0
0x20132014 [12]	clkg_set_timer0
0x20132014 [11]	clkg_clr_qspi
0x20132014 [10]	clkg_set_qspi
0x20132014 [9]	clkg_clr_cache
0x20132014 [8]	clkg_set_cache
0x20132014 [7]	clkg_clr_dmac

0x20132014 [6]	clkg_set_dmac
0x20132014 [5]	clkg_clr_wdt
0x20132014 [4]	clkg_set_wdt

7.15Control for clock gate(PER power domain)

The clocks for the modules in PER power domain is shown below. Some of these clocks have clock gates and some don't have. The control signals of these clock gates comes from the programmable control registers.



The table below shows the control signal for all of the clocks above.

name	souce	control reg
clk_hbus_sys	clk_src_hbus	
clk_pbus_sysc_per	clk_src_pbus	
clk_pbus_spim0	clk_src_pbus	clkg0_spim0

name	souce	control reg
clk_pbus_spim1	clk_src_pbus	clkg0_spim1
clk_pbus_spis	clk_src_pbus	clkg0_spis
clk_pbus_uart0	clk_src_pbus	clkg0_uart0 ，clkg1_uart0
clk_pbus_uart1	clk_src_pbus	clkg0_uart1, clkg1_uart1
clk_pbus_iic0	clk_src_pbus	clkg0_iic0, clkg1_iic0
clk_pbus_iic1	clk_src_pbus	clkg0_iic1, clkg1_iic1
clk_pbus_gpio	clk_src_pbus	clkg_gpio
clk_pbus_pwm	clk_src_pbus	clkg_pwm0~4
clk_iic0	clk_src_iic0	clkg0_iic0, clkg1_iic0
clk_iic1	clk_src_iic1	clkg0_iic1, clkg1_iic1
clk_uart0	clk_src_uart0	clkg0_uart0 ，clkg1_uart0
clk_uart1	clk_src_uart1	clkg0_uart1, clkg1_uart1
clk_pwm0	clk_src_pwm	clkg_pwm0
clk_pwm1	clk_src_pwm	clkg_pwm1
clk_pwm2	clk_src_pwm	clkg_pwm2
clk_pwm3	clk_src_pwm	clkg_pwm3
clk_pwm4	clk_src_pwm	clkg_pwm4
clk_32k_gpio	clk_src_32k	clkg_gpio

The clock gate control signal for clock clk_pbus_uart0 is the same as the clock control signal for clock clk_src_uart0 which is described in chapter7.10. When both of these two control signals are low, the clock is gated. Any of these two control signals is high, the clock is opened.

The clock gate control signal for clock clk_pbus_uart1 is the same as the clock control signal for clock clk_src_uart1 which is described in chapter7.11. When both of these two control signals are low, the clock is gated. Any of these two control signals is high, the clock is opened.

The clock gate control signal for clock clk_pbus_iic0 is the same as the clock control signal for clock clk_src_iic0 which is described in chapter7.8. When both of these two control signals are low, the clock is gated. Any of these two control signals is high, the clock is opened.

The clock gate control signal for clock clk_pbus_iic1 is the same as the clock control signal for clock clk_src_iic1 which is described in chapter7.9. When both of these two control signals are low, the clock is gated. Any of these two control signals is high, the clock is opened.

The clock gate of clk_pbus_pwm is control by 5 signals which are

clkg_pwm0~clkg_pwm4. Any of these control signals is high, the clock is opened. All of these control signals are 0, the clock gate is closed.

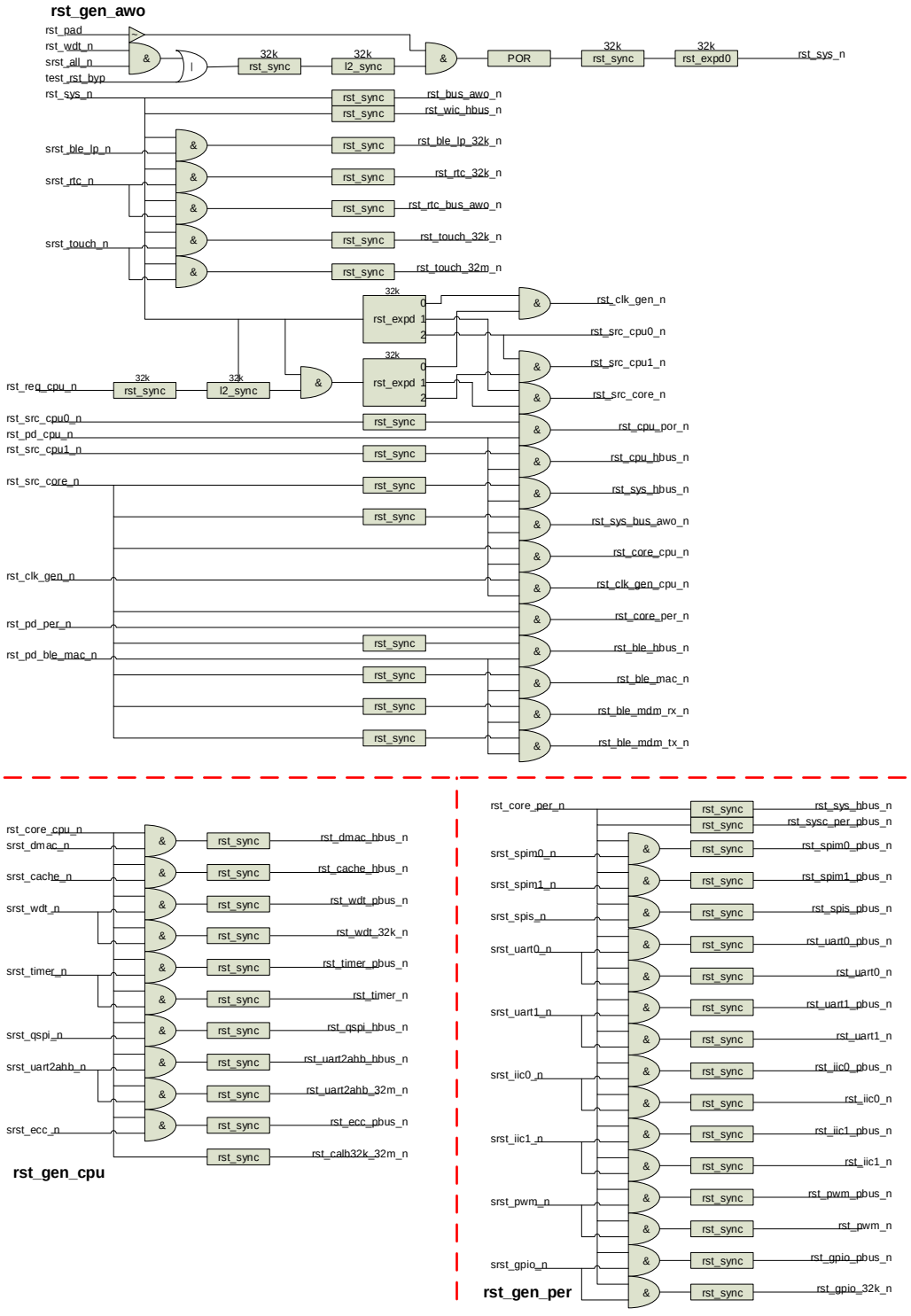
Each of the control signals represents a two bits control register in the system controller of CPU power domain which are named as clkg_clr_* and clkg_set_*. Writing 1 to clkg_clr_*, the control signal will be set low and the clock gate is closed, no clock will be output. Writing 1 to clkg_set_*, the control signal will be set high and the clock gate is opened, output clock is the same as the input clock. The control registers are listed below:

0x20149010[25]	clkg0_clr_spis
0x20149010 [24]	clkg0_set_spis
0x20149010 [21]	clkg0_clr_spim1
0x20149010 [20]	clkg0_set_spim1
0x20149010 [17]	clkg0_clr_spim0
0x20149010 [16]	clkg0_set_spim0
0x20149010 [15]	clkg1_clr_uart1
0x20149010 [14]	clkg1_set_uart1
0x20149010 [13]	clkg0_clr_uart1
0x20149010 [12]	clkg0_set_uart1
0x20149010 [11]	clkg1_clr_uart0
0x20149010 [10]	clkg1_set_uart0
0x20149010 [9]	clkg0_clr_uart0
0x20149010 [8]	clkg0_set_uart0
0x20149010 [7]	clkg1_clr_iic1
0x20149010 [6]	clkg1_set_iic1
0x20149010 [5]	clkg0_clr_iic1
0x20149010 [4]	clkg0_set_iic1
0x20149010 [3]	clkg1_clr_iic0
0x20149010 [2]	clkg1_set_iic0
0x20149010 [1]	clkg0_clr_iic0
0x20149010 [0]	clkg0_set_iic0
0x20149014 [13]	clkg_clr_gpio
0x20149014 [12]	clkg_set_gpio
0x20149014 [11]	clkg_clr_pwm4
0x20149014 [10]	clkg_set_pwm4
0x20149014 [9]	clkg_clr_pwm3
0x20149014 [8]	clkg_set_pwm3
0x20149014 [7]	clkg_clr_pwm2
0x20149014 [6]	clkg_set_pwm2
0x20149014 [5]	clkg_clr_pwm1

0x20149014 [4]	clkg_set_pwm1
0x20149014 [3]	clkg_clr_pwm0
0x20149014 [2]	clkg_set_pwm0

8 Reset system

8.1 General description



The reset block diagram of BX2400 is shown above.

As the clock system, the reset system is also composed of three modules: rst_gen_awo, rst_gen_per and rst_gen_cpu. Rst_gen_awo is in the AWO power domain and generates reset signals for the modules inside AWO power domain and reset signals for CPU and bus matrix inside the CPU power domain. Rst_gen_cpu is in the CPU power domain and generates reset signals for the modules inside AWO power domain. Rst_gen_per is in the PER power domain and generates reset signals for the modules inside AWO power domain

The block marked with “&” is an AND gate.

The block marked with “|” is an OR gate.

The block marked with “~” is an INVERT gate.

The block marked with “rst_sync” is a reset synchronizer.

The block marked with “l2_sync” is a two level synchronizer.

The block marked with “POR” is the analog module of power on reset. The POR generates power on reset when the system is powered on. The block POR also takes input from the rst_gen_awo. The low pulse on the input signal will also generate a global reset.

The block marked with “rst_expd0” will expand the input low pulse to 8 more cycles of the input clock.

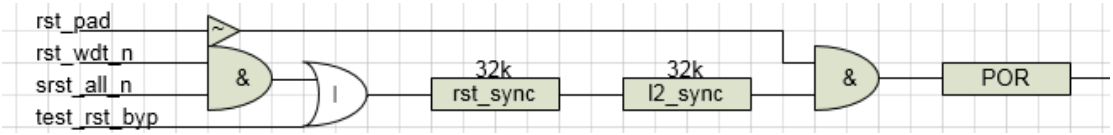
The block marked with “rst_expd” will expand the input reset low pulse and drive the reset output high one by one. The reset output 0 is expended 16 clock cycles according to the input reset signal, and is set to high first. The reset output 1 is set to high 16 clock cycles later than reset output 0. The reset output 2 is set to high 16 clock cycles later than reset output 1. The reset output 0 is used to reset the clock system. The reset output 1 is used to reset all of the other modules except CPU and clock generator. The reset output 2 is used to reset CPU. The reset of CPU is released last to make sure that when CPU starts to work, all of the other modules are not in reset state and ready to work.

In BX2400, except CPU and bus matrix, each of most of the other modules has a dedicated software reset which is from the programmable control register. This software reset is used to reset the module and will not affect the other modules which are working. The signals named as srst_*_n in the above diagram are the software reset signals.

The signals on the right side of the above diagram are the reset signals for the modules.

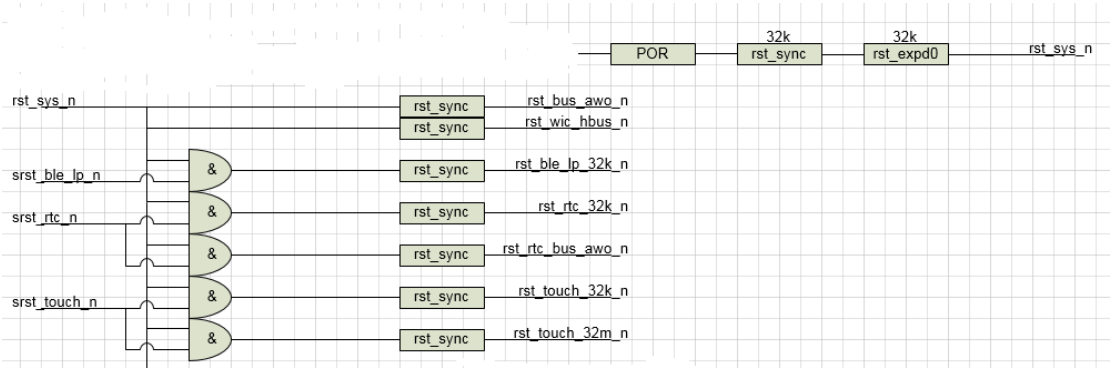
8.2 Reset system of AWO power domain

The logic in the below diagram in rst_gen_awo is used to generate the system reset which will be output to the analog module POR to reset the whole system including the analog part.



In BX2400 there are three sources which will cause system reset which are rst_pad, rst_wdt_n and srst_all_n. rst_pad directly comes from the chip IO named as RST_PAD. This signal is high active. Rst_wdt_n comes from the system watch dog module and is low active. Srst_all_n comes from the system software control register which is low active. Software can set the register low to reset the system. The address of the register is 0x20201040[1] and writing 1 to this register will reset the system.

The logic in the below diagram is used to generate reset signals for the modules inside the AWO power domain.



Some of the reset signals can be pull low by software reset and some signals can not be pulled low. The software reset signal is “AND” with the global reset signal and then synchronized to the clock domain of the target module. The reset signals of modules are shown in the table below:

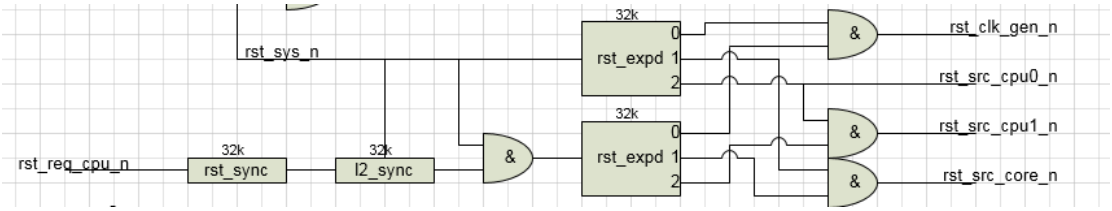
reset name	instant name	soft reset	clock domain
rst_sys_n	clk_gen_awo		clk_32k
rst_bus_awo_n	PMU		clk_bus_awo

rst_wic_hbus_n	WIC		clk_hbus
rst_ble_lp_32k_n	ble_lp	srst_ble_lp_n	clk_32k
rst_rtc_32k_n	RTC	srst_rtc_n	clk_32k
rst_rtc_bus_awo_n	RTC	srst_rtc_n	clk_bus_awo
rst_touch_32k_n	touch	srst_touch_n	clk_32k
rst_touch_32m_n	touch	srst_touch_n	clk_32m

Each of the software reset signals represents a two bits control register in the system controller of AWO power domain which are named as srst_*_n_clr and srst_*_n_set. Writing 1 to srst_*_n_clr, the control signal will be set low and the module is reset. Writing 1 to srst_*_n_set, the control signal will be set high. The control registers are listed below:

0x20201040[9]	srst_touch_n_clr
0x20201040 [8]	srst_touch_n_set
0x20201040 [5]	srst_rtc_n_clr
0x20201040 [4]	srst_rtc_n_set
0x20201040 [3]	srst_ble_lp_n_clr
0x20201040 [2]	srst_ble_lp_n_set

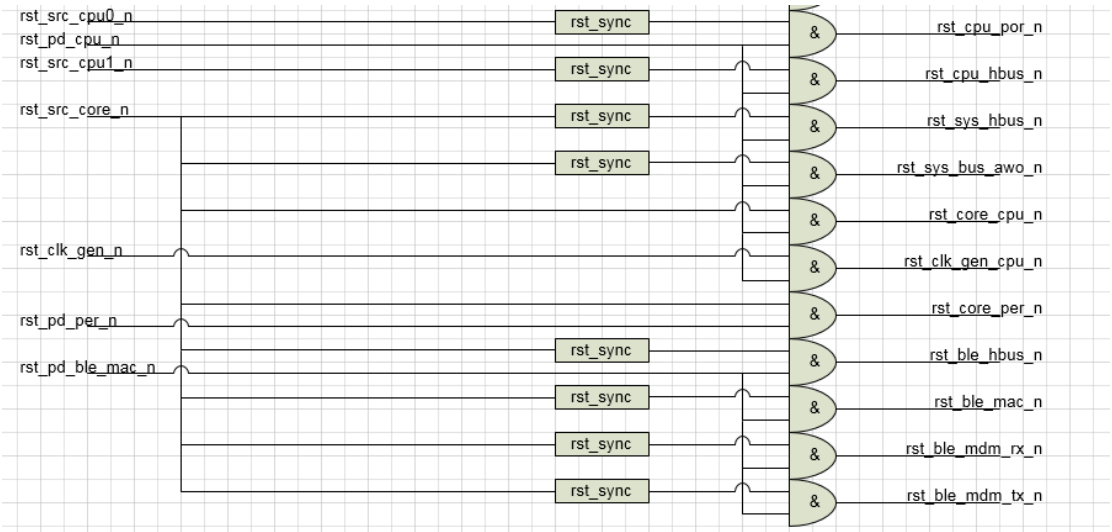
The logic shown in the below diagram is used to generate the global reset for the other power domain.



The reset from POR is expanded to three reset signals which is released one by one. The reset signal released first is for the clock generator module of CPU and PER power domain. The reset signal released second is for all of the other modules except CPU and clock generator of CPU, PER and BLE power domain. The reset signal released last is for CPU to make sure when the CPU starts to work all of the other modules are ready to work.

The reset request from the CPU named as rst_req_cpu_n is also expanded to three reset signals which is released one by one. But these reset signals can not be used to reset the debug related logic in CPU. So there are two reset signals for the CPU named as rst_src_cpu0_n and rst_src_cpu1_n. rst_src_cpu0_n is used to reset the whole CPU and rst_src_cpu1_n is used to reset the other logic in CPU except the debug logic.

The logic shown in the below diagram is used to generate the reset signals for the other power domain.



The reset signal named as rst_pd_cpu_n comes from the system PMU, and is used to reset CPU power domain when the CPU power domain is powered on/off. The reset signal named as rst_pd_per_n comes from the system PMU, and is used to reset PER power domain when the PER power domain is powered on/off. The reset signal named as rst_pd_ble_mac_n comes from the system PMU, and is used to reset BLE power domain when the BLE power domain is powered on/off.

The following table lists the reset relationship:

reset name	**system reset source**	**PMU reset source**	**clock domain**	**instance**
rst_cpu_por	rst_src_cpu0_n	rst_pd_cpu_n	clk_hbus	CPU
rst_cpu_hbus_n	rst_src_cpu1_n	rst_pd_cpu_n	clk_hbus	CPU but not include debug logic
rst_sys_hbus_n	rst_src_core_n	rst_pd_cpu_n	clk_hbus	AHB/APB bus in CPU domain
rst_sys_bus_awo_n	rst_src_core_n	rst_pd_cpu_n	clk_bus_awo	AHB/APB bus in CPU domain
rst_core_cpu_n	rst_src_core_n	rst_pd_cpu_n		other logic in CPU domain
rst_clk_gen_cpu_n	rst_src_core_n	rst_pd_cpu_n		clk_gen_cpu
rst_core_per_n	rst_src_core_n	rst_pd_per_n		PER domain
rst_ble_hbus_n	rst_src_core_n	rst_pd_ble_mac_n	clk_hbus	BLE domain
rst_ble_mac_n	rst_src_core_n	rst_pd_ble_mac_n	clk_ble	BLE domain
rst_ble_mdm_rx_n	rst_src_core_n	rst_pd_ble_mac_n	clk_ble_mdm_rx	BLE modem RX logic
rst_ble_mdm_tx_n	rst_src_core_n	rst_pd_ble_mac_n	clk_32m	BLE modem TX logic

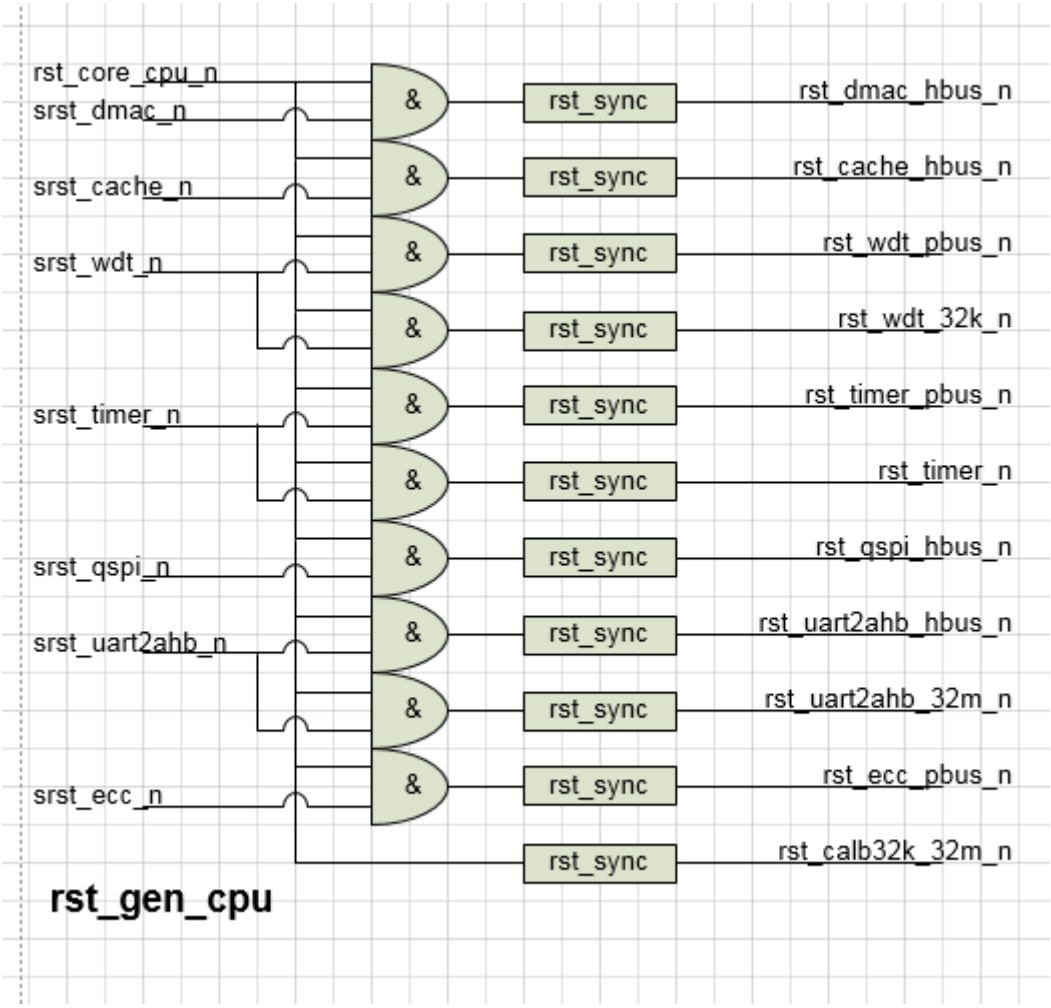
Note that the input signal of block “rst_sync” for BLE related reset comes from rst_src_core_n and srst_ble_n. The two signals are first “AND” together and then input to the “rst_sync” block. The address of control registers for srst_ble_n is :

0x20201040[7]	srst_ble_n_clr

0x20201040[6]	srst_ble_n_set

8.3 Reset system of CPU power domain

The block diagram of the reset system of CPU power domain is shown below:



Rst_gen_cpu implements software logic and reset synchronization logic. The reset relationship is listed below:

reset name	soft reset source	clock domain	instance
rst_dmac_hbus_n	srst_dma_n	clk_hbus	dma
rst_cache_hbus_n	srst_cache_n	clk_hbus	cache
rst_wdt_pbus_n	srst_wdt_n	clk_pbus	wdt
rst_wdt_32k_n	srst_wdt_n	clk_32k	wdt
rst_timer_pbus_n	srst_timer_n	clk_pbus	timer
rst_timer_n	srst_timer_n	clk_timer	timer
rst_qspi_hbus_n	srst_qspi_n	clk_hbus	qspi

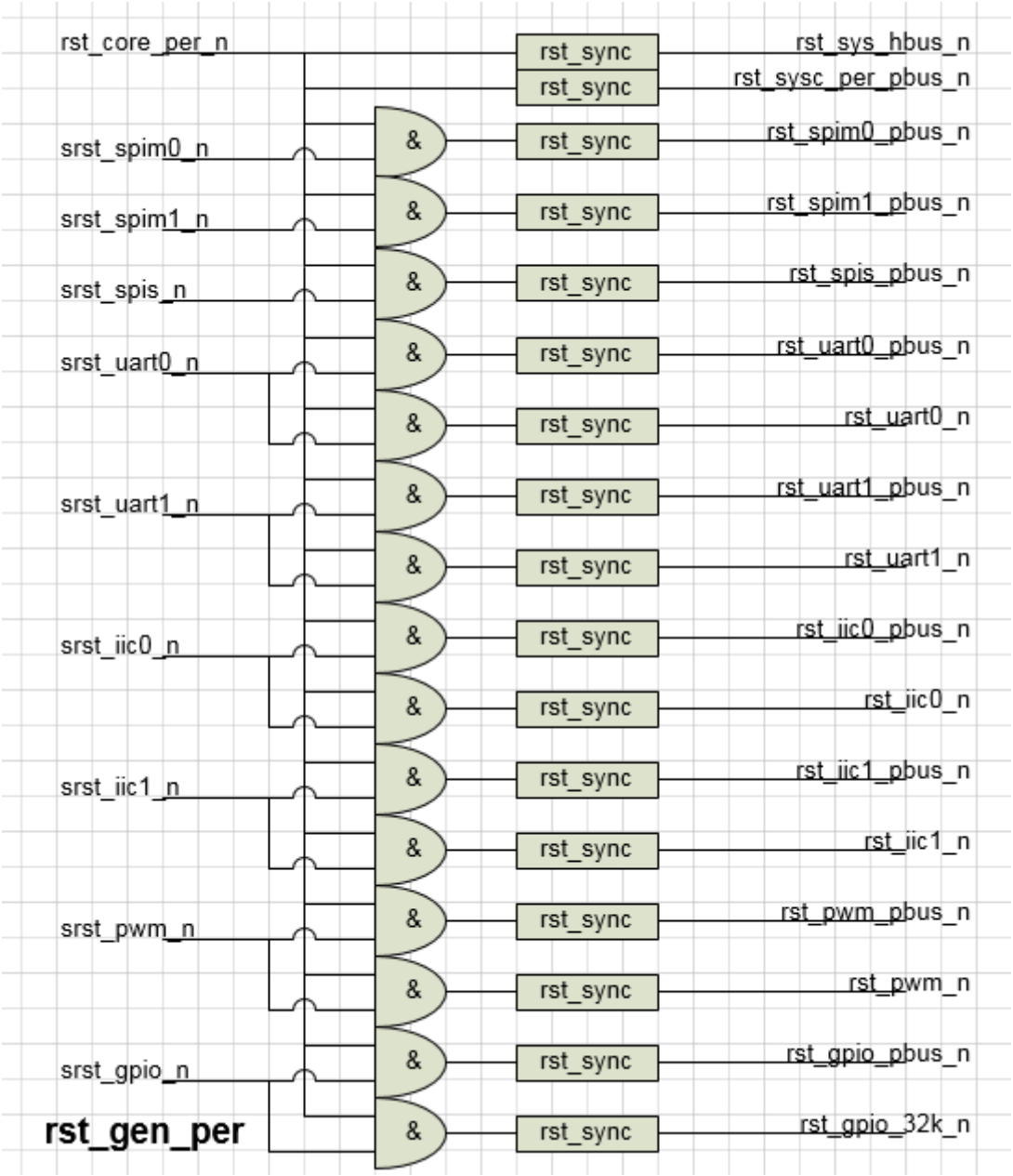
rst_uart2ahb_hbus_n	srst_uart2ahb_n	clk_hbus	uart2ahb
rst_uart2ahb_32m_n	srst_uart2ahb_n	clk_32m	uart2ahb
rst_ecc_pbus_n	srst_ecc_n	clk_pbus	ecc
rst_calb32k_32m_n		clk_32m	32K clock calibration logic

The global reset source for all of the reset signals in the above table is rst_core_cpu_n from AWO power domain. The software reset control registers are listed below:

0x20132018[23]	srst_clr_ecc_n
0x20132018 [22]	srst_set_ecc_n
0x20132018 [21]	srst_clr_uart2ahb_n
0x20132018 [20]	srst_set_uart2ahb_n
0x20132018 [13]	srst_clr_qspi_n
0x20132018 [12]	srst_set_qspi_n
0x20132018 [11]	srst_clr_timer_n
0x20132018 [10]	srst_set_timer_n
0x20132018 [9]	srst_clr_wdt_n
0x20132018 [8]	srst_set_wdt_n
0x20132018 [7]	srst_clr_cache_n
0x20132018 [6]	srst_set_cache_n
0x20132018 [5]	srst_clr_dmac_n
0x20132018 [4]	srst_set_dmac_n

8.4 Reset system of PER power domain

The block diagram of the reset system of CPU power domain is shown below:



Rst_gen_per implements software logic and reset synchronization logic. The reset relationship is listed below:

reset name	soft reset source	clock domain	instance
rst_sys_hbus_n		clk_hbus	AHB/APB bus
rst_sysc_per_pbus_n		clk_pbus	system controller
rst_spim0_pbus_n	srst_spim0_n	clk_pbus	SPIM0
rst_spim1_pbus_n	srst_spim1_n	clk_pbus	SPIM1
rst_spis_pbus_n	srst_spis_n	clk_pbus	SPIS
rst_uart0_pbus_n	srst_uart0_n	clk_pbus	UART0
rst_uart0_n	srst_uart0_n	clk_uart0	UART0
rst_uart1_pbus_n	srst_uart1_n	clk_pbus	UART1

reset name	soft reset source	clock domain	instance
rst_uart1_n	srst_uart1_n	clk_uart1	UART1
rst_iic0_pbus_n	srst_iic0_n	clk_pbus	IIC0
rst_iic0_n	srst_iic0_n	clk_iic0	IIC0
rst_iic1_pbus_n	srst_iic1_n	clk_pbus	IIC1
rst_iic1_n	srst_iic1_n	clk_iic1	IIC1
rst_pwm_pbus_n	srst_pwm_n	clk_pbus	PWM
rst_pwm_n	srst_pwm_n	clk_pwm	PWM
rst_gpio_pbus_n	srst_gpio_n	clk_pbus	GPIO
rst_gpio_32k_n	srst_gpio_n	clk_32k	GPIO

The global reset source for all of the reset signals in the above table is rst_core_per_n from AWO power domain. The software reset control registers are listed below:

0x20149018[17]	srst_clr_gpio_n
0x20149018[16]	srst_set_gpio_n
0x20149018 [15]	srst_clr_pwm_n
0x20149018 [14]	srst_set_pwm_n
0x20149018 [13]	srst_clr_spis_n
0x20149018 [12]	srst_set_spis_n
0x20149018 [11]	srst_clr_spim1_n
0x20149018 [10]	srst_set_spim1_n
0x20149018 [9]	srst_clr_spim0_n
0x20149018 [8]	srst_set_spim0_n
0x20149018 [7]	srst_clr_uart1_n
0x20149018 [6]	srst_set_uart1_n
0x20149018 [5]	srst_clr_uart0_n
0x20149018 [4]	srst_set_uart0_n
0x20149018 [3]	srst_clr_iic1_n
0x20149018 [2]	srst_set_iic1_n
0x20149018 [1]	srst_clr_iic0_n
0x20149018 [0]	srst_set_iic0_n

8.5 Table for system clock and reset

power domain	instance name	module clk port name	clock name	clkg	rst_name	srst
PD_AWO	clk_gen_awo	clk_32k	clk_32k		rst_por_n	
		clk_32m	clk_32m		rst_por_n	
		clk_pll	clk_pll		rst_por_n	
	rst_gen_awo	clk_src_32k	clk_src_32k		rst_por_n	
		clk_src_32m	clk_src_32m		rst_por_n	
		clk_src_bus_awo	clk_src_bus_awo		rst_por_n	
		clk_src_hbus	clk_src_hbus		rst_por_n	
		clk_src_ble_mac	clk_src_ble_mac		rst_por_n	
		clk_src_ble_mdm_rx	clk_src_ble_mdm_rx		rst_por_n	
	pmu	clk	clk_bus_awo		rst_bus_awo_n	
	sysc_awo	pclk	clk_bus_awo		rst_bus_awo_n	
	rtc	pclk	clk_bus_awo_rtc	clkg_rtc	rst_rtc_bus_awo_n	srst_rtc_n
		rtc_clk	clk_32k_rtc		rst_rtc_32k_n	
	wic	FCLK	clk_hbus_wic	clkg_wic	rst_wic_hbus_n	
	touch_ctrl	clk_32k	clk_src_32k		rst_touch_32k_n	
		clk_32m	clk_src_32m		rst_touch_32m_n	
	pwr_pwm	clk_32k	clk_src_32k		rst_por_sync_n	
	ble_timing_lp	low_power_clk	clk_32k_ble_lp	clkg_ble_lp	rst_ble_lp_32k_n	srst_ble_lp_n
PD_PER	clk_gen_per	clk_hbus_pd_per	clk_hbus_pd_per		rst_clk_gen_per_n	
		clk_pbus_pd_per	clk_pbus_pd_per		rst_clk_gen_per_n	
		clk_32k_pd_per	clk_32k_pd_per		rst_clk_gen_per_n	

power domain	instance name	module clk port name	clock name	clkg	rst_name	srst
		clk_32m_pd_per	clk_32m_pd_per		rst_clk_gen_per_n	
		clk_pll_pd_per	clk_pll_pd_per		rst_clk_gen_per_n	
	rst_gen_per				rst_core_per_n	
	sysc_per	clk_apb	clk_pbus_sysc_per		rst_sysc_per_pbus_n	
	spim0	pclk	clk_pbus_spim0	clkg_spim0	rst_spim0_pbus_n	srst_spim0_n
		ssi_clk	clk_spim0		rst_spim0_n	
	spim1	pclk	clk_pbus_spim1	clkg_spim1	rst_spim1_pbus_n	srst_spim1_n
		ssi_clk	clk_spim1		rst_spim1_n	
	spis	pclk	clk_pbus_spis	clkg_spis	rst_spis_pbus_n	srst_spis_n
		ssi_clk	clk_spis		rst_spis_n	
	dw_apb_uart0	pclk	clk_pbus_uart0	clkg_uart0	rst_uart0_pbus_n	srst_uart0_n
		sclk	clk_uart0		rst_uart0_n	
	dw_apb_uart1	pclk	clk_pbus_uart1	clkg_uart1	rst_uart1_pbus_n	srst_uart1_n
		sclk	clk_uart1		rst_uart1_n	
	dw_apb_iic0	pclk	clk_pbus_iic0	clkg_iic0	rst_iic0_pbus_n	srst_iic0_n
		ic_clk	clk_iic0		rst_iic0_n	
	dw_apb_iic1	pclk	clk_pbus_iic1	clkg_iic1	rst_iic1_pbus_n	srst_iic1_n
		ic_clk	clk_iic1		rst_iic1_n	
	dw_apb_gpio	pclk	clk_pbus_gpio	clkg_gpio	rst_gpio_pbus_n	srst_gpio_n
		pclk_intr	clk_pbus_gpio		rst_gpio_pbus_n	
		db_clk	clk_32k_gpio	clkg_gpio_db	rst_gpio_32k_n	
	pwm	clk_bus	clk_pbus_pwm	|clkg_pwm*	rst_pwm_pbus_n	srst_pwm_n

power domain	instance name	module clk port name	clock name	clkg	rst_name	srst
		clk_pwm0	clk_pwm0	clkg_pwm0	rst_pwm_n	
		clk_pwm1	clk_pwm1	clkg_pwm1	rst_pwm_n	
		clk_pwm2	clk_pwm2	clkg_pwm2	rst_pwm_n	
		clk_pwm3	clk_pwm3	clkg_pwm3	rst_pwm_n	
		clk_pwm4	clk_pwm4	clkg_pwm4	rst_pwm_n	
PD_CPU	cm0	FCLK	clk_hbus_cpu		rst_cpu_por_n	
		SCLK	clk_hbus_cpu		rst_cpu_por_n	
		HCLK	clk_hbus_cpu_g	hclk_gate	rst_cpu_hbus_n	
		DCLK	clk_hbus_cpu_g	hclk_gate	rst_cpu_por_n	
		SWCLKTCK	clk_sw		rst_cpu_por_n	
	sys_bus	clk_hbus	clk_hbus_sys		rst_sys_hbus_n	
		clk_pbus	clk_pbus_sys		rst_sys_hbus_n	
		clk_bus_awo	clk_bus_awo_sys		rst_sys_bus_awo	
	dw_ahb_dmac	hclk	clk_hbus_dmac	clkg_dmac	rst_dmac_hbus_n	srst_dmac_n
	sysc_cpu	clk_hbus	clk_hbus_sys		rst_sys_hbus_n	
		clk_apb	clk_pbus_sys		rst_sys_hbus_n	
		clk_32m	clk_32m_calb32k		rst_calb32k_32m_n	
	sram_ctrl	clk	clk_hbus_sys		rst_sys_hbus_n	
	brom_ctrl	clk	clk_hbus_sys		rst_sys_hbus_n	
	apb_trap	clk_apb	clk_pbus_sys		rst_sys_hbus_n	
	cache	hclk	clk_hbus_cache	clkg_cache	rst_cache_hbus_n	srst_cache_n

power domain	instance name	module clk port name	clock name	clkg	rst_name	srst
	qspi	pclk	clk_hbus_qspi	clkg_qspi	rst_qspi_hbus_n	srst_qspi_n
		ssi_clk	clk_hbus_qspi		rst_qspi_hbus_n	
	timer	pclk	clk_pbus_timer	|clkg_timer*	rst_timer_pbus_n	srst_timer_n
		timer_1_clk	clk_timer1	clkg_timer1	rst_timer_n	
		timer_2_clk	clk_timer2	clkg_timer2	rst_timer_n	
	wdt	pclk	clk_pbus_wdt	clkg_wdt	rst_wdt_pbus_n	srst_wdt_n
		tclk	clk_32k_wdt		rst_wdt_32k_n	
	adc_ctrl	clk_adc	clk_pbus_sys		rst_sys_hbus_n	
	ecc	clk_apb	clk_pbus_ecc	clkg_ecc	rst_ecc_pbus_n	srst_ecc_n
	uart2ahb	clk_ahb	clk_hbus_uart2ahb	clkg_uart2ahb	rst_uart2ahb_hbus_n	srst_uart2ahb_n
		clk_uart	clk_32m_uart2ahb		rst_uart2ahb_32m_n	
PD_MEM	sram_wrap_sys0	clk	clk_hbus_sys			
	sram_wrap_sys1	clk	clk_hbus_sys			
	sram_wrap_sys2	clk	clk_hbus_sys			
	sram_wrap_sys3	clk	clk_hbus_sys			
	sram_wrap_sys4	clk	clk_hbus_sys			
	sram_wrap_cache	clk	clk_hbus_sys			
PD_BLE	ble_mac	master1_gclk	clk_ble_mac	clkg_ble	rst_ble_mac_n	srst_ble_n
		master2_gclk	clk_ble_mac	clkg_ble	rst_ble_mac_n	srst_ble_n
		crypt_gclk	clk_ble_mac	clkg_ble	rst_ble_mac_n	srst_ble_n

power domain	instance name	module clk port name	clock name	clkg	rst_name	srst
		hclk	clk_hbus_ble	clkg_ble	rst_ble_hbus_n	srst_ble_n
		hgclk	clk_hbus_ble	clkg_ble	rst_ble_hbus_n	srst_ble_n
	ble_mdm	hclk	clk_hbus_ble	clkg_ble	rst_ble_hbus_n	srst_ble_n
		clk_rf_rx	clk_ble_mdm_rx	clkg_ble	rst_ble_mdm_rx_n	srst_ble_n
		clk_rf_tx	clk_32m_mdm_tx	clkg_ble	rst_ble_mdm_tx_n	srst_ble_n

The first column indicates the power domain of the module.
The second column indicates the name of the module.
The third column indicates the clock input port name of the module
The fourth column indicates the clock name which is connected to the clock port.
The fifth column indicates the clock gate control signal
The sixth column indicates the reset signal name.
The seventh column indicates the clock domain.

9 IO Mux

The IO mux table of BX2400 is shown below.

pad name	IO	func0		func1			func2			func3			func4		
		sig_name	io	sig_name	io	en	sig_name	io	en	sig_name	io	en	sig_name	io	en
P24	B	gpio[24]	B	qspi_cs_n	O	qspi_en[0]									
P25	B	gpio[25]	B	qspi_clk	O	qspi_en[0]									
P26	B	gpio[26]	B	qspi_dat0	B	qspi_en[0]									
P27	B	gpio[27]	B	qspi_dat1	B	qspi_en[1]									
P28	B	gpio[28]	B	qspi_dat2	B	qspi_en[2]									
P29	B	gpio[29]	B	qspi_dat3	B	qspi_en[3]									
P0	B	swck	I	gpio[0]	B	gpio00_en									
P1	B	swd	B	gpio[1]	B	gpio01_en									
P2	B	gpio[2]	B	spim0_cs1	O	spim0_cs1_en				func_io[0]	B	func_io_en[0]			
P3	B	gpio[3]	B	spim0_cs0	O	spim0_en	spis_cs	I	spis_en	func_io[1]	B	func_io_en[1]			
P4	B	gpio[4]	B	spim0_clk	O	spim0_en	spis_clk	I	spis_en	func_io[2]	B	func_io_en[2]			
P5	B	gpio[5]	B	spim0_miso	I	spim0_en	spis_miso	O	spis_en	func_io[3]	B	func_io_en[3]			
P6	B	gpio[6]	B	spim0_mosi	O	spim0_en	spis_mosi	I	spis_en	func_io[4]	B	func_io_en[4]			

P7	B	gpio[7]	B	spim1_cs1	O	spim1_cs1_en	ble_mac_dbg[0]	O	ble_mac_dbg_en[0]	func_io[5]	B	func_io_en[5]	rfif_clk	I	rfif_en
P8	B	gpio[8]	B	spim1_cs0	O	spim1_en	ble_mac_dbg[1]	O	ble_mac_dbg_en[1]	func_io[6]	B	func_io_en[6]	rfif_rx[0]	O	rfif_en
P9	B	gpio[9]	B	spim1_clk	O	spim1_en	ble_mac_dbg[2]	O	ble_mac_dbg_en[2]	func_io[7]	B	func_io_en[7]	rfif_rx[1]	O	rfif_en
P10	B	gpio[10]	B	spim1_miso	I	spim1_en	ble_mac_dbg[3]	O	ble_mac_dbg_en[3]	func_io[8]	B	func_io_en[8]	rfif_tx[0]	I	rfif_en
P11	B	gpio[11]	B	spim1_mosi	O	spim1_en	ble_mac_dbg[4]	O	ble_mac_dbg_en[4]	func_io[9]	B	func_io_en[9]	rfif_tx[1]	I	rfif_en
P12	B	uart2ahb_txd	O	gpio[12]	B	gpio14_en	ble_mac_dbg[5]	O	ble_mac_dbg_en[5]	func_io[10]	B	func_io_en[10]	rfif_tx[2]	I	rfif_en
P13	B	uart2ahb_rxd	I	gpio[13]	B	gpio15_en	ble_mac_dbg[6]	O	ble_mac_dbg_en[6]	func_io[11]	B	func_io_en[11]	rfif_tx[3]	I	rfif_en
P14	B	gpio[14]	B				ble_mac_dbg[7]	O	ble_mac_dbg_en[7]	func_io[12]	B	func_io_en[12]	rfif_tx[4]	I	rfif_en
P15	B	gpio[15]	B							func_io[13]	B	func_io_en[13]	rfif_tx[5]	I	rfif_en
P16	B	gpio[16]	B							func_io[14]	B	func_io_en[14]			
P17	B	gpio[17]	B							func_io[15]	B	func_io_en[15]	rfif_tx[6]	I	rfif_en
P18	B	gpio[18]	B							func_io[16]	B	func_io_en[16]	rfif_tx[7]	I	rfif_en
P19	B	gpio[19]	B							func_io[17]	B	func_io_en[17]	rfif_tx[8]	I	rfif_en

P20	B	gpio[20]	B							func_io[18]	B	func_io_en[18]	rfif_tx[9]	I	rfif_en
P21	B	gpio[21]	B							func_io[19]	B	func_io_en[19]	rfif_tx[10]	I	rfif_en
P22	B	gpio[22]	B							func_io[20]	B	func_io_en[20]	rfif_tx[11]	I	rfif_en
P23	B	gpio[23]	B							func_io[21]	B	func_io_en[21]			

The first column of the above table indicates the name of the IO of the chip.
The second column of the above table indicates the attribute of the IO. All of the IOs in the table are bidirectional IO.
The following columns are composed of 5 function groups which are named as fun0 to fun4. Each function group represents an optional function for the corresponding IO. Blank cells means no function. Each function group is composed of three column which are named as sig_name, io and en except the first function group which is composed of only two column which are named as sig_name and en. Column sig_name indicates the signal name for the corresponding IO when current function group is enabled. Column io indicates the direction for the corresponding IO when the current function group enabled. Column en indicates the enable signal name comes from the programmable control registers. The enable signal is high active and when the enable signal is set high, then the corresponding IO will act as the enabled function. There is only one of the enable signals in the same row can be high at the same time. And if all of the enable signals are low then the first function group is enabled for the corresponding IO.

The function group is illustrated below:

- Gpio[29:0] is the system general purpose IO function, the control module is the GPIO in the PER power domain.
- Swck and swd is the debug port of ARM cortex M0+ cpu.
- Uart2ahb_txd and uart2ahb_rxd is the system uart interface. The control module is the UART2AHB module in the CPU power domain.
- Qspi* is the four wire SPI interface, The control module is the QSPI in the CPU power domain.
- Spim0_* is the first SPI master interface. The control module is the SPIM0 in the PER power domain.
- Spim1_* is the second SPI master interface. The control module is the SPIM1 in the PER power domain.
- Ble_mac_dbg[7:0] is the debug port for BLE MAC and BLE modem. The control module is the BLE MAC and BLE modem in the BLE power domain.
- Rfif_* is the RF interface. The control module is the RF modules in the ANALOG power domain.
- Func_io[21:0] is the system programmable shared IO. The control module is the IO share logic.

The programmable registers for the enable signals are listed below:

0x20132020[23:16]	ble_mac_dbg_en
0x20132020 [13]	rfif_en
0x20132020 [12]	gpio15_en
0x20132020 [11]	gpio14_en
0x20132020 [10]	gpio01_en
0x20132020 [9]	gpio00_en

0x20132020 [8]	spis_en
0x20132020 [7]	spim1_cs1_en
0x20132020 [6]	spim1_en
0x20132020 [5]	spim0_cs1_en
0x20132020 [4]	spim0_en
0x20132020 [3:0]	qspi_en
0x20132024[21:0]	func_io_en

The func_io is shared by many interface modules and each bit of the func_io can be programmed to act as any of the shared function. The share table is listed below:

signal name	io	idx
uart0_txd	O	0
uart0_rxd	I	1
uart0_cts	I	2
uart0_rts	O	3
uart1_txd	O	4
uart1_rxd	I	5
iic0_scl	B	6
iic0_sda	B	7
iic1_scl	B	8
iic1_sda	B	9
pwm0	O	10
pwm1	O	11
pwm2	O	12
pwm3	O	13
pwm4	O	14

Column signal name indicates the signal name for the corresponding IO when current function is selected. The function in the above table is illustrated below:

- Uart0_* is the first uart interface. The control module is the UART0 in the PER power domain.
- Uart1_* is the second uart interface. The control module is the UART1 in the PER power domain.
- Iic0_* is the first iic interface. The control module is the IIC0 in the PER power domain.
- Iic1_* is the second iic interface. The control module is the IIC1 in the PER power domain.
- Pwm* is the pulse width modulation interface. The control module is the PWM in the PER power domain.

Column io indicates the direction for the corresponding IO when the current function is selected.

Column idx indicates the predefined index for the function. There is a programmable

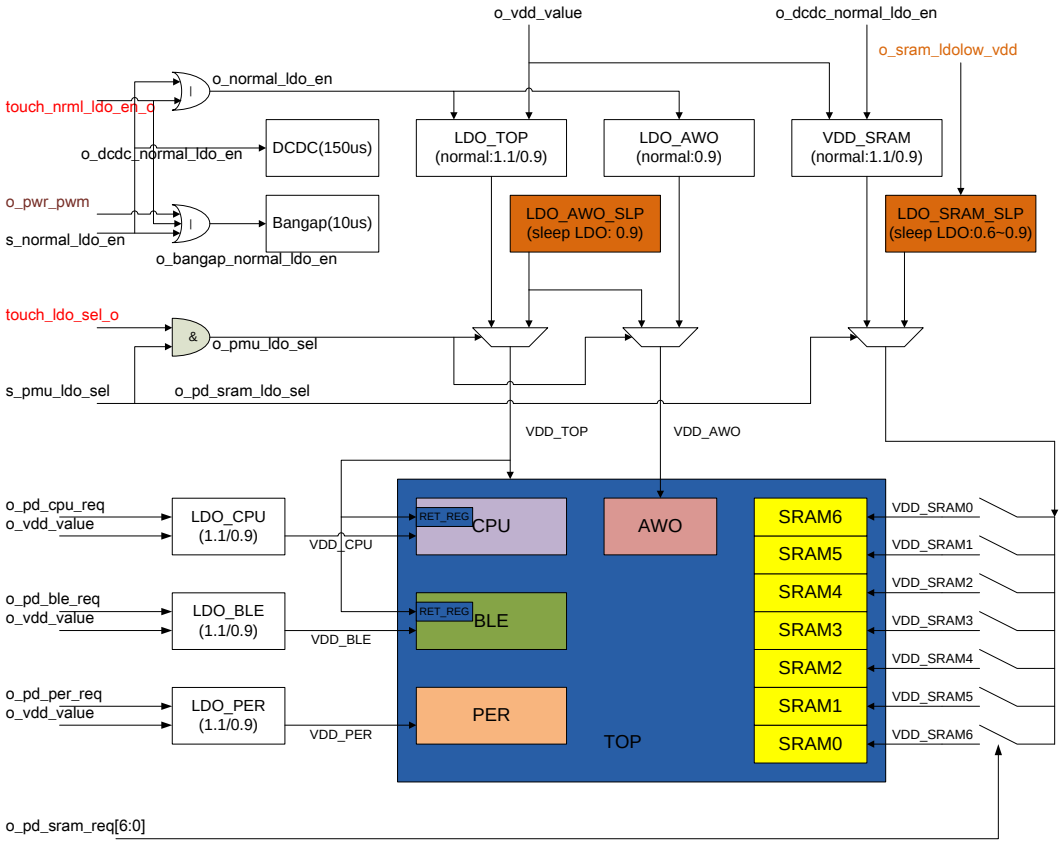
control register name as func_io*_sel for each of func_io. The function in the above table which index equals to the value of control register is selected for the func_io. The control registers are listed below:

0x20132030[31:28]	func_io07_sel
0x20132030 [27:24]	func_io06_sel
0x20132030 [23:20]	func_io05_sel
0x20132030 [19:16]	func_io04_sel
0x20132030 [15:12]	func_io03_sel
0x20132030 [11:8]	func_io02_sel
0x20132030 [7:4]	func_io01_sel
0x20132030 [3:0]	func_io00_sel
0x20132034[31:28]	func_io15_sel
0x20132034 [27:24]	func_io14_sel
0x20132034 [23:20]	func_io13_sel
0x20132034 [19:16]	func_io12_sel
0x20132034 [15:12]	func_io11_sel
0x20132034 [11:8]	func_io10_sel
0x20132034 [7:4]	func_io09_sel
0x20132034 [3:0]	func_io08_sel
0x20132038 [23:20]	func_io21_sel
0x20132038 [19:16]	func_io20_sel
0x20132038 [15:12]	func_io19_sel
0x20132038 [11:8]	func_io18_sel
0x20132038 [7:4]	func_io17_sel
0x20132038 [3:0]	func_io16_sel

10 PMU

10.1Power supply diagram

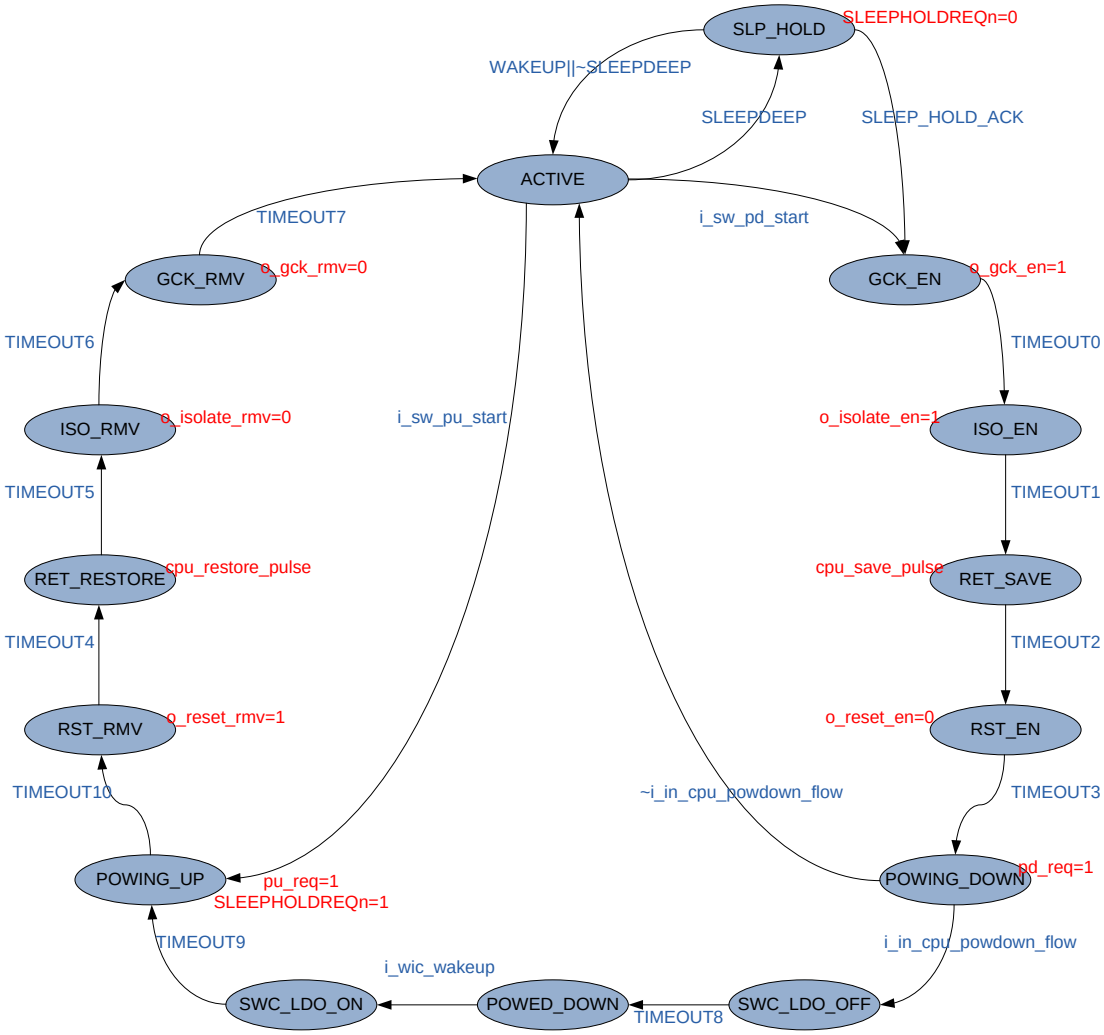
The power domain is described in the chapter 4.5. The relationship of the power supply and the power domains is shown in the diagram below:



The control signal o_sram_ldolow_vdd in orange is from control register which defines the voltage level of the sleep LDO for SRAM. The control signals touch_nrml_ldo_en_o and touch_ldo_sel_o in red is from the touch controller. When touch controller is active, the BANGAP, the LDO_TOP and the LDO_AWO must be on and the LDO switch must be switched to the normal LDO to supply current for the requirement of the touch controller logic and the touch ADC. The control signal o_pwr_pwm in brown is from the power PWM controller which controls the on/off of the LDO for external device, which will power on the BANGAP too. All of the other control signals are from PMU FSM.

10.2PMU FSM

The FSM of PMU is shown in the diagram below:



The states are described below:

- ACTIVE: normal working mode, CPU is power on and at least one SRAM is power on.
- SLP_HOLD: this state is used to handshake with CPU to hold the CPU in the SLEEP state
- GCK_EN: close the clock gate for the target power domain
- ISO_EN: isolate the signal for the target power domain
- RET_SAVE: save the value of the retention registers
- RST_EN: reset the target power domain
- POWING_DOWN: power off the target LDO
- SWC_LDO_OFF: switch the clock of the AWO power domain to the 32KHz clock, shut down the 32MHz clock, switch to the sleep LDO and shut down the normal LDO
- POWED_DOWN: deep sleep or extended sleep state
- SWC_LDO_ON: power up the normal LDO, switch to the normal LDO, power up the 32MHz clock and switch the clock of the AWO power domain to 32MHz clock
- POWING_UP: power up the target LDO

- RST_RMV: disable the reset of the target power domain
- RET_RESTORE: load the value of the retention register
- ISO_RMV: disable the isolation of the target power domain
- GCK_RMV: disable the clock gating of the target power domain

The TIMEOUT condition in the diagram is determined by the following programmable registers:

0x20201014[31:24]	pmu_seq_time
0x20201014[23:20]	clk_swc_dly
0x20201014[19:16]	ldo_swc_dly
0x20201014[13:8]	osc_stb_dly
0x20201014[4:0]	ldo_stb_dly
0x20201018[7:0]	ldo_setup_time

- clk_swc_dly is in both 32MHz clock cycle and 32KHz clock cycle and is the latency of the AWO power domain clock switching between 32MHz clock and 32KHz clock
- ldo_swc_dly is in 32KHz clock cycle and is the latency of the power supply switching between the normal LDO and the sleep LDO.
- osc_stb_dly is in 32KHz clock cycle and is the latency of the powering up time of the 32MHz RC OSC clock.
- ldo_stb_dly is in 32KHz clock cycle and is the latency of the powering up time of the DCDC and normal LDO.
- ldo_setup_time is in 32MHz clock cycle and is the latency of the powering up time of the normal LDO.

The condition TIMEOUT0~7 are determined by the register pmu_seq_time. The condition TIMEOUT8 is determined by register clk_swc_dly and register ldo_swc_dly. The condition TIMEOUT9 is determined by register clk_swc_dly, register ldo_swc_dly, register osc_stb_dly and register ldo_stb_dly. The condition TIMEOUT10 is determined by register ldo_setup_time.

There are three working mode of the PMU:

- MODE1: Power on one or more power domains of BLE/PER/SRAM*
- MODE2: Power off one or more power domains of BLE/PER/SRAM*
- MODE3: Power on/off all of the power domains of CPU/BLE/PER/SRAM*

10.3PMU mode1

The programming flow is listed below:

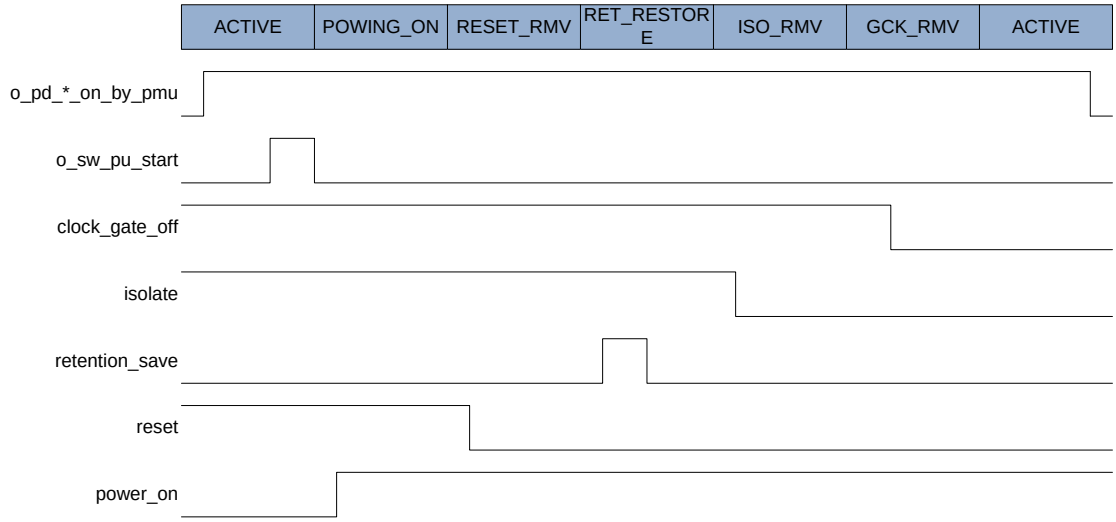
- Writing to programmable control register 0x2020100C to define which power domain will be powered on. The related bit of register 0x2020100C is illustrated below:

0x2020100C [26]	pd_per_on_by_pmu	high means PER power domain will be powered on
0x2020100C [24]	pd_blemac_on_by_pmu	high means BLE power domain will be powered on

0x2020100C [22:16]	pd_sram_on_by_pmu	Each bit represents one block of SRAM. High means corresponding SRAM will be powered on

- If BLE power domain will be powered on, then write to the programmable register 0x20201000[23] which is called blemac_ret_en to determine whether the value stored in the retention registers will be reloaded to the registers of BLE MAC. Writing 1 to this register means the value of retention registers will be reloaded to the registers of BLE MAC which means BLE MAC will return to the status before sleep. Writing 0 to this register means the value of retention registers will not be reloaded to the registers of BLE MAC which means BLE MAC will return to the reset status.
- Writing 1 to programmable register 0x2020100C [31] which is named as sw_pu_start to start the power on flow.

The waveform of MODE1 is shown below:



10.4PMU mode2

The programming flow is listed below:

- Writing to programmable control register 0x2020100C to define which power domain will be powered on. The related bit of register 0x2020100C is illustrated below:

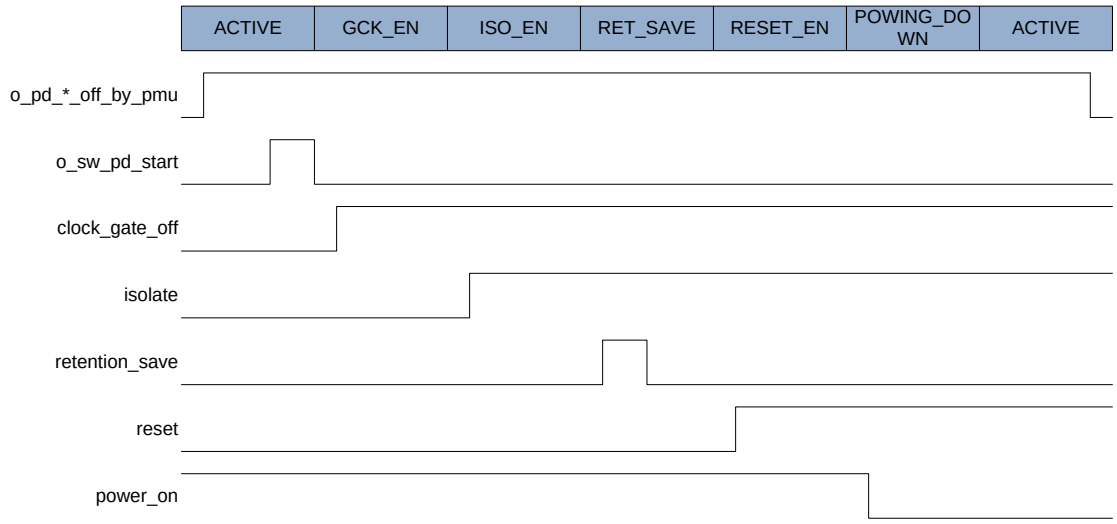
0x2020100C [10]	pd_per_off_by_pmu	high means PER power domain will be powered off
0x2020100C [8]	pd_blemac_off_by_pmu	high means BLE power domain will be powered off
0x2020100C [6:0]	pd_sram_of_by_pmu	Each bit represents one block of SRAM. High means corresponding SRAM will be powered off

- If BLE power domain will be powered off, then write to the programmable register

0x20201000[23] which is called blemac_ret_en to determine whether the value of the BLE MAC register will be stored into the retention register. Writing 1 to this register means the value of the BLE MAC registers will be stored into the retention registers. Writing 0 to this register means the value of the BLE MAC registers will not be stored into the retention registers.

- Writing 1 to programmable register 0x2020100C [30] which is named as sw_pd_start to start the power off flow.

The waveform of MODE2 is shown below:



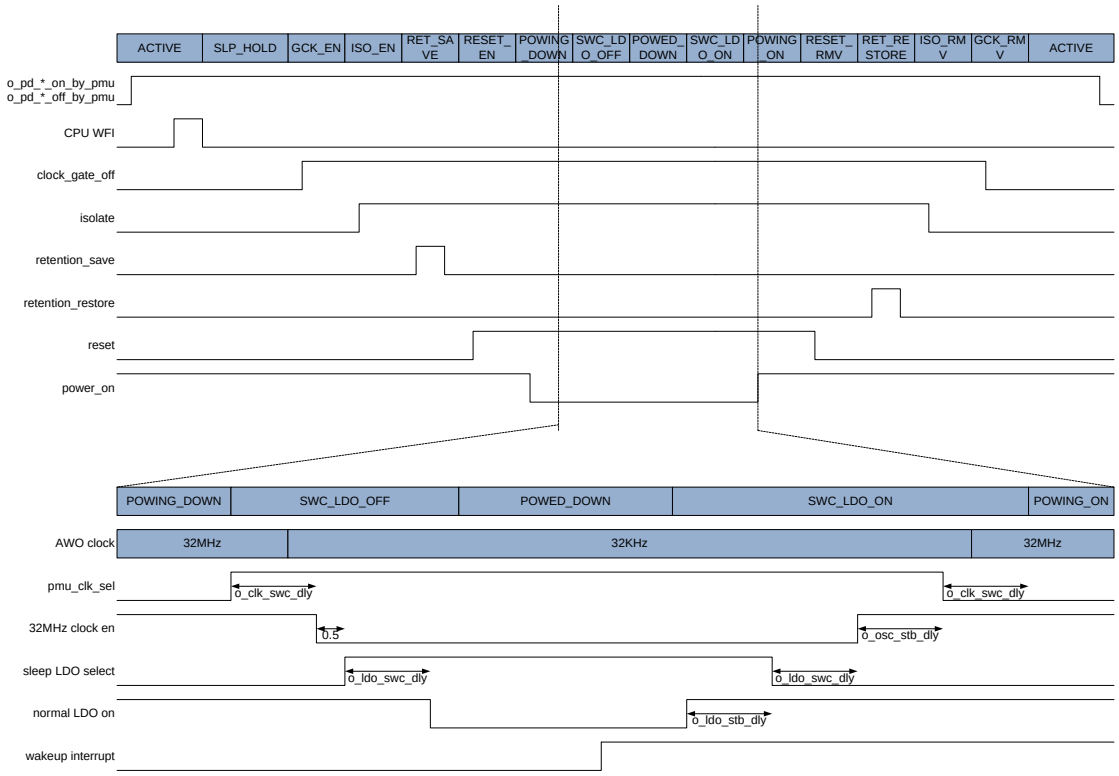
10.5PMU mode3

MODE3 is triggered by executing WFI of CPU with the DEEPSLEEP bit in the SCR register (0xE000ED10) of CPU is set to high.

Before the PMU start to power off the related LDOs, one handshake between the PMU and the CPU must be done to keep the CPU in the deep sleep status during the process of powering down the LDOs. This handshake is done in the SLP_HOLD status. And if the handshake fails, then the PMU will return to the ACTIVE status without powering down any LDO.

When the PMU enters POWED_DOWN state, it is hold in this state until the WIC detects any of the wakeup interrupts is triggered. The wake up interrupt includes the BLE low power interrupt, the RTC interrupt, the external interrupt and the touch interrupt.

The waveform of MODE3 is shown below:



There are many function related to the deep sleep status implemented in BX2400 which will be described below.

10.5.1 IO status retention function

BX2400 can hold the IO status after BX2400 goes into deep sleep or extended sleep mode. If this function is enabled, the IO status after sleep is the same as the status before sleep. For example if one of the IO is output high before sleep, then the IO is also output high after sleep if the io retention function is enabled. When BX2400 wakes up, BX2400 can select to quit the IO retention status automatically or quit the IO retention status by software through programming some register. If quit the IO retention automatically, then after BX2400 wakes up, all of the IOs are input. If quit the IO retention by programming registers by software, then the IO status can be the same as the status before sleep after quit from the IO retention status. The related registers are listed below:

0x20201008 [2]	io_ctrl_sel	selection of the control source of the IO status. High means the IO status is controlled by the IO retention register. Low means the IO status is controlled by the system function module
0x20201008 [1]	io_stat_unret_pmu	IO un-retention enable. High means quit from the IO retention status automatically after the system

		wakes up. Low means remain in the IO retention status after the system wakes up.
0x20201008 [0]	io_stat_ret_pmu	IO retention enable. High means when the system sleep, the value of the IO control signal will be stored into the IO retention registers and the IO state is controlled by the IO retention registers to hold the IO status as the same state before sleep. Low means IO retention function is not enabled and all of the IO will be input after system sleeps.

The programming flow for IO retention function is listed below:

- Writing 1 to the register 0x20201008 [0] to enable the IO retention function
- Writing 0 to the register 0x20201008 [1] to disable the automatically IO un-retention function
- CPU execute WFI instruction, and system goes to sleep
- System wakes up
- Configure the related function modules to expected status
- Writing 0 to 0x20201008 [2] to change the IO control source from the IO retention logic to the system function modules.

10.5.2 SRAM retention function

The SRAM of BX2400 is composed of 7 sub SRAM blocks which can hold data during system sleeping period. Then when system wakes up, the data in the SRAM does not lost and the system status is the same as the status before sleep. The related registers are listed below:

0x2020100C[22:16]	pd_sram_on_by_pmu	indicates whether the SRAM blocks will be powered on together with CPU power domain. Each bit represents one SRAM block.
0x2020100C [6:0]	pd_sram_off_by_pmu	indicates whether the SRAM blocks will be powered off together with CPU power domain. Each bit represents one SRAM block.
0x20201010 [19:16]	sram_retention_vdd	indicates the voltage of the SRAM during sleep state when the SRAM is programmed to hold data in the sleep status.
0x20201010 [6:0]	sram_retention_req	indicates whether the SRAM blocks need to hold data during sleep state. Each bit represents one SRAM block. High means the VDD of the SRAM block will not be shut down to hold the data in the SRAM.

Note that if the SRAM block is programmed to hold data in the sleep state, then the corresponding bit of register pd_sram_on_by_pmu must be 1, the corresponding bit of register pd_sram_off_by_pmu must be 0 and the corresponding bit of register sram_retention_req must be 1. If the SRAM block is programmed to not hold the data then the corresponding bit of register pd_sram_off_by_pmu must be 1 and the corresponding bit of register sram_retention_req must be 0.

10.5.3 Switching voltage of CPU/BLE/PER power domain

The voltage of the VDD of CPU, PER and BLE power domain is the same and can be 0.9V or 1.1V. When the frequency of CPU is bigger than 32MHz, then the voltage of VDD must be set to 1.1V. When the frequency of CPU is less than or equal to 32MHz, then the voltage of VDD can be set to 0.9V to save power. The VDD can not be changed when the CPU is working, so the switching of voltage is done during sleep status by the hardware FSM. The related registers are listed below:

0x20201048[0]	vdd_value	voltage of VDD after next wake up from sleep status. High means the voltage of VDD will be changed to 0.9V after the next wake up. Low means the voltage of VDD will be changed to 1.1V after the next wake up.

10.5.4 CPU register retention function

BX2400 supports to hold the CPU status during sleep status and when CPU wakes up, the CPU will be in the same status as it is before sleep. Then CPU can execute the instruction from where it sleeps. If this function is not enabled, the CPU will reboot after CPU wakes up. The related registers are listed below:

0x20201000[22]	o_cpu_ret_en	CPU retention function enable. High means CPU will hold the values of all of the registers in the CPU and will continue to execute the instruction from where it sleeps. Low means CPU will not hold the register value and will reboot after CPU wakes up.

10.5.5 Programming flow

The programming flow is shown below:

- Program the registers described in chapter 10.2 to determine the delay of the FSM

- Program the registers described in chapter 10.2 and 10.3 to determine which power domains will be powered on/off together with CPU power domain.
- Program the registers described in chapter 10.5.1 for the IO retention function
- Program the registers described in chapter 10.5.2 for the wake up voltage of VDD
- Program the registers described in chapter 10.5.3 for the CPU retention function
- Program the register 0x20201000[23] for the BLE retention function
- CPU execute WFI instruction to go to sleep
- CPU wakes up
- Program the registers for the clock generator
- Program the related peripheral interface function modules
- Program 0x20201008 [2] to switch the IO control source from the IO retention register to the function module
- CPU works

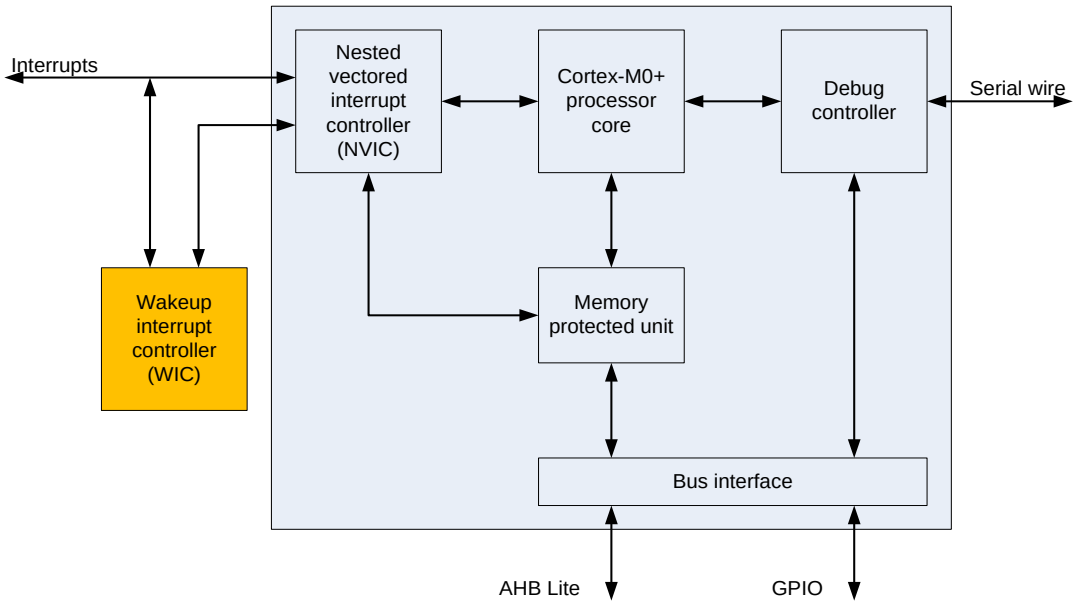
10.6Clock and reset

The clock of PMU is clk_bus_awo. (refer to chapter 7)
The reset of PMU is rst_bus_awo_n. (refer to chapter 8)
There are no clock gating and software reset for PMU.

11 CPU

11.1General description

BX2400 integrates Cortex M0+ processor as the CPU. The Cortex-M0+ processor is an ultra-low power 32-bit ARM Cortex processor designed for a broad range of embedded applications. The block diagram of Cortex M0+ processor is shown below:



The Cortex-M0+ processor is built on a highly area and power optimized 32-bit processor core, with a 2-stage pipeline von Neumann architecture. The processor delivers exceptional energy efficiency through a small but powerful instruction set and extensively optimized design, providing high-end processing hardware including a single-cycle multiplier.

The Cortex-M0+ processor implements the ARMv6-M architecture, which is based on the 16-bit Thumb instruction set and includes Thumb-2 technology. This provides the exceptional performance expected of a modern 32-bit architecture, with a higher code density than other 8-bit and 16-bit microcontrollers.

The cortex-M0+ processer integrates a nested vectored interrupt controller, a debug controller and a memory protect unit.

11.2Feature list

- The ARMv6-M Thumb instruction set.
- Thumb-2 technology.
- An ARMv6-M compliant 24-bit SysTick timer.
- A single cycle 32-bit hardware multiplier.
- C Application Binary Interface compliant exception model. This is the ARMv6-M, C Application Binary Interface (C-ABI) compliant exception model that enables the use of pure C functions as interrupt handlers.
- Low power sleep-mode entry using Wait For Interrupt (WFI), Wait For Event (WFE) instructions, or the return from interrupt sleep-on-exit feature.
- Up to 32 external interrupt inputs, each with four levels of priority.
- Dedicated Non-Maskable Interrupt (NMI) input.
- Support for both level-sensitive and pulse-sensitive interrupt lines.
- Wake-up Interrupt Controller (WIC), providing ultra-low power sleep mode support.
- Relocation of the vector table.
- Four hardware breakpoints.
- Two watchpoints.
- Program Counter Sampling Register (PCSR) for non-intrusive code profiling, if at least one hardware data watchpoint is implemented.
- Single step and vector catch capabilities.
- Support for unlimited software breakpoints using BKPT instruction.
- Non-intrusive access to core peripherals and zero-waitstate system slaves through a compact bus matrix. A debugger can access these devices, including memory, even when the processor is running.
- Full access to core registers when the processor is halted.
- Optional, low gate-count CoreSight compliant debug access through a *Debug Access Port* (DAP) supporting either Serial Wire or JTAG debug connections.
- Single 32-bit AMBA-3 AHB-Lite system interface that provides simple. integration to all system peripherals and memory.
- Optional single 32-bit single-cycle I/O port.
- Optional single 32-bit slave port that supports the DAP.
- Eight user configurable memory regions.
- Eight sub-region disables per region.
- Execute never (XN) support.
- Default memory map support.

11.3NVIC

The Cortex-M0+ processor closely integrates a configurable Nested Vectored Interrupt Controller (NVIC), to deliver industry-leading interrupt performance. The NVIC:

- Includes a Non-Maskable Interrupt (NMI).
- Provides zero jitter interrupt option.
- Provides four interrupt priority levels.

The tight integration of the processor core and NVIC provides fast execution of Interrupt Service Routines (ISRs), dramatically reducing the interrupt latency. This is achieved through the hardware stacking of registers, and the ability to abandon and restart load-multiple and store-multiple operations. Interrupt handlers do not require any assembler wrapper code, removing any code overhead from the ISRs. Tail-chaining optimization also significantly reduces the overhead when switching from one ISR to another.

External interrupt signals connect to the NVIC, and the NVIC prioritizes the interrupts. Software can set the priority of each interrupt. The NVIC and the Cortex-M0+ processor core are closely coupled, providing low latency interrupt processing and efficient processing of late arriving interrupts.

A 24 bits system timer(SysTick) is implemented in NVIC which can also interrupt the CPU.

11.4Memory protection unit

the processor supports the ARMv6 Protected Memory System Architecture model. The MPU provides full support for:

- Eight unified protection regions.
- Overlapping protection regions, with ascending region priority: 7 is the highest priority, 0 is the lowest priority.
- Access permissions.
- Exporting memory attributes to the system.

MPU mismatches and permission violations invoke the HardFault handler.

11.5Clock and reset

The clock of CPU is clk_hbus_cpu, clk_hbus_cpu_g and clk_sw. (refer to chapter 7)
The reset of CPU is rst_cpu_por_n and rst_cpu_hbus_n. (refer to chapter 8)
There is no clock gating logic and software reset for CPU.

12 System interrupt

There are altogether 19 system IRQ in BX2400 which are listed below:

- IQR No. 0: system watch dog interrupt which is from WDT in the CPU power domain
- IQR No. 1: BLE sleep timer interrupt which is from the BLE low power controller in the AWO power domain
- IQR No. 2: BLE MAC interrupt which is from the BLE MAC in the BLE power domain
- IQR No. 3: system RTC interrupt which is from the RTC in the AWO power domain
- IQR No. 4: external interrupt shared with the touch controller interrupt which is from the external interrupt controller in the AWO power domain and the touch controller in the AWO power domain.
- IQR No. 5: ECC interrupt which is from the ECC module in the CPU power domain
- IQR No. 6: system DMA interrupt which is from the DMA controller in the CPU power domain
- IQR No. 7: 4 wire SPI master controller interrupt which is from the QSPI in the CPU power domain
- IQR No. 8: SPI master 0 interrupt which is from the SPIM0 in the PER power domain
- IQR No. 9: SPI master 1 interrupt which is from the SPIM1 in the PER power domain
- IQR No. 10: SPI slave interrupt which is from the SPIS in the PER power domain
- IQR No. 11: UART controller 0 interrupt which is form the UART0 in the PER power domain
- IQR No. 12: UART controller 1 interrupt which is form the UART1 in the PER power domain
- IQR No. 13: IIC controller 0 interrupt which is form the IIC0 in the PER power domain
- IQR No. 14: IIC controller 1 interrupt which is form the IIC1 in the PER power domain
- IQR No. 15: GPIO interrupt which is from the GPIO in the PER power domain
- IQR No. 16: system timer interrupt which is from the TIMER in the CPU power domain
- IQR No. 17: system software interrupt which is from the software interrupt register in the system controller of CPU power domain
- IQR No. 18: system SRAM monitor interrupt which is from the system SRAM monitor in the CPU power domain

12.1BLE sleep timer interrupt

BX2400 may go to sleep when there are no BLE transfers and wakes up before the next BLE transfer. There is a dedicated time counter implemented for BLE. When there are no BLE transfers, BLE MAC may go to sleep, and then the BLE sleep timer will start to work and is responsible to wake up system and BLE MAC before next transfer. Before BLE starts the next transfer, BX2400 needs some time to recover the status from the sleep state. So the BLE sleep timer will wake up the system with a predefined time ahead of the next BLE transfer. BLE sleep timer wakes up the system

through interrupt. There are two interrupts implemented for BLE sleep timer. And the time period from the interrupt to the next BLE transfer is programmable for both of the interrupt. The control registers are listed below:

0x2010003C[20:10]	twosc	defines the time period from the second BLE sleep timer interrupt to the next BLE transfer in system 32KHz clock
0x2010003C[9:0]	twrm	defines the time period from the first BLE sleep timer interrupt to the next BLE transfer in system 32KHz clock

The interrupt related registers are listed below:

0x20201030[9:8]	blelp_rawinrp_value	raw interrupt status
0x20201030[1:0]	blelp_inrp_value	interrupt status
0x20201034[1:0]	blelp_inrp_clr	interrupt clear, writing 1 to the register will clear the interrupt
0x20201000[6:5]	blelp_intr_en	interrupt mask for the BLE sleep timer interrupt

12.2External interrupt

External interrupt is shared with the touch interrupt in BX2400. External interrupt can be used to wake up the system when system is in sleep status. There are altogether 5 IOs which can be programmed to be the source of the external interrupt. The relationship between the IO and the external interrupt is listed below:

interrupt	**IO**	**enable signal**
ext_intr0	GPIO15	0x20201024[20]
ext_intr1	GPIO16	0x20201024[21]
ext_intr2	GPIO17	0x20201024[22]
ext_intr3	GPIO22	0x20201024[23]
ext_intr4	GPIO23	0x20201024[24]

The external interrupt related programmable registers are listed below:

0x20201024[24:20]	pin_is_inrp_src	pin share enable for the external interrupt. Each bit represents an external interrupt. High means the corresponding GPIO is the source of external interrupt.
0x20201024 [19:18]	ext_inrp_sense_5	The trigger mode of the touch interrupt, must be 0 which means high level triggers the interrupt.
0x20201024 [17:16]	ext_inrp_sense_4	ext_intr4 interrupt trigger mode. 0 means high level triggers the interrupt. 1 means low level triggers the interrupt. 2 means rising edge

		triggers the interrupt. 3 means falling edge triggers the interrupt
0x20201024 [15:14]	ext_inrp_sense_3	ext_intr3 interrupt trigger mode. 0 means high level triggers the interrupt. 1 means low level triggers the interrupt. 2 means rising edge triggers the interrupt. 3 means falling edge triggers the interrupt
0x20201024 [13:12]	ext_inrp_sense_2	ext_intr2 interrupt trigger mode. 0 means high level triggers the interrupt. 1 means low level triggers the interrupt. 2 means rising edge triggers the interrupt. 3 means falling edge triggers the interrupt
0x20201024 [11:10]	ext_inrp_sense_1	ext_intr1 interrupt trigger mode. 0 means high level triggers the interrupt. 1 means low level triggers the interrupt. 2 means rising edge triggers the interrupt. 3 means falling edge triggers the interrupt
0x20201024 [9:8]	ext_inrp_sense_0	ext_intr0 interrupt trigger mode. 0 means high level triggers the interrupt. 1 means low level triggers the interrupt. 2 means rising edge triggers the interrupt. 3 means falling edge triggers the interrupt
0x20201024 [5:0]	ext_inrp_en	external interrupt enable. Bit 5 is for the touch interrupt. The other 5 bits are for the external interrupt.
0x20201028[13:8]	ext_rawinrp_value	raw interrupt status of the external interrupt and the touch interrupt. Bit 5 is for the touch interrupt and the other 5 bits are for the external interrupt.
0x20201028[5:0]	ext_inrp_value	interrupt status of the external interrupt and the touch interrupt. Bit 5 is for the touch interrupt and the other 5 bits are for the external interrupt.
0x2020102C[5:0]	ext_inrp_clr	interrupt clear of the external interrupt and the touch interrupt. Bit 5 is for the touch interrupt and the other 5 bits are for the external interrupt. Writing 1 to the register bit will clear the corresponding interrupt.

12.3Software interrupt

BX2400 supports to interrupt the CPU by programming the register in the system controller of CPU power domain. The register address is 0x20132044[0]. Writing 1 to the register will interrupt the CPU and writing 0 to the register will clear the interrupt.

12.4System SRAM monitor interrupt

BX2400 supports monitoring the writing to the system SRAM. There are two address ranges which is defined by programmed registers. If CPU writes data to any of these address ranges, the system SRAM monitor interrupt will be triggered. The related programmable registers are listed below:

0x20132050[23:0]	pgspy_addr0_l	start address of the first address range
0x20132054[23:0]	pgspy_addr0_h	end address of the first address range
0x20132058[23:0]	pgspy_addr1_l	start address of the second address range
0x2013205C[23:0]	pgspy_addr1_h	end address of the second address range
0x20132060[1:0]	pgspy_en	address range enable, high active. Each bit is for one address range. High means that writing to the corresponding address range will trigger the interrupt. Writing 0 to the register will clear the corresponding interrupt.
0x20132064[1:0]	pgspy_intr_mask	interrupt mask for the system SRAM monitor interrupt
0x20132068[1:0]	pgspy_intr_raw	raw status of the system SRAM monitor interrupt
0x2013206C[1:0]	pgspy_intr	status of the system SRAM monitor interrupt

13 Power PWM

13.1Genera description

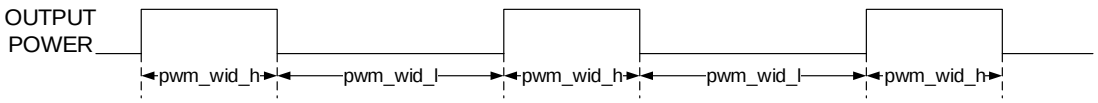
BX2400 has three power supply output which enables BX2400 to be power supplier for external ICs. One of the three power supply output is 1.8V and the others are 3V. The on/off of the three power supply output is controlled by the power PWM.
The on/off control signal can be set to high or low always. High of the control signal power on the external power supply LDO and supply power for the external ICs. Low of the control signal power off the external power supply LDO and disable the power supply for the external ICs. The on/off control signal can also work like a PWM which can repeatly set the LDO on for some time and off for some time.
There are three power PWMs in BX2400 and each controls one of the external power supply LDO. Power PWM 0 controls the 1.8V power output. Power PWM1 and power PWM2 control the 3V power output.
The power PWM is controlled by the following registers

0x202010c0[31:16]	pwm0_wid_h	the high pulse width of the PWM control signal for external power supply 0 in 32KHz clock
0x202010c0[15:0]	pwm0_wid_l	the low pulse width of the PWM control signal for external power supply 0 in 32KHz clock
0x202010c4[31:16]	pwm1_wid_h	the high pulse width of the PWM control signal for external power supply 1 in 32KHz clock
0x202010c4[15:0]	pwm1_wid_l	the low pulse width of the PWM control signal for external power supply 1 in 32KHz clock
0x202010c8[31:16]	pwm2_wid_h	the high pulse width of the PWM control signal for external power supply 1 in 32KHz clock
0x202010c8[15:0]	pwm2_wid_l	the low pulse width of the PWM control signal for external power supply 1 in 32KHz clock
0x202010cc[9]	pwm2_en	power supply enable for external power supply 2. If pwm2_fc_h is high, then the external power supply 2 is forced to be enabled. If pwm2_fc_h is low and pwm2_en is high, then the control of the external power supply 2 is like a PWM signal defined by pwm2_wid_h and pwm2_wid_l.
0x202010cc[8]	pwm2_fc_h	force high enable for external power supply 2. High means the external power supply 2 is forced to be enabled despite of the value of register pwm2_en.
0x202010cc[5]	pwm1_en	power supply enable for external power supply 1. If pwm1_fc_h is high, then the external power supply 1 is forced to be enabled. If pwm1_fc_h is low and

		pwm1_en is high, then the control of the external power supply 1 is like a PWM signal defined by pwm1_wid_h and pwm1_wid_l.
0x202010cc[4]	pwm1_fc_h	force high enable for external power supply 1. High means the external power supply 1 is forced to be enabled despite of the value of register pwm1_en.
0x202010cc[1]	pwm0_en	power supply enable for external power supply 0. If pwm0_fc_h is high, then the external power supply 0 is forced to be enabled. If pwm0_fc_h is low and pwm0_en is high, then the control of the external power supply 0 is like a PWM signal defined by pwm0_wid_h and pwm0_wid_l.
0x202010cc[0]	pwm0_fc_h	force high enable for external power supply 0. High means the external power supply 0 is forced to be enabled despite of the value of register pwm0_en.

If pwm*_fc_h is set to high, then the output is forced to be high. If pwm*_fc_h is set to be low and pwm*_en is set to be high, then the output waveform is a waveform like a PWM which high pulse width and low pulse width is defined by pwm*_wid_h and pwm*_wid_l. If pwm*_fc_h and pwm*_en are both low, then the output is forced to be low.

The waveform of the output power supply is shown below:



13.2Clock and reset

The clock of power PWM is clk_src_32k. (refer to Chapter 7)
The reset of power PWM is rst_por_sync_n. (refer to Chapter 8)
There are no clock gating control register and software reset register for power PWM.

14 WIC (Wakeup interrupt controller)

14.1General description

The wakeup interrupt controller is one component of ARM cortex M0 plus. The WIC is in the AWO power domain. Only 3 interrupts of 19 system interrupts can wake up the system from deep sleep or extended sleep status. The three interrupts are BLE sleep timer interrupt, RTC interrupt and external interrupt which is shared with the touch interrupt. The PMU is waked up by the WIC from deep sleep mode or extended sleep mode. The WIC detects the status of the wakeup interrupts and inform the PMU that it is time to wake up.

The WIC is enabled only when the DEEPSLEEP bit in the SCR register (0xE000ED10) of CPU is set to high. When the WIC is enabled and the processor enters deep sleep mode, the power management unit in the system can power down most of the Cortex-M0+ processor. This has the side effect of stopping the SysTick timer in the CPU. When WIC receives a wakeup interrupt, it takes a number of clock cycles to wake up the processor and restored its state.

14.2Clock and reset

The clock for WIC is clk_hbus_wic. (refer to Chapter 7)
The reset for WIC is rst_wic_hbus_n. (refer to Chapter 8)
The clock gating control register is :

sysc_awo	0x20201000	0x004	[9]	clkg_wic_clr
sysc_awo	0x20201000	0x004	[8]	clkg_wic_set

There is no software reset for WIC.

15 RTC

15.1General description

A real time counter is integrated in the AWO power domain to provide the real time for the system. The RTC has a programmable 32 bits counter. The counter is counting in the 32KHz clock. When the counter load register is programmed, the counter is loaded with a start value that allows the counter to increment. The RTC interrupt is triggered when the counter reaches the preprogrammed value. After the counter reaches the preprogrammed value, the counter will wrap to zero or count continuously which is programmable. The RTC is connected to the system APB bus. The APB bus clock is asynchronous to the counter clock. Note that, After system wakes up from deep sleep or extended sleep state, the register of the RTC current counter value can not be read from the APB bus until two 32KHz clocks later.

15.2Clock and reset

The clock of RTC is clk_32k_rtc and clk_bus_awo_rtc. (refer to Chapter 7)
The reset of RTC is rst_rtc_32k_n and rst_rtc_bus_awo_n. (refer to Chapter 8)
The clock gating control register of RTC is:

sysc_awo	0x20201000	0x004	[3]	clkg_rtc_clr
sysc_awo	0x20201000	0x004	[2]	clkg_rtc_set

The software reset register of RTC is:

sysc_awo	0x20201000	0x040	[5]	srst_rtc_n_clr
sysc_awo	0x20201000	0x040	[4]	srst_rtc_n_set

16 BLE MAC sleep timer

16.1General description

The BLE MAC sleep timer is a time counter for BLE MAC during the period that the BLE MAC is in sleep mode. When BLE MAC enters sleep mode. The time count loads the time value from the BLE MAC. And also the wakeup time is loaded in to local registers. The low power counter continues to count until the timer reaches the preprogrammed wakeup time. After the time of wake up, the low power counter will stop counting and the BLE MAC will take over the time counting for the BLE protocol. Before the BLE MAC wakes up, the low power controller will trigger the BLE low power interrupt to wake up the system to provide sometime for the system to do the preparation for the running of BLE. The ahead of time is programmable in the BLE MAC and loaded to the sleep timer together with the wakeup time. The wakeup time is a 32 bits register and the maximum sleep duration will be more than 37 hours with the 32KHz sleep clock.

16.2Clock and reset

The clock of BLE MAC low power controller is clk_32k_ble_lp
The reset of BLE MAC low power controller is rst_ble_lp_32k_n
The clock gating control register of BLE MAC low power controller is:

sysc_awo	0x20201000	0x004	[1]	clkg_ble_lp_clr
sysc_awo	0x20201000	0x004	[0]	clkg_ble_lp_set

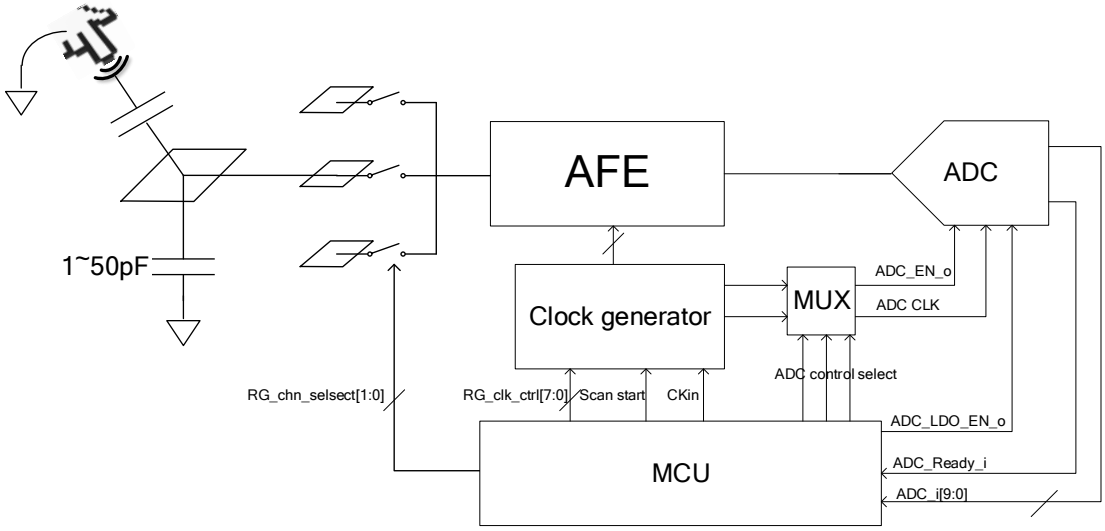
The software reset register of BLE MAC low power controller:

sysc_awo	0x20201000	0x040	[3]	srst_ble_lp_n_clr
sysc_awo	0x20201000	0x040	[2]	srst_ble_lp_n_set

17 Touch key function

17.1General description

BX2400 offers an ultra-low power solution of self-capacitance controller applications where an awake/activate on proximity/touch function is required. This feature is realized with integrated 6-bit ADC in BX2400. One ADC channel will be occupied if one touch function is required. One touch function is always available even in sleep mode.



The touch controller controls the analog touch module, captures the touch ADC output and generates the touch interrupt which is used to wake up the system when system is in deep sleep or extended sleep mode. The touch controller is composed of three parts, the clock divider, the time counter and the ADC controller.

17.2Clock divider

The analog touch module needs a clock which frequency is programmable. The touch controller provides the clock based on the 32MHz clock. The clock divide parameter is controlled by the following register:

0x20201078[17:14]	clk_div_sel_touch	the output clock frequency is 32MHz/n and the relationship between clk_div_sel_touch and n is as below: when clk_div_sel_touch is 0 then n is 1 when clk_div_sel_touch is 1 then n is 2

		when clk_div_sel_touch is 2 then n is 3 when clk_div_sel_touch is 3 then n is 4 when clk_div_sel_touch is 4 then n is 5 when clk_div_sel_touch is 5 then n is 6 when clk_div_sel_touch is 6 then n is 7 when clk_div_sel_touch is 7 then n is 8 when clk_div_sel_touch is 8 then n is 9 when clk_div_sel_touch is 9 then n is 10 when clk_div_sel_touch is 10 then n is 12 when clk_div_sel_touch is 11 then n is 14 when clk_div_sel_touch is 12 then n is 18 when clk_div_sel_touch is 13 then n is 24 when clk_div_sel_touch is 14 then n is 36 when clk_div_sel_touch is 15 then n is 60

17.3Touch interval counter

The touch can be programmed to do the touch detection repeatly. And the touch controller can work even if the system is in deep sleep state or extended sleep state. The time interval between two continuous detections is programmable which is controlled by the following register.

0x20201064[15:0]	touch_itv	The time interval between two continuous touch detections in 32KHz clock.

17.4Touch ADC controller

For each time of detection of touch, the touch controller power up the normal LDO and switch from the sleep LDO to the normal LDO and power up the 32MHz RC oscillator first, wait for enough time until the power of normal LDO is stable and the 32MHz clock is stable. Then start to sample the value of the touch ADC. The touch controller can be programmed to do more than one times of touch ADC sample. The value of all of the touch ADC samples of one touch detection are accumulated and compared with the preprogrammed threshold. If the accumulated value is bigger than the preprogrammed threshold, then the touch interrupt will be triggered. If the system is in deep sleep or extended sleep state, then the system will wakes up. The related registers are listed below:

0x20201064[20]	touch_en	enables the touch controller
0x20201064[19:16]	rcosc_16m_dly	the delay for the stable of the 32MHz RC oscillator in 32KHz clock
0x20201068[15:8]	touch_scan_num	the times of sample of the touch ADC for one time of

		touch detection
0x20201068[7:0]	touch_scan_dly	the delay between two continuous touch ADC sampling in 32MHz clock
0x2020106c[17:0]	touch_thr	the threshold of triggering the touch interrupt
0x20201028[13]	touch_intr_raw	the raw interrupt value of touch interrupt
0x20201028[5]	touch_intr_value	the masked interrupt value
0x2020102c[5]	touch_intr_clr	touch interrupt clear
0x20201024[5]	touch_intr_en	mask for the touch interrupt, low means the touch interrupt is masked
0x20201014[19:16]	ldo_swc_dly	the latency of the power supply switching between the normal LDO and the sleep LDO in 32KHz clock
0x20201014[4:0]	ldo_stb_dly	the latency of the powering up time of the DCDC and normal LDO in 32KHz clock.

17.5Clock and reset

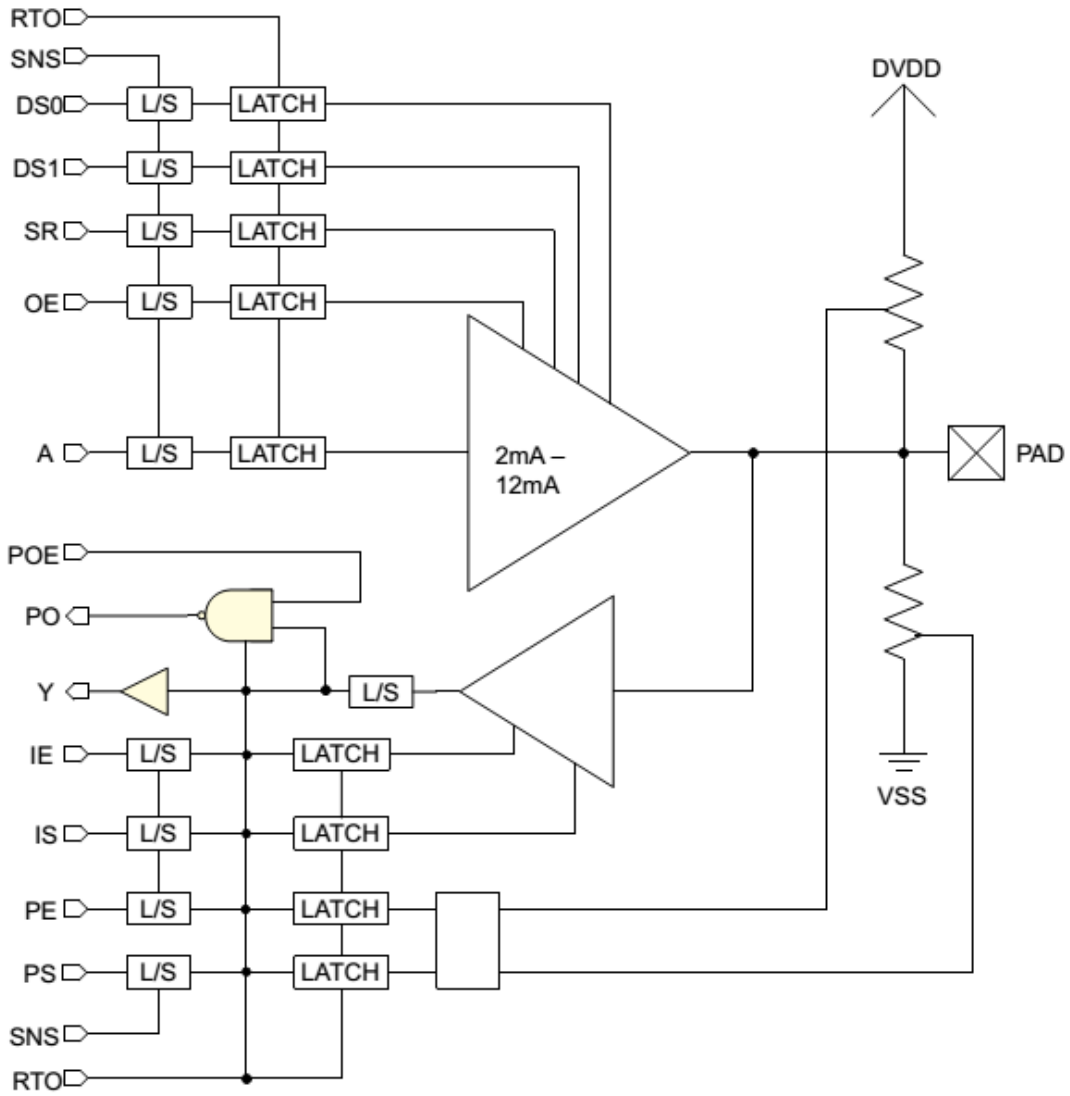
The clock of touch controller is clk_src_32k and clk_src_32m. (refer to Chapter 7)
The reset of touch controller is rst_touch_32k_n and rst_touch_16m_n. (refer to Chapter 8)
There is no clock gating for touch clock.
The software reset register for touch controller is:

sysc_awo	0x20201000	0x040	[9]	srst_touch_n_clr
sysc_awo	0x20201000	0x040	[8]	srst_touch_n_set

18 PAD ring

18.1General description

BX2400 has 30 general GPIOs(P0 to P29), one IO for external reset and one IO for test mode. Each of the 30 GPIOS has multiple functions which are defined by the pin share logic. The IO pads for the general GPIOs are all bi-directional IO PADs and have many programmable features. The block diagram of the GPIOs is shown below:



The port description of the PAD cell is shown below:

Pin name	Pin direction	Function	Voltage level	Description
OE	I	Output enable	Core	Logic HIGH enables the output buffer.
A	I	Driver data input	Core	Buffered data from the core to the bond-pad.
IE	I	Input enable	Core	Logic HIGH enables the input buffer.
IS	I	Input select	Core	Logic LOW selects CMOS input, and logic HIGH selects Schmit input.
Y	O	Receiver data output	Core	Buffered data from PAD to the core logic.
PE	I	Pull enable	Core	Logic HIGH enables weak pull device.
PS	I	Pull select	Core	Logic HIGH selects pull-up, logic LOW selected pull-down.
PAD	I/O	Pad	I/O	External PAD.
SNS	I	Sense input pin	I/O	Logic LOW keeps output driver in tri-state. Logic HIGH enables the output driver for normal operation. Voltage level must be same as DVDD.
SR	I	Slew rate	Core	Logic HIGH selects slow slew rate, logic LOW selects fast slew rate.
DS0	I	Drive select 0	Core	Used to select output drive strength.
DS1	I	Drive select 1	Core	Used to select output drive strength.
PO	O	Parametric output	Core	Parametric inverted data for PAD to core logic.
POE	I	Parametric output enable	Core	Logic LOW forces PO HIGH.
RTO	I	Retention enable	I/O	Logic LOW retains the value of output driver, logic HIGH enables output driver to follow pin A. Voltage level is same as DVDD.
VDD	I/O	Core power supply	Core	Core power routed through the I/O ring and to the core.
VSS	I/O	Core ground	Ground	Ground routed through the I/O ring and to the core.
DVDD	I/O	I/O power supply	I/O	I/O power supply.
DVSS	I/O	Output driver ground	Ground	Output driver ground.

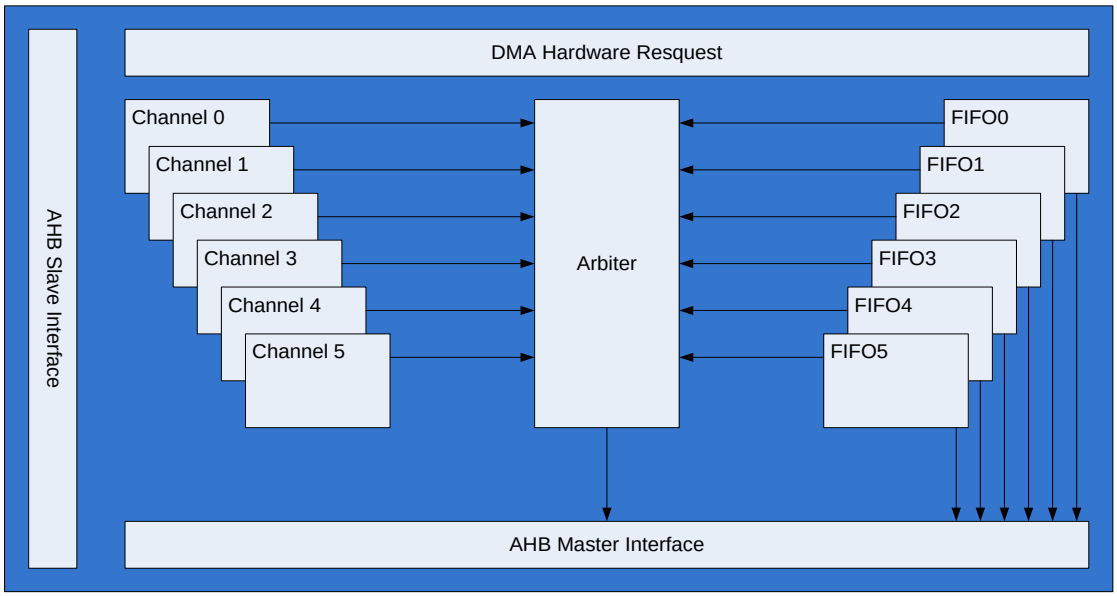
The related registers are listed below:

0x2020104c[29:0]	o_gpio_ds0	controls the DS0 of the 30 GPIOs, bit 0 represents to DS0 of P0 and bit 29 represents to DS0 of P29
0x20201050[29:0]	o_gpio_ds1	controls the DS1 of the 30 GPIOs, bit 0 represents to DS1 of P0 and bit 29 represents to DS1 of P29
0x20201054[29:0]	o_gpio_ie	controls the IE of the 30 GPIOs, bit 0 represents to IE of P0 and bit 29 represents to IE of P29
0x20201058[29:0]	o_gpio_is	controls the IS of the 30 GPIOs, bit 0 represents to IS of P0 and bit 29 represents to IS of P29
0x2020105c[29:0]	o_gpio_pe	controls the PE of the 30 GPIOs, bit 0 represents to PE of P0 and bit 29 represents to PE of P29
0x20201060[29:0]	o_gpio_ps	controls the PS of the 30 GPIOs, bit 0 represents to PS of P0 and bit 29 represents to PS of P29

19 DMA controller (DMAC)

19.1General description

The DMA controller has 6 programmable direct memory access channels for fast data transfers from/to peripheral interfaces and system SRAMs. The block diagram of DMAC is shown below:



The features are listed below:

- 6 DMA channels
- Flow control by DMAC
- Programmable channel priority
- Programmable DMA transfer length
- Programmable source and destination for each channel
- Support for memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral DMA transfers

19.2DMA Hardware Request

The DMAC supports 16 pairs of hardware DMA request and acknowledge signals:

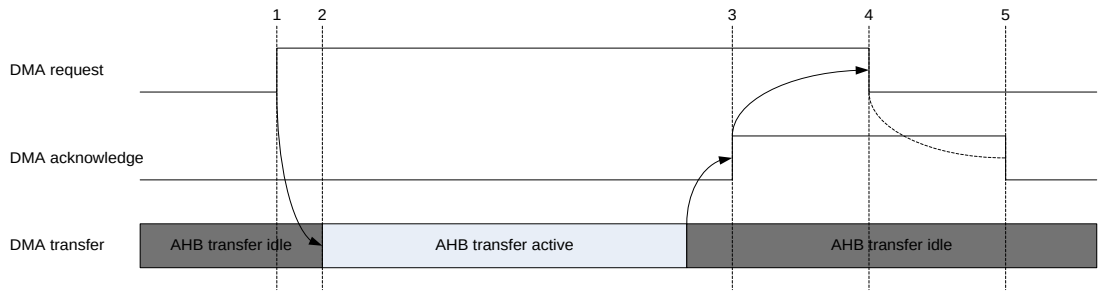
index	DMA hardware request	Option 2
0	QSPI TX	
1	QSPI RX	

index	DMA hardware request	Option 2
2	SPIM0 TX	
3	SPIM0 RX	
4	SPIM1 TX	
5	SPIM1 RX	
6	SPIS TX	
7	SPIS RX	
8	UART0 TX	
9	UART0 RX	
10	UART1 TX	
11	UART1 RX	
12	IIC0 TX	
13	IIC0 RX	
14	IIC1 TX	
15	IIC1 RX	ADC controller

The DMA request and acknowledge pair of the ADC controller is shared with the DMA request and acknowledge pair of the IIC1 RX. The sharing is controlled by the register below:

0x20132040[1]	adc_has_dma	High means DMA request and acknowledge pair of the ADC controller is muxed to the 15th DMA request and acknowledge pair of DMAC. Low means DMA request and acknowledge pair of the IIC1 RX is muxed to the 15th DMA request and acknowledge pair of DMAC.

The waveform for the DMA hardware request and acknowledge signal pair is shown below:



1) The peripheral module pulls up the DMA request to inform the DMA controller to start the next DMA transfer.
2) The DMA controller gets the DMA request and do the DMA transfer. The data size and number of the DMA transfer are predefined.
3) The DMA transfer has finished, then the DMA controller pulls up the DMA request to

inform the peripheral module that current DMA transfer has finished.

4) The peripheral module gets the DMA acknowledgement, and pulls down the DMA request.
5) The DMA controller monitor that the DMA request has been cleared, then pulls down the DMA acknowledgement.

19.3Interrupts

The DMAC has 6 DMA channels and each channel has 5 interrupts:

- Intr_err: This interrupt is generated when an ERROR response is received from the AHB slave on the HRESP bus during a DMA transfer. The error response will finish the current DMA transfer and also disable the current DMA channel.
- Intr_src_tran: This interrupt is generated when the last AHB transfer from the source module has finished.
- Intr_dst_tran: This interrupt is generated when the last AHB transfer to the destination module has finished.
- Intr_tran: This interrupt is generated on DMA transfer completion to the destination peripheral.
- Intr_block: This interrupt is generated on DMA block transfer completion to the destination peripheral.

19.4Clock and reset

The clock of DMAC is clk_hbus_dmac(refer to Chapter 7)
The reset of DMAC is rst_dmac_hbus_n(refer to chapter 8)
The clock gating control register of DMAC is:

sysc_cpu	0x20132000	0x014	[7]	clkg_clr_dmac
sysc_cpu	0x20132000	0x014	[6]	clkg_set_dmac

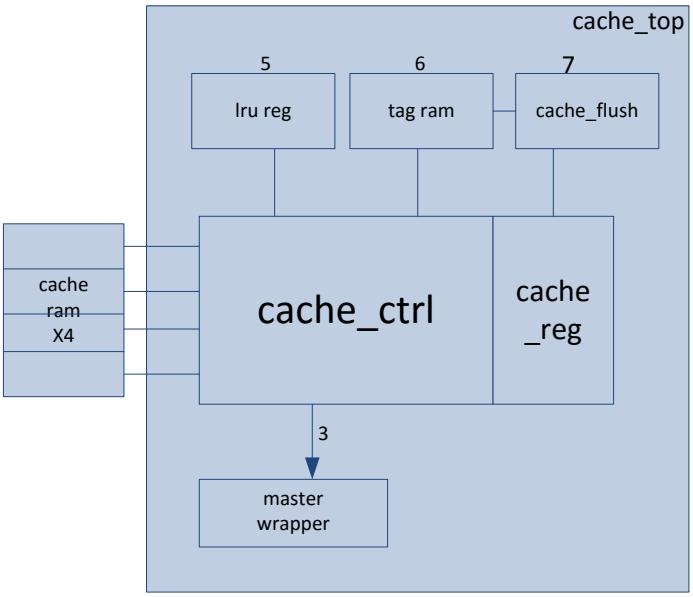
The software reset register of DMAC is:

sysc_cpu	0x20132000	0x040	[5]	srst_clr_dmac_n
sysc_cpu	0x20132000	0x040	[4]	srst_set_dmac_n

20 Cache controller

20.1General description

The cache controller is used to improve the system performance when CPU is executing on the external SPI flash and to reduce the power consumption by reducing the access to the external SPI flash. The cache controller is a read only cache controller which means that CPU can only read data and code from the cache controller. Any write to the cache controller will be ignored by the cache controller and has no effect on the SPI flash. The block diagram of the cache controller is shown below:



The data RAM size of the cache controller is 16KB and can be shared with the system SRAM if the cache controller is not enabled. The cachable address space is 8MB. The LRU algorithm is implemented in the cache controller to update the cache data. An AHB master wrapper is implemented to convert the cache data access to the SPI command of the QSPI controller.

20.2Feature List

- Zero wait cycle when cache hit
- 8MB cachable address space with the start address programmable

- 16KB cache data size which is shared with the system SRAM
- Supports cache flush command
- 4 Way cache with LRU algorithm
- Read only cache
- 32 bytes cache line size

20.3Least Recently Used (LRU) algorithm

The least recently used (LRU) algorithm is used to update the cache data when cache missing. The cache is organized by 4 ways. The relationship between the 4 ways can be represented by 6 bits.
Bit 0: high means way 0 is used more recently than way 1.
Bit 1: high means way 0 is used more recently than way 2.
Bit 2: high means way 0 is used more recently than way 3.
Bit 3: high means way 1 is used more recently than way 2.
Bit 4: high means way 1 is used more recently than way 3.
Bit 5: high means way 2 is used more recently than way 3.

When cache missing, a new cache line data is read from the external SPI flash, then the least recently used way can be found according to the above 6 bits, and will be replaced by the new cache line data.
When bit [0,1,2] are low, then way 0 is least recently used.
When bit [0] is high and bit [4,5] are low, then way 1 is least recently used.
When bit [1,3] are high and bit [5] is low, then way 2 is least recently used.
When bit [2,4,5] are high, then way 3 is least recently used.
Then the above 6 bits are updated because the updated way is now the most recently used way.
When way 0 is replaced, then bit [0,1,2] are set to high.
When way 1 is replaced, then bit [0] is set to low and bit [4,5] are set to high.
When way 2 is replaced, then bit [1,3] are set to low and bit [6] is set to high.
When way 3 is replaced, then bit [2,4,5] are set to low.

20.4Control registers

The control registers are listed below:

module name	base address	address offset	bits	register name
sysc_cpu	0x20132000	0x040	[0]	cache_has_sram

module name	base address	address offset	bits	register name
cache	0x20121000	0x000	[0]	cache_en
cache	0x20121000	0x004	[31:0]	flash_base_addr
cache	0x20121000	0x008	[31:0]	qspi_dr_addr
cache	0x20121000	0x00C	[10]	flash_rd_cmd_format
cache	0x20121000	0x00C	[9]	flash_rd_data_format
cache	0x20121000	0x00C	[8]	flash_rd_addr_format
cache	0x20121000	0x00C	[7:0]	flash_rd_cmd
cache	0x20121000	0x010	[31:0]	flush_addr_l
cache	0x20121000	0x014	[31:0]	flush_addr_h
cache	0x20121000	0x018	[0]	flush_all
cache	0x20121000	0x01C	[0]	flush_en

- cache_has_sram: This bit must be set to high if cache is enabled. This bit controls the mux to the data RAM of the cache. High means the cache controls the accessing to the data RAM. Low means the data RAM of the cache is used as the system SRAM.
- cache_en: enable the cache controller, high active.
- flash_base_addr: this is the base address of the cacheable address space
- qspi_dr_addr: this is the data FIFO address of QSPI controller. The cache controller read data from this address of the QSPI controller if the data is ready to ready in the QSPI controller.
- flash_rd_cmd_format: defines the position of SPI command in a 32 bit word. Low means the command is at the LSB. High means the command is at the MSB.
- flash_rd_data_format: defines the data format of SPI. Low means the data is little endian and high means big endian.
- flash_rd_addr_format: defines the position of SPI flash address in a 32 bit word. Low means the command is at the LSB. High means the command is at the MSB.
- flush_addr_l: start address of the address space to be flushed.
- flush_addr_h: end address of the address space to be flushed.
- flush_all: flush all of the tag memory of the cache controller.
- flush_en: start to flush the tag memory of cache controller. If flush_all is high, then flush all of the tag memory. If flush_all is low, then flush the tag memory entry which address is in the address range defined by flush_addr_l and flush_addr_h.

Note: after reset of the system, the tag memory of the cache is invalid and must be flushed by software before accessing the cacheable address space.

20.5Clock and reset

The clock of cache controller is clk_hbus_cache (refer to chapter 7)

The reset of cache controller is rst_cache_hbus_n (refer to chapter 8)

The clock gating control register of cache controller is:

sysc_cpu	0x20132000	0x014	[9]	clkg_clr_cache
sysc_cpu	0x20132000	0x014	[8]	clkg_set_cache

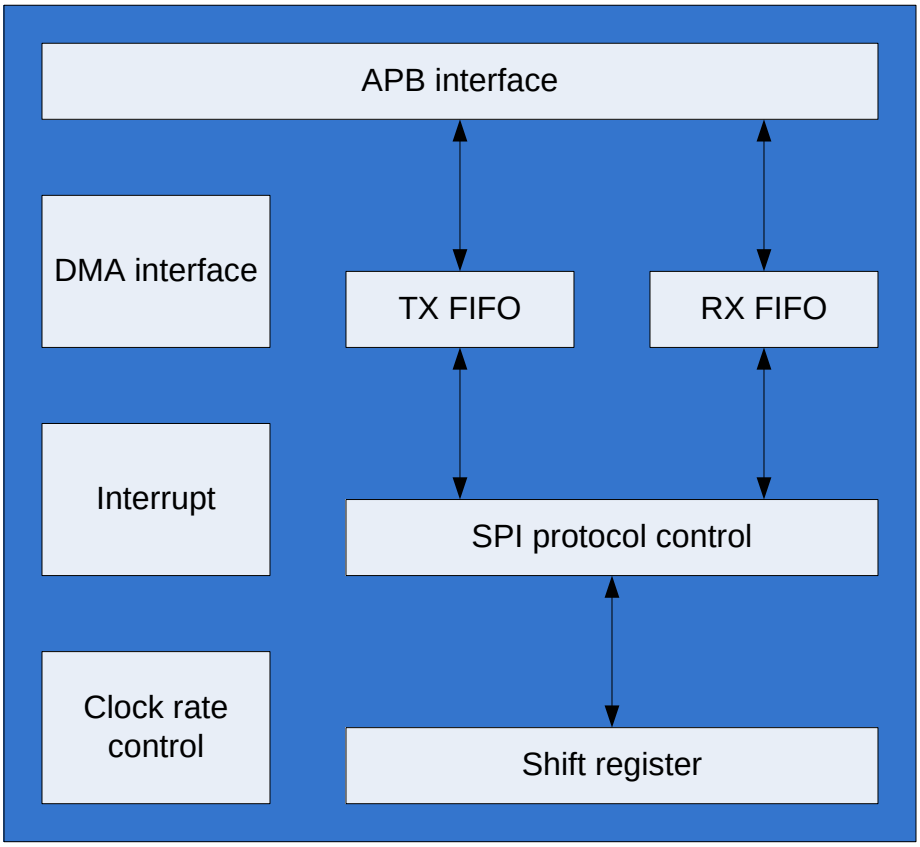
The software reset register of cache controller is:

sysc_cpu	0x20132000	0x040	[7]	srst_set_cache_n
sysc_cpu	0x20132000	0x040	[6]	srst_set_cache_n

21 Quard-SPI controller

21.1General description

The quard-SPI controller is a SPI master interface controller which supports 4 wires SPI protocol with 4 data lines. The quard-SPI controller can also work under 2 wires mode or 1wire mode. The quard-SPI is compliant with Motorola Serial Peripheral Interface protocol. The SPI interface clock rate is programmable with the maximum frequency of 24MHz, and hence the maximum throughput is 96Mbps. The block diagram of quard-SPI controller is shown below:



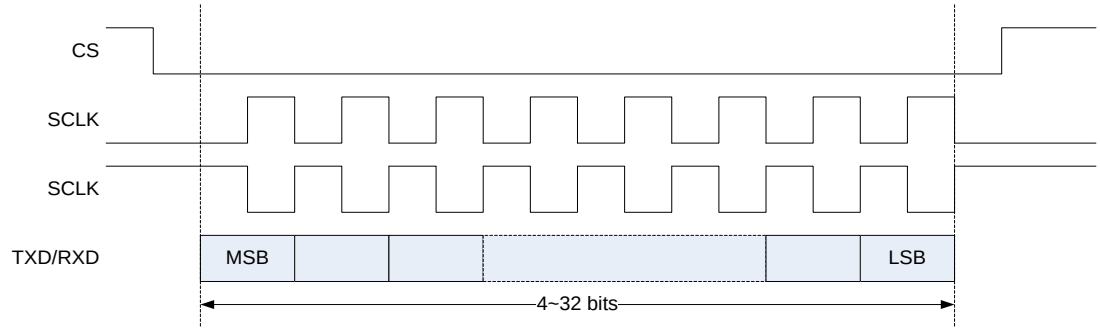
21.2Feature list

- APB interface with 32 bit data bus
- Supports 4 wires, 2 wires and 1wire mode
- Supports DMA interface
- 64 words RX and TX data FIFO

- Programmable SPI interface clock, maximum frequency is 24MHz
- Compliant with Motorola SPI interface
- Programmable SPI data size: 4 bits to 32 bits
- SPI mode programmable(phase and clock edge)
- Only one chip select output
- Programmable RX data sample edge
- Read data byte revert when in 32 bits frame size

21.3Read data byte revert

The data frame of the SPI interface is shown below:



The quard-SPI controller supports 32 bits data frame. If working in 32 bits mode, continuous 32 bits data is combined together to be a 32 bits word with the first received bit on the highest bit of the word and then be passed to the bus. But most of the flashes are working under the 8 bits mode which means that the first received 8 bits need to be put at the lowest 8 bits of a 32 bit word. Then If reading data from the SPI flash and the quard-SPI controller is working under 32 bits mode to increase the utilization of the data FIFO of quard-SPI controller and also increase the efficiency of the bus transfers, the data to the bus need to be byte reverted. The quard-SPI controller supports read data byte reversion to support the above working scenario. The data FIFO can be accessed by the bus through a range of address which is from 0x20300060 to 0x203000EC. If accessing to the address 0x203000EC, then the read data is byte reverted.

21.4Interface

The interface signals are described below:

Signal name	1 wire mode	2 wires mode	4 wires mode
qspi_cs_n	Chip select, low	Chip select, low active	Chip select, low active

Signal name	1 wire mode	2 wires mode	4 wires mode
	active		
qspi_clk	Serial clock	Serial clock	Serial clock
qspi_dat0	Data output	Bidirectional Data wire 0	Bidirectional Data wire 0
qspi_dat1	Data input	Bidirectional Data wire 1	Bidirectional Data wire 1
qspi_dat2			Bidirectional Data wire 2
qspi_dat3			Bidirectional Data wire 3

21.5Interrupts

The quard-SPI controller has the following interrupts:

- intr_tx_fifo_empt: Set when the TX FIFO fill level is equal to or below the preprogrammed threshold register. This interrupt is cleared automatically when the TX FIFO fill level is bigger than the threshold.
- intr_tx_fifo_over: Set when the TX FIFO is over run.
- intr_rx_fifo_full: Set when the RX FIFO fill level is above the preprogrammed threshold register. This interrupt is cleared automatically when the RX FIFO fill level is equal to or less than the threshold.
- intr_rx_fifo_over: Set when the RX FIFO is over run.
- intr_rx_fifo_under: Set when the RX FIFO is under run.

21.6Clock and reset

The clock of QSPI is clk_hbus_qspi(refer to chapter 7)
The reset of QSPI is rst_qspi_hbus_n(refer to chapter 8)
The clock gating control register of QSPI is:

sysc_cpu	0x20132000	0x014	[11]	clkg_clr_qspi
sysc_cpu	0x20132000	0x014	[10]	clkg_set_qspi

The software reset control register of QSPI is:

sysc_cpu	0x20132000	0x040	[13]	srst_clr_qspi_n
sysc_cpu	0x20132000	0x040	[12]	srst_set_qspi_n

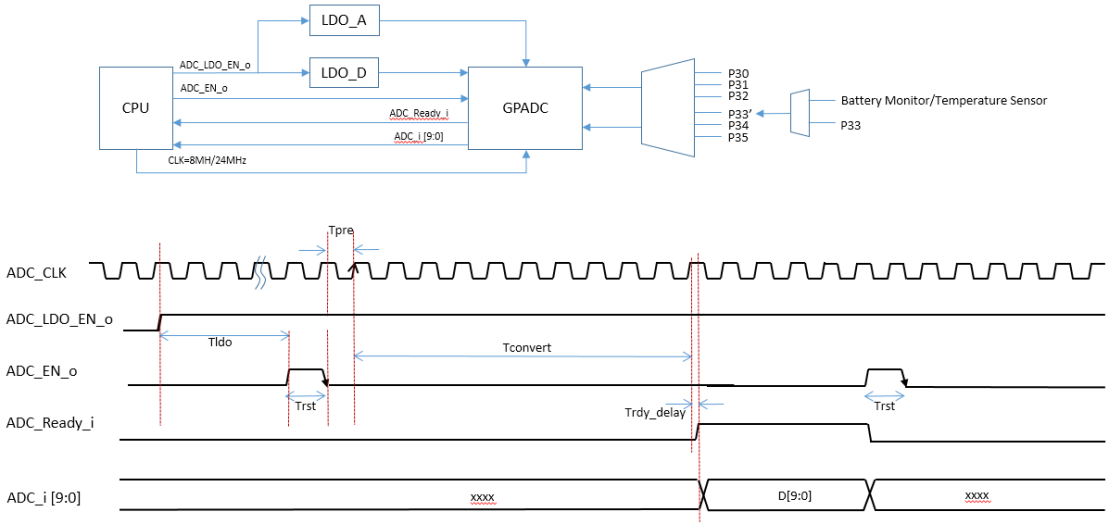
22 ADC

22.1 General description

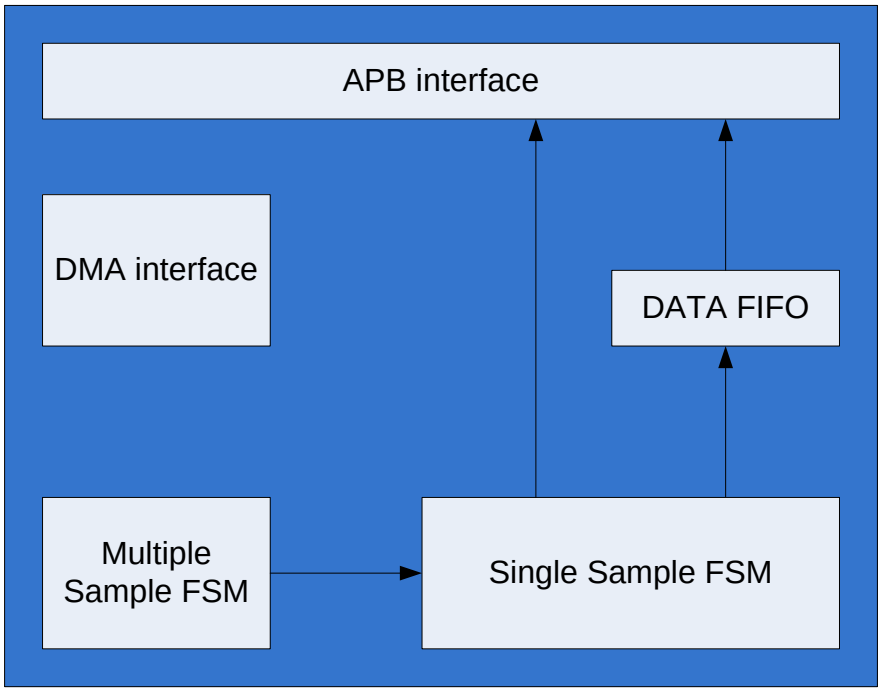
The BX2400 is equipped with a low power 10-bit general purpose Analog-to-Digital Converter (GPADC). It can operate in unipolar (single ended) mode as well as in bipolar (differential) mode. The ADC has its own voltage regulators (LDO) of 3V. The full scale reference voltage of GPADC is optional set from 2.2V to 2.8V.

Features

- 10-bit dynamic ADC with average capability
- Maximum sampling rate 2 Msample/s at 96MHz ADC clock
- Single-ended as well as differential input with two input scales
- Six single-ended or two differential external input channels
- Oversampling up to 64 steps providing effectively up to 12 bits resolution
- Support battery monitoring function from 2.0V to 5.5V
- Support temperature Sensing function from -40 to 125 degree
- DMA support



The ADC controller controls the analog module multiple channel ADC, samples the ADC data and pass the data to the CPU through the APB bus interface. The block diagram of the ADC controller is shown below:



The ADC controller has two working modes: single sample mode and multiple samples mode. Under single sample mode, when the ADC controller is enabled, the ADC controller will sample one of the ADC channel once, save the sampled data in local register and then go back to idle state. Under multiple samples mode, the ADC controller will sample one of the ADC channel multiple times and save the sampled data in the local data FIFO and then go back to idle state. The time interval between multiple sampling is programmable and the ADC can be powered off during the time interval of multiply times of sampling. The single sample mode and multiple sample modes can be enabled simultaneously and if request to sample ADC at the same time by the two modes, the multiple samples mode has the higher priority. DMA accessing is supported under multiple samples mode and only single DMA requesting is supported. The data FIFO size is 16 samples.

22.1 Oversampling up to 12-bit

Oversampling is always implemented by multiple successive single samplings, being handled in software. Based on 10-bit resolution ADC, averaging following accumulation will realize true resolution up to 12-bit or more. Care should be taken that software computing will definitely consume MCU ticks, so the higher resolution means the lower sample rate.
It is recommended to set GPADC operate as differential sampling mode for better ENOB performance. Compare to the single-ended mode, the maximum sampling frequency of differential mode is 32k sampling per second (SPS) which is twice as the former when oversampling function is utilized for 12-bit resolution.

22.2 Control registers

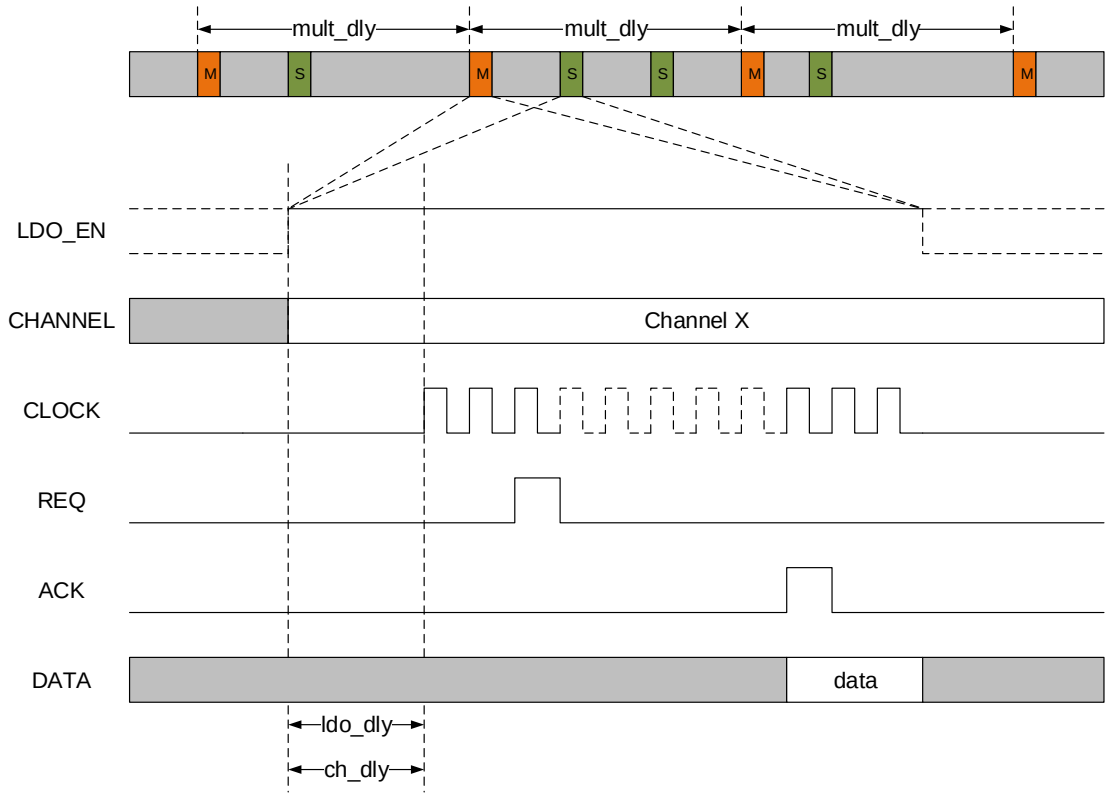
The Control registers of the ADC controller is listed below:

0x20136000[5]	dma_en	DMA enable, high active. High means the DMA hardware handshake signals will be driven by the ADC controller.
0x20136000[4]	adc_clk_pol	ADC clock polarity. High means the output clock to the analog module ADC will be inverted for timing consideration.
0x20136000[3]	ldo_sd_sngl	LDO shut down enable for single capture mode. High means the LDO of ADC will be shut down automatically after single capture to the ADC.
0x20136000[2]	ldo_sd_mult	LDO shut down enable for multiply capture mode. High means the LDO of ADC will be shut down automatically after each times of sampling of ADC data under multiply sampling mode.
0x20136000[1]	ldo_force_off	ADC LDO force off enable. If high, the LDO of ADC is forced off.
0x20136000[0]	ldo_force_on	ADC LDO force on enable. If high, the LDO of ADC is forced on.
0x20136004[31:16]	mult_dly	Time intervals in 32MHz clock between two continuous sampling to the ADC under multiply sampling mode.
0x20136004[15:0]	mult_num	the number of sampling to the ADC under multiply sampling mode.
0x20136008[25:16]	ch_dly	the time delay in 32MHz clock for the

		channel switching of the ADC.
0x20136008[9:0]	ldo_dly	the time delay in 32MHz clock for the stable of LDO of the ADC.
0x2013600c[25:16]	adc_data_sngl	the sampled data for single sampling mode
0x2013600c[14:12]	sngl_ch	the ADC channel index for single sampling mode
0x2013600c[3]	fifo_full	FIFO full flag, high active
0x2013600c[2]	fifo_empt	FIFO empty flag, high active
0x2013600c[1]	sngl_start	single sampling mode start, high active. Will be cleared to 0 after the ADC data is ready to read
0x20136010[9:0]	fifo data	FIFO data access point
0x20136014[10:8]	mult_ch	the ADC channel index for multiply sampling mode
0x20136014[0]	mult_start	multiply sampling mode start, high active. Will be cleared to 0 after all of the samplings finish.

22.3 Waveform of the ADC sampling

The waveform of the ADC sampling is shown below:



The rectangle in orange and marked with “M” is the sampling for multiply sampling mode. The rectangle in green and marked with “S” is the sampling for single sampling mode. LDO_EN is the signal which controls the on/off of the LDO of ADC. CHANNEL is the signal which controls the current channel index to be sampled. CLOCK is the clock for the ADC. REQ is the sampling request to the ADC. ACK is the acknowledgement from the ADC which indicates the DATA is ready to sample.

22.4Clock and reset

The clock of ADC controller is clk_pbus_sys. (refer to Chapter 7)
The reset of ADC controller is rst_sys_hbus_n. (refer to Chapter 8)
There are no clock gating logic and software reset for the ADC controller

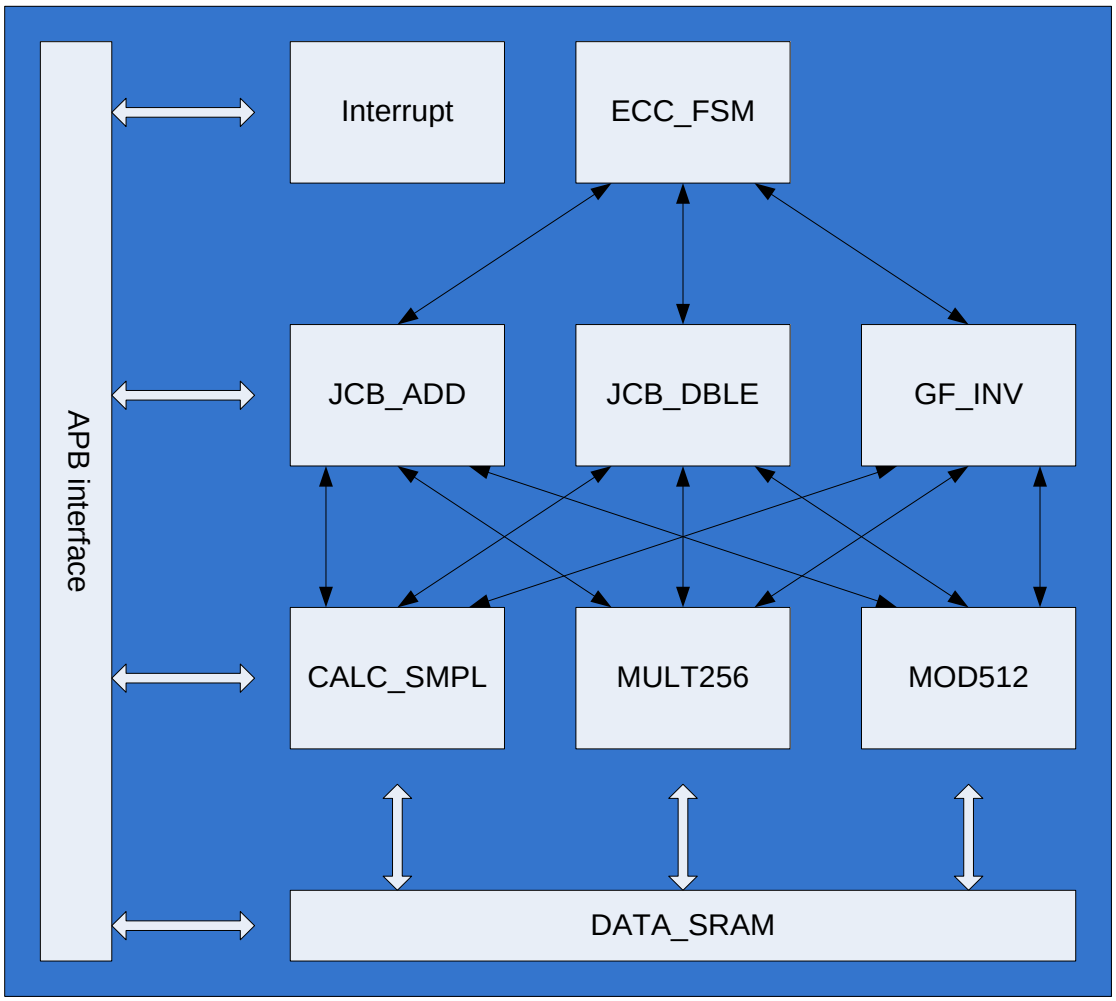
23 ECC

23.1General description

BX2400 supports ECC calculation by hardware. The cost of time of one time ECC calculation is much more less than software calculation. The ECC algorithm recommended by BLE is P-256. The ECC algorithm is defined on finite field and the mainly calculation in ECC algorithm is the addition and double on the finite field. The ECC calculation process is described below:

1) Generate a private key, 256 bits.
2) Get the public key: P, and set the infinite field point Q to be infinity point.
3) For each bit of the private key(lowest bit first), do the following calculation:
   a) If the bit is high, Q = P + Q
   b) P = 2P;
4) Calculate the public key

The block diagram of ECC is shown below:



The ECC calculation flow is controlled by ECC_FSM which is composed of Jacobian add and Jacobian double, and point reverse which are handled by modules named JCB_ADD, JCB_DBLE and GF_INV. The Jacobian add, Jacobian double and point reverse are composed of multiple times of base calculations including simple 256 bits addition/subtraction, 256 bits multiply and 512 bits mod handled by modules named CALC_SMPL, MULT256 and MOD512. The DATA_SRAM is used to save the public key, the temporary result and the ECC output. The size of DATA_SRAM is 512 Bytes which is organized in 16 256 bits data.

In the block diagram, each kind of calculation can be enabled independently. And each kind of calculation will trigger the interrupt after finished.

- The module named CALC_SMPL is in charge of simple calculations for 256 bits data. The calculations include multiply by 2/3/4/8 and add or sub between two 256 bits data.
- The module named MULT256 is in charge of the multiply of two 256 bits data and the result is 512 bits data.
- The module named MOD512 is in charge of the mod of one 512 bits data which is the result of the MULT256.
- The module named JCB_ADD is in charge of Jacobian add.

- The module named JCB_DBLE is in charge of Jacobian sub.
- The module named GF_INV is in charge of point reverse of one 256 bits data on the finite field.
- The module named ECC_FSM in charge of the process control of the ECC calculation.

23.2Interrupts

The interrupt of ECC is described below:

- intr_ecc: set when the current calculation started by software is finished.

23.3Data SRAM address mapping

The size of DATA_SRAM is 512 Bytes and organized in 32 bits word. 8 32 bits words composed of one 256 bits data. There are 16 256 bits data which can be saved in the DATA SRAM. The address mapping of the DATA_SRAM is shown below:

Address	Before ECC	After ECC
0	Low 256 bits of the result of 256 bits multiply	Low 256 bits of the result of 256 bits multiply
1	High 256 bits of the result of 256 bits multiply	High 256 bits of the result of 256 bits multiply
2	X coordinate of public key	
3	Y coordinate of public key	
4	Constant 1	
5	Constant 0	
6	Constant 1	
7	Constant 0	
8		
9		
10		
11		
12		X coordinate of public key
13		Y coordinate of public key
14		
15	Private key	

23.4Clock and reset

The clock of ECC is: clk_pbus_ecc. (refer to chapter 7)

The reset of ECC is rst_ecc_pbus_n(refer to chapter 8)

The clock gating control register is:

sysc_cpu	0x20132000	0x014	[19]	clkg_clr_ecc
sysc_cpu	0x20132000	0x014	[18]	clkg_set_ecc

The software reset register is:

sysc_cpu	0x20132000	0x040	[23]	srst_clr_ecc_n
sysc_cpu	0x20132000	0x040	[22]	srst_set_ecc_n

24 Timer

24.1General description

BX2400 integrates 2 independent 24 bits timer which share one APB bus interface. Both of the timers are counting in the same clock (clk_timer in chapter 7) which is divided from the 32MHz clock. Each timer counts down from a preprogrammed value and generates an interrupt when the count reaches zero. After the count reaches zero, the count will load the preprogrammed value again and starts to count down.

24.2Clock and reset

The clock of timer is clk_pbus_timer, clk_timer1 and clk_timer2(refer to chapter 8)
The reset of timer is rst_timer_pbus_n and rst_timer_n(refer to chapter 9)
The clock gating control register is:

sysc_cpu	0x20132000	0x010	[7]	clkg_clr_timer_div
sysc_cpu	0x20132000	0x010	[6]	clkg_set_timer_div
sysc_cpu	0x20132000	0x014	[15]	clkg_clr_timer1
sysc_cpu	0x20132000	0x014	[14]	clkg_set_timer1
sysc_cpu	0x20132000	0x014	[13]	clkg_clr_timer0
sysc_cpu	0x20132000	0x014	[12]	clkg_set_timer0

The software reset register is:

sysc_cpu	0x20132000	0x040	[11]	srst_clr_timer_n
sysc_cpu	0x20132000	0x040	[10]	srst_set_timer_n

25 Watch dog timer

25.1General description

The watch dog timer is an ABP slave which can be used to prevent the system lockup. If the system dose not acknowledge the watch dog interrupt or does not kick the watch dog in preprogrammed time, the watch dog timer will reset the system. The watch dog timer can work in two modes:
Mode1: the watch dog timer will reset the system directly if the system does no kick the watch dog timer in preprogrammed time.
Mode2: the watch dog timer will interrupt the system for the first time the timer reaches the maximum value and the watch dog timer will reset the system for the second time the timer reaches the maximum value.
The width of the counter is 16 bits and the maximum value is 0xFFFF.

25.2Clock and reset

The clock of the watch dog timer is clk_pbus_wdt and clk_32k_wdt(refer to chapter 7)
The reset of the watch dog timer is rst_wdt_pbus_n and rst_wdt_32k_n(refer to chapter 8)
The clock gating control register is:

sysc_cpu	0x20132000	0x014	[5]	clkg_clr_wdt
sysc_cpu	0x20132000	0x014	[4]	clkg_set_wdt

The software reset register is:

sysc_cpu	0x20132000	0x040	[9]	srst_set_wdt_n
sysc_cpu	0x20132000	0x040	[8]	srst_clr_wdt_n

26 Debug host

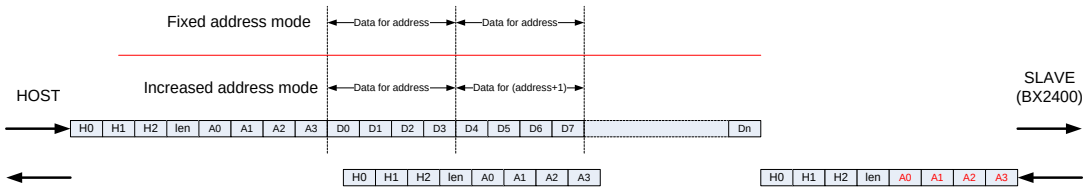
26.1General description

The debug host is an UART interface to AHB bus master bridge which enables the external UART host access the registers and SRAMs of BX2400 directly. The access can be done without the help of CPU. The baud rate of the UART interface is fixed at 921600bps. The access to the BX2400 address space must be done in 32 bits and the address is aliased with 4 bytes. The maximum data length of the accessing is 256 32 bits words. And the address of the target 32 bits words can be increased or fixed for multiple data access. There is an 8 32bits words FIFO implemented in the debug host to hold data from and to the external UART device.

26.2Frame format

The access to the BX2400 address space through the debug host must follow a predefined frame format.

26.2.1 Write frame format

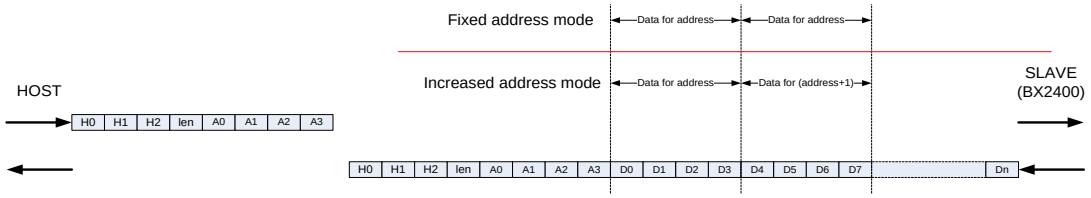


The above is the write access to the BX2400 address space.

A. Host sends 4 bytes header which includes 3 constant byte and one byte of transfer length to BX2400. H0 is 0xFF, H1 is 0xAA and H2 is 0x00.
B. Host sends 32 bits address to BX2400 and low byte of the address is send first. The high 30 bits of the address represents the address of the BX2400 in 32 bits word. And bit 0 of the address is fixed at 1 for write access. Bit 1 of the address represents the address mode of current access: 0 means fixed address accessing and 1 means increased address accessing.
C. Host send write data to BX2400 word by word and first byte in a word is send first. At the same time BX2400 will send the header and the start address received back to the host.

D. After all of the words have been received by BX2400, BX2400 will send the header and the end address back to the host. If current access is fixed address access, then the end address is the same as the start address. If current access is increased address access, then the end address is the sum of the start address and the length.

26.2.2 Read frame format



The above is the read access to the BX2400 address space.

A. Host sends 4 bytes header which includes 3 constant byte and one byte of transfer length to BX2400. H0 is 0xFF, H1 is 0xAA and H2 is 0x00.

B. Host sends 32 bits address to BX2400 and low byte of the address is send first. The high 30 bits of the address represents the address of the BX2400 in 32 bits word. And bit 0 of the address is fixed at 0 for read access. Bit 1 of the address represents the address mode of current access: 0 means fixed address accessing and 1 means increased address accessing.

C. BX2400 sends the header and the start address received back to the host.

D. BX2400 sends the read data to the host word by word and the first byte of a word is send first.

26.3Clock and reset

The clock of debug host is clk_hbus_uart2ahb and clk_32m_uart2ahb(refer to chapter 7)

The reset of debug host is rst_uart2ahb_hbus_n and rst_uart2ahb_32m_n(refer to chapter 8)

The clock gating control register of debug host is:

sysc_cpu	0x20132000	0x014	[17]	clkg_clr_uart2ahb
sysc_cpu	0x20132000	0x014	[16]	clkg_set_uart2ahb

The software reset of debug host is:

sysc_cpu	0x20132000	0x040	[21]	srst_clr_uart2ahb_n
sysc_cpu	0x20132000	0x040	[20]	srst_set_uart2ahb_n

27 ROM patch

27.1General description

The ROM patch is used to patch the ROM data if there is something wrong in the unmodifiable ROM code. There are altogether 16 items implemented in the module which means that there are altogether 16 words can be patched in the ROM code. Each item has an enable register and an address register and a replaced word register. If the enable register is set to high and the access address from the bus matches the address register, then the ROM output data is replaced with the word register. The related registers are listed below:

0x20133000[16:2]	trap0	ROM address 0
0x20133004[16:2]	trap1	ROM address 1
0x20133008[16:2]	trap2	ROM address 2
0x2013300C[16:2]	trap3	ROM address 3
0x20133010[16:2]	trap4	ROM address 4
0x20133014[16:2]	trap5	ROM address 5
0x20133018[16:2]	trap6	ROM address 6
0x2013301C[16:2]	trap7	ROM address 7
0x20133020[16:2]	trap8	ROM address 8
0x20133024[16:2]	trap9	ROM address 9
0x20133028[16:2]	trapa	ROM address 10
0x2013302C[16:2]	trapb	ROM address 11
0x20133030[16:2]	trapc	ROM address 12
0x20133034[16:2]	trapd	ROM address 13
0x20133038[16:2]	trape	ROM address 14
0x2013303C[16:2]	trapf	ROM address 15
0x20133040[31:0]	out0	replaced word 0
0x20133044[31:0]	out1	replaced word 1
0x20133048[31:0]	out2	replaced word 2
0x2013304C[31:0]	out3	replaced word 3
0x20133050[31:0]	out4	replaced word 4
0x20133054[31:0]	out5	replaced word 5
0x20133058[31:0]	out6	replaced word 6
0x2013305C[31:0]	out7	replaced word 7
0x20133060[31:0]	out8	replaced word 8
0x20133064[31:0]	out9	replaced word 9

0x20133068[31:0]	outa	replaced word 10
0x2013306C[31:0]	outb	replaced word 11
0x20133070[31:0]	outc	replaced word 12
0x20133074[31:0]	outd	replaced word 13
0x20133078[31:0]	oute	replaced word 14
0x2013307C[31:0]	outf	replaced word 15
0x20133080[15:0]	trap_en	enable for the address and data. Each bit is for one pair of address and replaced word. High means the ROM data at the address will be replaced by the replaced word.

27.2Clock and reset

The clock of the ROM patch is: clk_pbus_sys and clk_hbus_sys(refer to chapter 8)
The reset of the ROM patch is: rst_sys_hbus_n (refer to chapter 9)
There are no clock gating control register and software reset for ROM patch.

28 32KHz clock calibration

28.1General description

Bx2400 has a crystal clock input which frequency is 32MHz and accurate is 20ppm. But the frequency of the 32KHz RC oscillator depends on the temperature and the voltage of the LDO. Software need to know the current frequency of the 32KHz RC oscillator. The calibration logic is used to calculate the accurate frequency of the 32KHz RC oscillator. There is a counter counts in 32MHz clock and will count for 64 32KHz clock cycle. The value of the counter can be accessed by software when the counter stops. Then the frequency of 32KHz RC oscillator is 32MHz/(count/64).

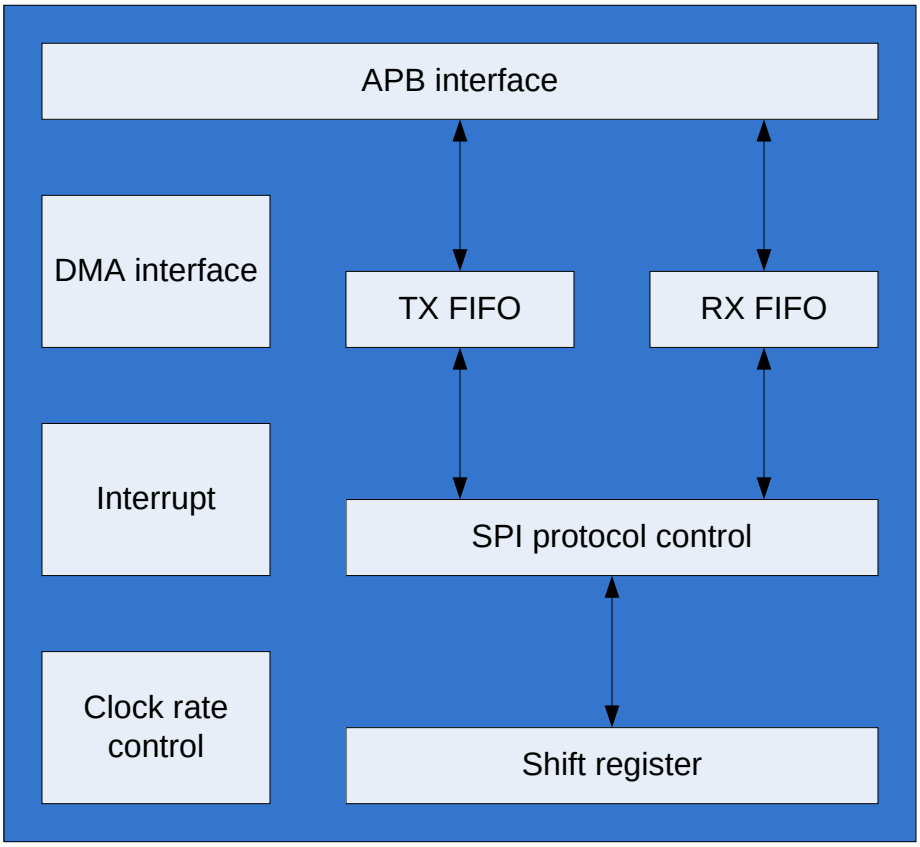
28.2Calibration accuracy

The maximum frequency of the 32KHz RC oscillator is 100KHz and the minimum frequency is 10KHz. Then the maximum cycle number of the 32MHz clock for 64 32KHz RC oscillator clock is 32MHz/10KHz*64 = 204800. And the minimum cycle number of the 32MHz clock for 64 32KHz RC oscillator clock is 32MHz/100KHz*64 = 20480. Because of the synchronization logic in the calibration logic, the accuracy of the counter is less than 2 cycles. So the accuracy of the calibration result is 2/20480 = 98ppm. Because the accuracy of the 32MHz clock is 20ppm, so the accuracy of the frequency of the 32KHz RC oscillator clock is less than 118ppm.

29 SPI master

29.1General description

BX2400 integrates two SPI master controllers: spim0 and spim1. The SPI master controller is a SPI master interface controller which supports standard SPI protocol with one transmit data line and one receive data line. The SPI is compliant with Motorola Serial Peripheral Interface protocol. The SPI interface clock rate is programmable with the maximum frequency of 24MHz, and hence the maximum throughput is 24Mbps. The block diagram of SPI master controller is shown below:



29.2Feature list

- APB interface with 32 bit data bus
- Supports 1wire mode, one wire of TX data and one wire of RX data
- Supports DMA interface
- 32 words RX and TX data FIFO

- Programmable SPI interface clock, maximum frequency is 24MHz
- Compliant with Motorola SPI interface
- Programmable SPI data size: 4 bits to 32 bits
- SPI mode programmable(phase and clock edge)
- Two chip select output
- Programmable RX data sample edge

29.3Interrupts

The SPI master controller has the following interrupts:

- intr_tx_fifo_empt: Set when the TX FIFO fill level is equal to or below the preprogrammed threshold register. This interrupt is cleared automatically when the TX FIFO fill level is bigger than the threshold.
- intr_tx_fifo_over: Set when the TX FIFO is over run.
- intr_rx_fifo_full: Set when the RX FIFO fill level is above the preprogrammed threshold register. This interrupt is cleared automatically when the RX FIFO fill level is equal to or less than the threshold.
- intr_rx_fifo_over: Set when the RX FIFO is over run.
- intr_rx_fifo_under: Set when the RX FIFO is under run.

29.4Clock and reset

The clock of the SPI master is clk_pbus_spim0 and clk_pbus_spim1(refer to chapter 8)

The reset of the SPI master is rst_spim0_pbus_n and rst_spim1_pbus_n(refer to chapter 9)

The clock gating control registers of SPI master controllers are:

sysc_per	0x20149000	0x010	[19]	clkg1_clr_spim0
sysc_per	0x20149000	0x010	[18]	clkg1_set_spim0
sysc_per	0x20149000	0x010	[17]	clkg0_clr_spim0
sysc_per	0x20149000	0x010	[16]	clkg0_set_spim0
sysc_per	0x20149000	0x010	[23]	clkg1_clr_spim1
sysc_per	0x20149000	0x010	[22]	clkg1_set_spim1
sysc_per	0x20149000	0x010	[21]	clkg0_clr_spim1
sysc_per	0x20149000	0x010	[20]	clkg0_set_spim1

The software reset registers of the SPI master controller are:

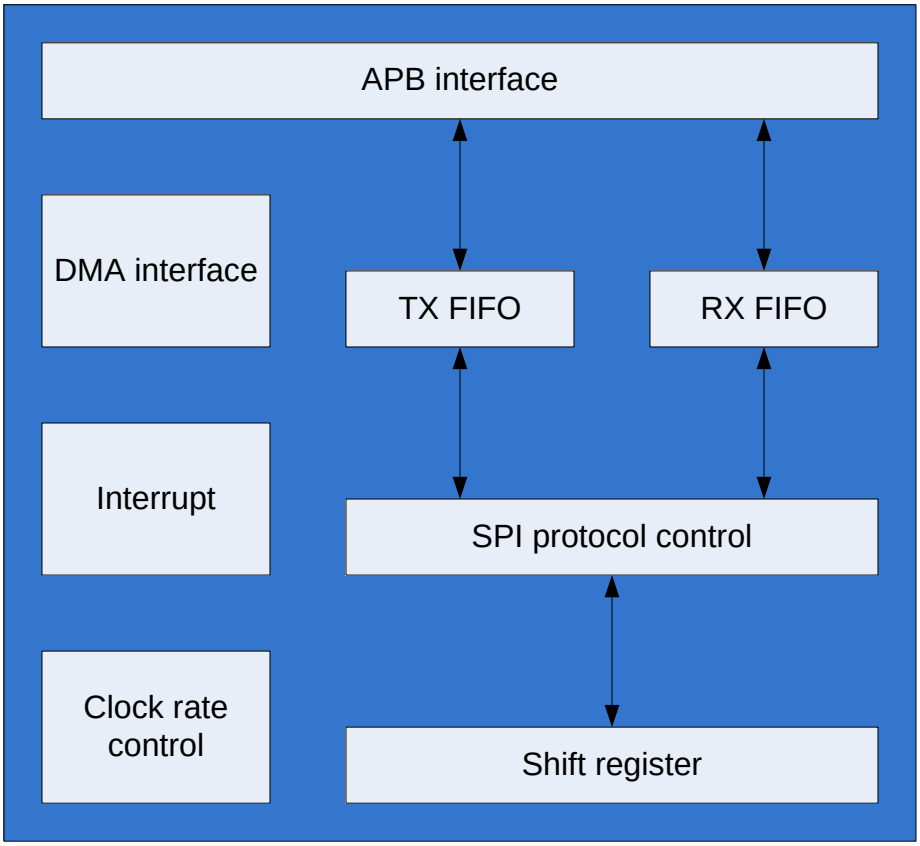
sysc_per	0x20149000	0x018	[11]	srst_clr_spim1_n
sysc_per	0x20149000	0x018	[10]	srst_set_spim1_n

sysc_per	0x20149000	0x018	[9]	srst_clr_spim0_n
sysc_per	0x20149000	0x018	[8]	srst_set_spim0_n

30 SPI slave

30.1General description

The SPI slave controller is a SPI slave interface controller which supports standard SPI protocol with one transmit data line and one receive data line. The SPI is compliant with Motorola Serial Peripheral Interface protocol. The SPI interface clock rate is programmable with the maximum frequency of 2MHz, and hence the maximum throughput is 2Mbps. The block diagram of SPI master controller is shown below:



30.2Feature list

- APB interface with 32 bit data bus
- Supports 1wire mode, one wire of TX data and one wire of RX data
- Supports DMA interface
- 32 words RX and TX data FIFO
- Programmable SPI interface clock, maximum frequency is 2MHz
- Compliant with Motorola SPI interface

- Programmable SPI data size: 4 bits to 32 bits
- SPI mode programmable(phase and clock edge)
- Programmable RX data sample edge

30.3Interrupts

The SPI slave controller has the following interrupts:

- intr_tx_fifo_empt: Set when the TX FIFO fill level is equal to or below the preprogrammed threshold register. This interrupt is cleared automatically when the TX FIFO fill level is bigger than the threshold.
- intr_tx_fifo_over: Set when the TX FIFO is over run.
- intr_rx_fifo_full: Set when the RX FIFO fill level is above the preprogrammed threshold register. This interrupt is cleared automatically when the RX FIFO fill level is equal to or less than the threshold.
- intr_rx_fifo_over: Set when the RX FIFO is over run.
- intr_rx_fifo_under: Set when the RX FIFO is under run.

30.4Clock and reset

The clock of the SPI slave controller is clk_pbus_spis(refer to chapter 8)
The reset of the SPI slave controller is rst_spis_pbus_n(refer to chapter 9)
The clock gating control register of SPI slave controller is:

sysc_per	0x20149000	0x010	[27]	clkg1_clr_spis
sysc_per	0x20149000	0x010	[26]	clkg1_set_spis
sysc_per	0x20149000	0x010	[25]	clkg0_clr_spis
sysc_per	0x20149000	0x010	[24]	clkg0_set_spis

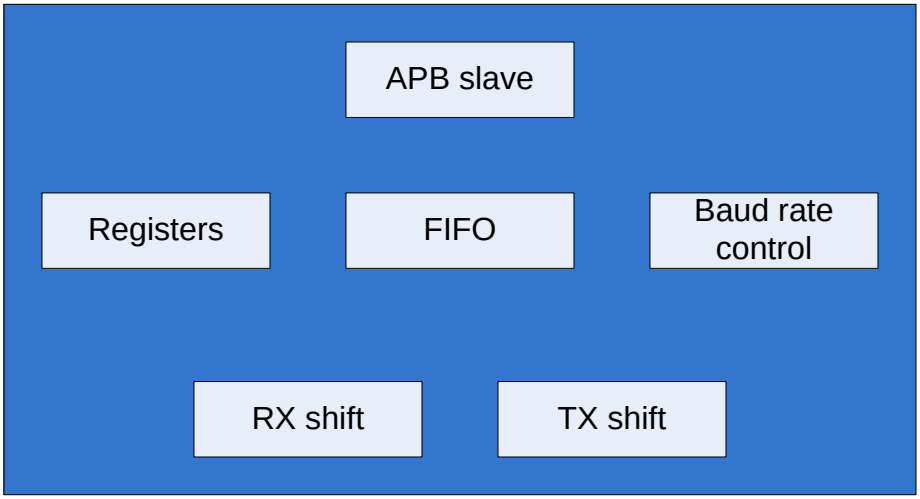
The software reset of SPI slave controller is:

sysc_per	0x20149000	0x018	[13]	srst_clr_spis_n
sysc_per	0x20149000	0x018	[12]	srst_set_spis_n

31 UART

31.1General description

There are two UART interface controllers integrated in BX2400, UART0 and UART1. The UART controller is compliant with the industry standard 16550 and is used for serial communication with other UART devices. The UART controller is an APB slave device. DMA transfers are supported by the UART controller. The baud rate of the UART interface is programmable and the maximum baud rate is 2Mbps. The block diagram of the UART interface controller is shown below:



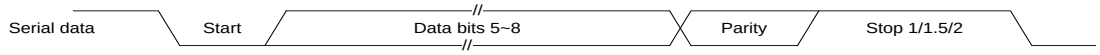
31.2Feature list

- 32 bytes transmit and receive FIFO
- Hardware flow control(UART0 only)
- IRDA 1.0 SIR mode support(UART0 only)
- Programmable baud rate
- Programmable frame format of data bits per frame
- Optional parity bit and programmable number of stop bits
- DMA transmission

31.3UART serial protocol

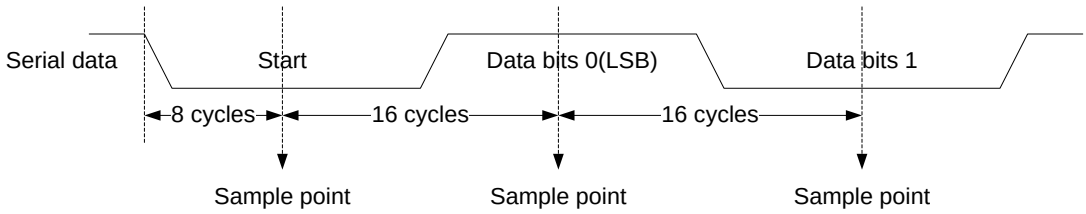
The two device connected through the UART interface are asynchronous, so the start

and stop bits are added to the serial data to indicate the beginning and end of the transfer. The waveform of the UART frame is shown below:



The parity bit is optional and enables the UART device to perform simple error checking on the receive data. The length of the data bits is programmable from 5 to 8. The length of the stop bits is also programmable. The UART line control register is used to control the serial character characteristics. The data word is sent after the start bit, starting with the LSB. All the bits in the transmission are transmitted for exactly the same time duration; the exception to this is the half-stop bit when 1.5 stop bits are used. This duration is referred to as a Bit Period or Bit Time; one Bit Time equals sixteen baud clocks.

To ensure stability on the line, the receiver samples the serial input data at approximately the midpoint of the Bit Time once the start bit has been detected. Because the exact number of baud clocks is known for which each bit was transmitted, calculating the midpoint for sampling is simple; that is, the internal clock rate is programmed to be 16 times of the UART baud rate. Then from the first falling edge the serial data which indicates the beginning of the start bit, 8 clock cycles later is the midpoint of the start bit. Then every sixteen baud clocks after the midpoint sample of the start bit is the midpoint of the data bit. Together with serial input debouncing, this sampling helps to avoid the detection of false start bits. Short glitches are filtered out by debouncing logic. If a glitch is wide enough to avoid filtering by debouncing, a falling edge is detected. However, a start bit is detected only if the line is again sampled low after half a bit time has elapsed. The following waveform shows the sampling points of the data:

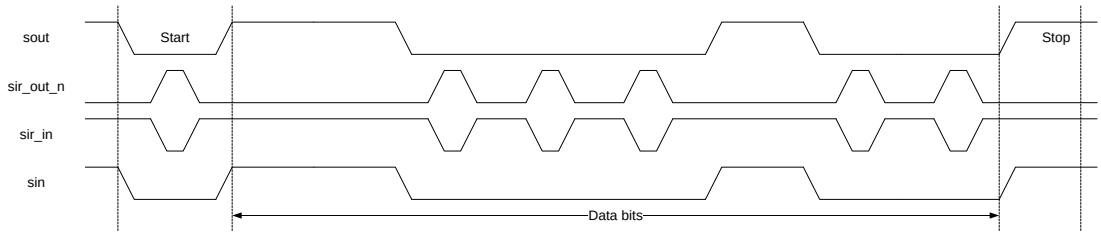


31.4IRDA 1.0 SIR protocol

The Infrared Data Association (IrDA) 1.0 Serial Infrared (SIR) mode supports bi-directional data communications with remote devices using infrared radiation as the transmission medium. The maximu baud of the IrDA mode is 115.2 Kbps.

The frame format is similar to the UART data format. Begins with a start bit, followed by 8 data bits and ends with one or more stop bits. The frame format is shown

below:



The bit format is not the same as the UART. In IrDA, transmitting a single infrared pulse indicates logic 0, Non-transmission of a pulse indicates logic 1. The width of each pulse is 3/16ths of the serial bit time defined in chapter 34.3. Thus, each frame begins with an infrared pulse for the start bit. However, received data is inverted from transmitted data due to infrared pulses energizing the photo transistor base of the IrDA receiver, which pulls its output low.

31.5Auto flow control

UART0 supports auto flow control. The UART interface uses RTS and CTS to control the data rate of the interface. When the receiver FIFO level reaches a preprogrammed threshold defined in the register, the RTS is forced active to inform the transmitter that no space for more data. When the transmitter's CTS is set to active, then the transmitter will stop to send more data until the CTS is set to inactive.

31.6Clock requirement

The internal baud rate clock of UART must be 16 times of the baud rate. There is a clock divider implemented to divide the input clock clk_uart* to be 16 times of the baud rate. And the error rate of the frequency of the internal baud rate clock must be less than 2%.

31.7Interrupts

There are 6 interrupts implemented in UART.

- Intr_receive_error: indicates something is error during receiving, such as FIFO overrun, parity error, framing error, break interrupt. This interrupt is cleared by reading the line status register.
- Intr_receive_data_avail: indicates receive data is available in non-FIFO mode or receive FIFO fill level is bigger than the preprogrammed threshold. This interrupt is cleared by reading the receive data buffer.

- Intr_char_timout: indicates no characters in or out of the RCVR FIFO during the last 4 character times and there is at least 1 character in it during this time. This interrupt is cleared by reading the receive data buffer.
- Intr_transmit_hold_empty: indicates the transmitter holding register empty or TX FIFO at or below threshold. This interrupt is cleared by reading the IIR register or writing to transmit data buffer to make fill level bigger than the threshold.

31.8Clock and reset

31.9UART0

The clock of UART0 is clk_pbus_uart0 and clk_uart0 (refer to chapter 8)
The reset of UART0 is rst_uart0_pbus_n and rst_uart0_n(refer to chapter 9)
The clock gating control registers for UART0 are:

sysc_per	0x20149000	0x010	[11]	clkg1_clr_uart0
sysc_per	0x20149000	0x010	[10]	clkg1_set_uart0
sysc_per	0x20149000	0x010	[9]	clkg0_clr_uart0
sysc_per	0x20149000	0x010	[8]	clkg0_set_uart0

The software reset for UART0 is:

sysc_per	0x20149000	0x018	[5]	srst_clr_uart0_n
sysc_per	0x20149000	0x018	[4]	srst_set_uart0_n

31.10 UART1

The clock of UART1 is clk_pbus_uart1 and clk_uart1 (refer to chapter 8)
The reset of UART1 is rst_uart1_pbus_n and rst_uart1_n(refer to chapter 9)
The clock gating control registers for UART1 are:

sysc_per	0x20149000	0x00C	[27:24]	clk_div_uart1_para_num1_m1(m1)
sysc_per	0x20149000	0x00C	[19:16]	clk_div_uart1_para_num0_m1(m0)
sysc_per	0x20149000	0x00C	[13:8]	clk_div_uart1_para_len1_m1(n1)
sysc_per	0x20149000	0x00C	[5:0]	clk_div_uart1_para_len1_m1(n0)

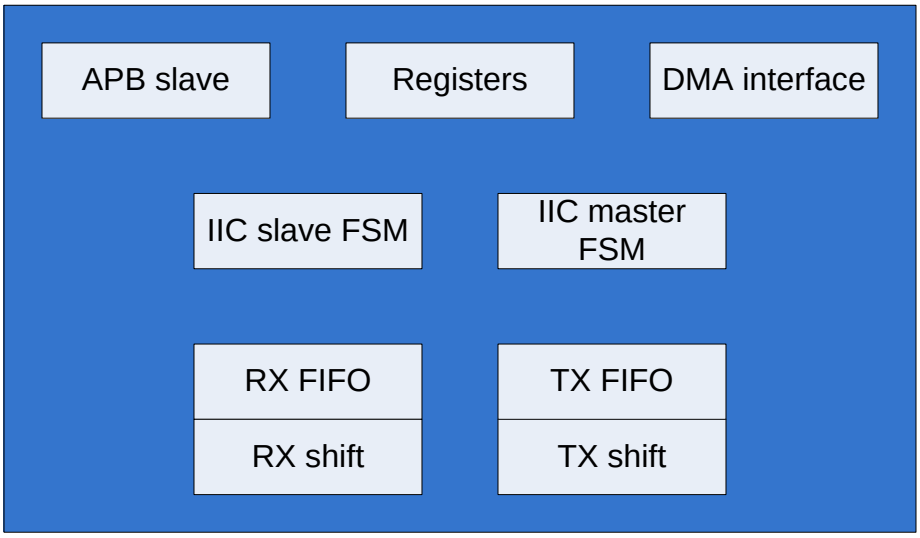
The software reset for UART0 is:

sysc_per	0x20149000	0x018	[7]	srst_clr_uart1_n
sysc_per	0x20149000	0x018	[6]	srst_set_uart1_n

32 IIC

32.1General description

BX2400 integrates two IIC interface controllers and both have the same configuration.
The IIC interface controller is a simple two wires bus with a software defined protocol. The IIC interface controller is an APB bus slave. The IIC interface can work as an IIC master and an IIC slave. The block diagram is shown below:



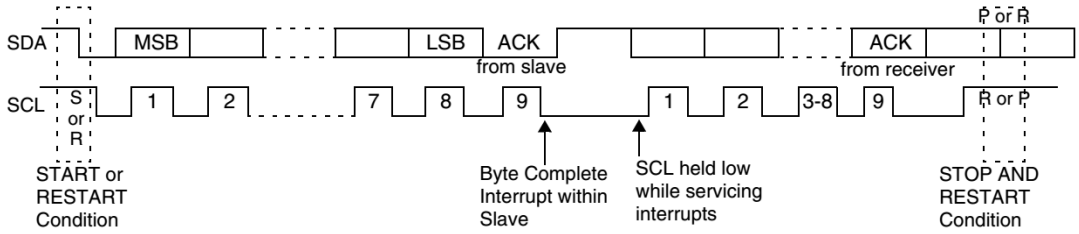
32.2Feature list

- Two-wire IIC serial interface – consists of a serial data line (SDA) and a serial clock (SCL)
- Three speeds are supported: Standard mode (0 to 100Kbps); Fast mode (400Kbps) or fast mode plus (kbps High-speed mode (3.4Mbps)
- Master OR slave IIC operation
- 7- or 10-bit addressing
- 7- or 10-bit combined format transfers
- Transmit and receive FIFO
- Handles Bit and Byte waiting at all bus speeds
- Supports DMA transfer
- Programmable SDA hold time

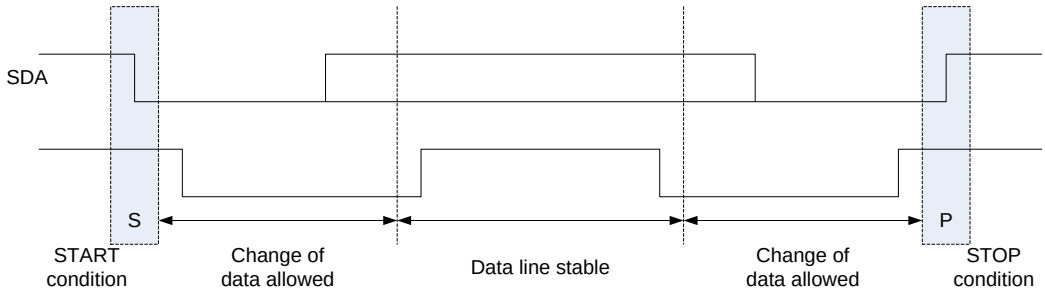
32.3IIC protocol

32.3.1 General description

The IIC bus is a two-wire serial interface, consists of a serial data line (SDA) and a serial clock (SCL). The IIC devices can be considered as masters or slaves when performing data transfers. A master is a device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. A slave is a device receiving or transmitting data according to the clock from the IIC master. Each slave has a unique predefined address. When a master wants to communicate with a slave, the master transmits a START/RESTART bit followed by the slave's address and a control bit (R/W) to determine if the master wants to transmit data or receive data from the slave. The slave then sends an acknowledgement (ACK) bit after the address. Then the data can be transferred between the master and the slave.
During data transmission, the IIC master or IIC slave operates as either a "transmitter" or "receiver," depending on the function of the device. The transmitter sends the data and the receiver receives the data and acknowledges the data.
If the master is sending data to the slave, the master is the transmitter and the slave is the receiver. The transaction between the master and the slave continues until the master terminates the transmission with a STOP condition. If the slave is sending data to the master, the slave is the transmitter and the master is the receiver. The transaction between the master and the slave continues until the master terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a RESTART condition. An example of IIC transfer is shown below:



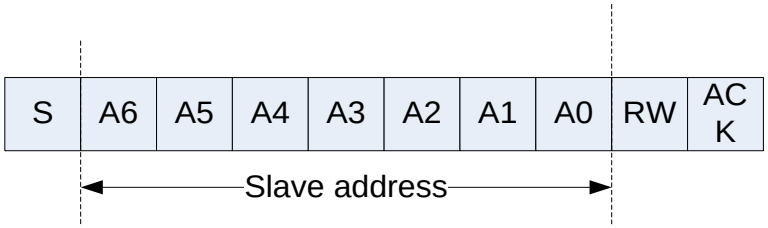
The IIC transfer begins with a START or RESTART bit: The level of the SDA data line changes from high to low, while the SCL clock line remains high. The IIC transfer ends with a STOP bit: The level on the SDA data line passes from the low state to the high state, while the SCL clock line remains high. When the data transfer has been terminated, the bus is free or idle once again. The bus stays busy if a RESTART is generated instead of a STOP condition. The START and STOP condition is shown in the waveform below.



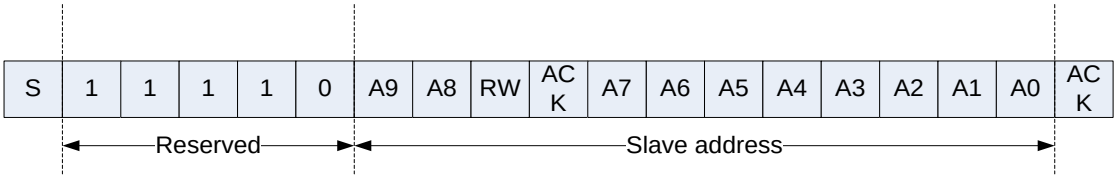
The data line SDA can only changes when the clock line is low except for the START and STOP condition.

32.3.2 Address format

The IIC interface supports both 7 bits slave address and 10 bits slave address. The 7 bits address format is shown below:



The 10 bits address format is shown below:



32.3.3 Transmitting and receiving

Both IIC master and IIC slave can be transmitter and receiver during data exchange. After the master sends the address and R/W bit, the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge signal (ACK). When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a STOP condition. The slave must leave the SDA line high so that the master can abort the transfer. The following diagram shows the scenario that master is the transmitter:

From master to slave

From slave to master

If the master is receiving data then the master responds to the slave-transmitter with an acknowledge pulse after a byte of data has been received, except for the last byte. This is the way the master-receiver notifies the slave-transmitter that this is the last byte. The slave-transmitter relinquishes the SDA line after detecting the No Acknowledge (NACK) so that the master can issue a STOP condition. The following diagram shows the scenario that master is the receiver:

From master to slave

From slave to master

32.4Interrupt

There are 15 interrupts implemented in the IIC interface controller.

- intr_scl_stuck_low: set when the SCL is stuck at low for some time more than a preprogrammed period of time.
- Intr_mst_on_hold: set when the master is holding the bus and the TX FIFO is empty
- Intr_restart_det: set when a restart condition is detected on the IIC interface when the IIC interface is working as an IIC slave and current IIC interface controller is addressed by an IIC master.
- Intr_gen_call: set when a general call address is received and it is acknowledged.
- Intr_start_det: set when a start or restart condition is detected on the IIC interface.
- Intr_stop_det: set when a stop condition is detected on the IIC interface
- Intr_act: set when the IIC interface is in active.
- Intr_rx_done: set when the interface controller is working as a slave transmitter and the transfer on the IIC interface is disabled by the IIC master.

- Intr_tx_abrt: set when not all of the data in the TX FIFO is send when the IIC transfer finish.
- Intr_rd_req: set when the controller is an IIC slave and an IIC master is requesting data from the controller.
- Intr_tx_empt: set when the TX FIFO fill level is at or below a preprogrammed threshold.
- Intr_tx_over: set when the TX FIFO is over run.
- Intr_rx_full: set when the RX FIFO is full
- Intr_rx_over: set when the RX FIFO is over run

32.5Clock and reset

32.5.1 IIC0

The clock of IIC0 is clk_pbus_iic0 and clk_iic0(refer to chapter 8)
The reset of IIC0 is rst_iic0_pbus_n and rst_iic0_n(refer to chapter 9)
The clock gating control register of IIC0 is:

sysc_per	0x20149000	0x010	[3]	clkg1_clr_iic0
sysc_per	0x20149000	0x010	[2]	clkg1_set_iic0
sysc_per	0x20149000	0x010	[1]	clkg0_clr_iic0
sysc_per	0x20149000	0x010	[0]	clkg0_set_iic0

The software reset of IIC0 is:

sysc_per	0x20149000	0x018	[1]	srst_clr_iic0_n
sysc_per	0x20149000	0x018	[0]	srst_set_iic0_n

32.5.2 IIC1

The clock of IIC1 is clk_pbus_iic1 and clk_iic1(refer to chapter 8)
The reset of IIC1 is rst_iic1_pbus_n and rst_iic1_n(refer to chapter 9)
The clock gating control register of IIC1 is:

sysc_per	0x20149000	0x010	[7]	clkg1_clr_iic1
sysc_per	0x20149000	0x010	[6]	clkg1_set_iic1
sysc_per	0x20149000	0x010	[5]	clkg0_clr_iic1
sysc_per	0x20149000	0x010	[4]	clkg0_set_iic1

The software reset of IIC1 is:

sysc_per	0x20149000	0x018	[3]	srst_clr_iic1_n
sysc_per	0x20149000	0x018	[2]	srst_set_iic1_n

33 PWM

33.1General description

The PWM controller outputs 5 pulse width modulation signals. The duty cycle of the PWM signals is programmable. The frequency of the working clock of PWM controller is programmable which is divided from the 32MHz clock. All of the PWM related parameters are calculated based on the divided clock. Each PWM signal has two related parameters: high pulse width and low pulse width. The high pulse width represents the width of the high period of the PWM signal. The low pulse width represents the width of the low period of the PWM signal. Both of the parameters are programmable.

33.2Clock and reset

The clock of the PWM controller is clk_pbus_pwm and clk_pwm0 to clk_pwm5(refer to chapter 8)
The reset of the PWM controller is rst_pwm_pbus_n and rst_pwm_n(refer to chapter 9)
The clock gating control registers of PWM controller are:

sysc_per	0x20149000	0x014	[1]	clkg_clr_pwm_div
sysc_per	0x20149000	0x014	[0]	clkg_set_pwm_div
sysc_per	0x20149000	0x014	[11]	clkg_clr_pwm4
sysc_per	0x20149000	0x014	[10]	clk_set_pwm4
sysc_per	0x20149000	0x014	[9]	clkg_clr_pwm3
sysc_per	0x20149000	0x014	[8]	clk_set_pwm3
sysc_per	0x20149000	0x014	[7]	clkg_clr_pwm2
sysc_per	0x20149000	0x014	[6]	clk_set_pwm2
sysc_per	0x20149000	0x014	[5]	clkg_clr_pwm1
sysc_per	0x20149000	0x014	[4]	clk_set_pwm1
sysc_per	0x20149000	0x014	[3]	clkg_clr_pwm0
sysc_per	0x20149000	0x014	[2]	clk_set_pwm0

The software reset register of PWM controller is:

sysc_per	0x20149000	0x018	[15]	srst_clr_pwm_n
sysc_per	0x20149000	0x018	[14]	srst_set_pwm_n

34 GPIO

34.1General description

The GPIO controller controls the output data and direction of the IOs. It also can read back the data on the external IOs through the APB bus interface. For each bit of GPIO, there are three related registers, one register controls the direction of the current IO, another register controls the output value of the current IO if the direction of current IO is output. The last register represents the value of the current IO. If the direction is input, then the register represents the input value of the IO. If the direction is output, then the register represents the output value of the IO.

34.2Interrupt

Each bit of the GPIO can be used as an interrupt source. The type of the interrupt is programmable with one of the following settings:

- Active high and level
- Active low and level
- Rising edge
- Falling edge

34.3Debounce logic

The GPIO controller implements the debounce logic to filter unexpected pulse on the GPIOs. The debounce logic is running under the 32KHz clock. The capability of the debounce logic is that any spurious glitches less than one 32KHz clock period can be filtered.

When input interrupt signals are debounced using a debounce clock, the signals must be active for a minimum of two cycles of the debounce clock to guarantee that they are registered. Any input pulse widths less than a debounce clock period are bounced. A pulse width between one and two debounce clock widths may or may not propagate, depending on its phase relationship to the debounce clock. If the input pulse spans two rising edges of the debounce clock, it is registered. If it spans only one rising edge, it is not registered.

34.4Clock and reset

The clock of GPIO controller is clk_pbus_gpio and clk_32k_gpio(refer to chapter 8)
The reset of GPIO controller is rst_gpio_pbus_n and rst_gpio_32k_n(refer to chapter 9)
The clock gating control register of GPIO controller is:

sysc_per	0x20149000	0x014	[13]	clkg_clr_gpio
sysc_per	0x20149000	0x014	[12]	clk_set_gpio

The software reset of the GPIO controller is:

sysc_per	0x20149000	0x018	[17]	srst_clr_gpio_n
sysc_per	0x20149000	0x018	[16]	srst_set_gpio_n

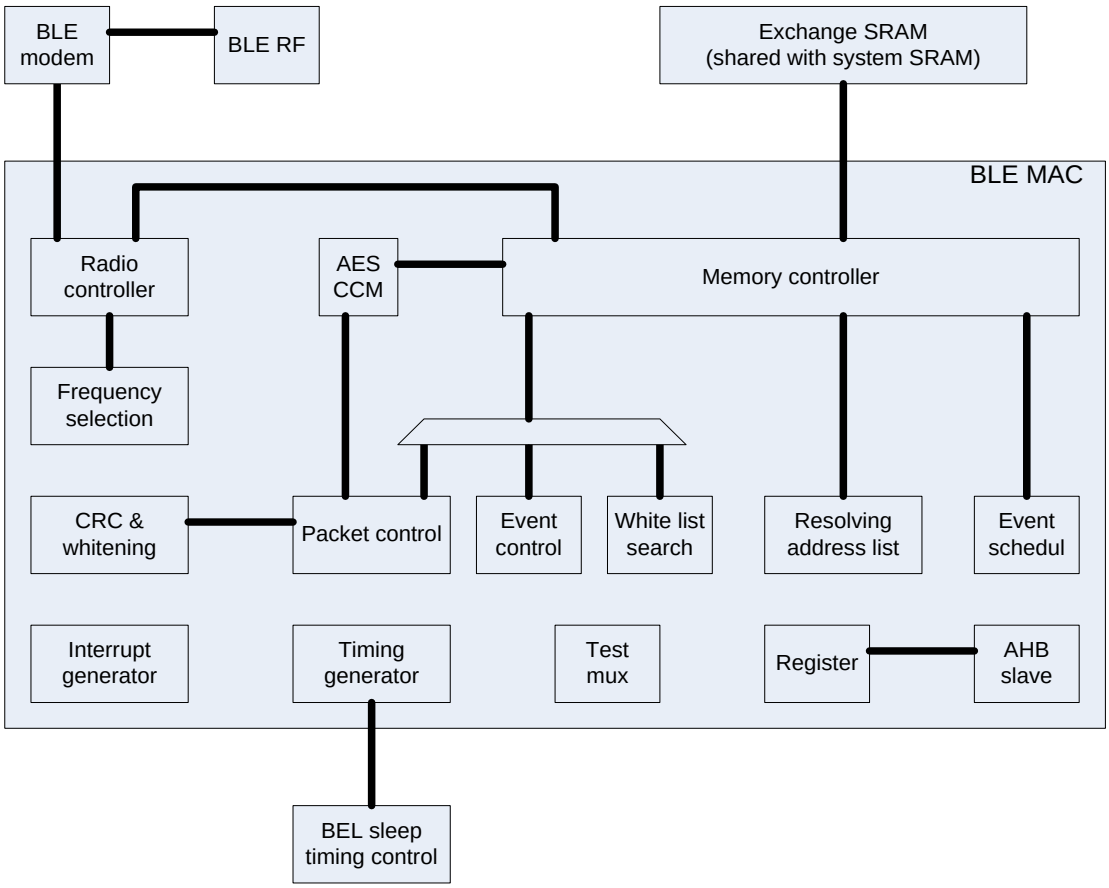
35 BLE MAC

The BLE (Bluetooth Low Energy) MAC is a fully compatible with Bluetooth Smart specification. It is in charge of data packaging and frame scheduling.

35.1Feature list

- Bluetooth Low Energy v4.2 Specifications compliant
- Supports Bluetooth Low Energy v5.0 2Mbps feature
- All packet types (Broadcasting / Advertising / Data / Control)
- Encryption / Decryption (AES-CCM)
- Bit stream processing (CRC, Whitening)
- Frequency Hopping calculation
- FDMA / TDMA / events formatting and synchronization
- All device classes support (Broadcaster, Central, Observer, Peripheral)
- Low power modes supporting 32.0kHz or 32.768kHz low–power clock frequencies
- Simple, clean and optimized interface to RW-BLE Software
- Low power consumption
- Low frequency
- Can be powered down during the protocol’s idle periods
- AHB slave interface for register access
- Direct accessing to the system SRAM through exchange memory interface

35.2Architecture



The block diagram of the BLE MAC is shown above. The exchange memory used by the BLE MAC is shared with the system SRAM. This makes the allocation of the exchange memory and the system memory programmable. The maximum addressable size of the exchange memory is 32KB which is addressed at 0x00128000 of the system address space. The exchange memory contains the exchange table, control structures, TX/RX descriptors and the associated data buffers, and RAL structure.

The exchange table is used to indicate the addresses of the different control structures in the exchange memory. The exchange table pointer is provided through programmable register. The control structures contain all the parameters required to control the events and their timings, and link to the TX descriptors through CS-TXDESCPTR field. RX descriptor pointer is provided through programmable register. TX/RX descriptors describe the packets exchanged during the event. A set of TX/RX descriptors can describe either a finite chained list, or a ring buffer. The TX/RX descriptors contain the next TX/RX descriptor pointer value, the status of the current TX/RX packet, and the data buffers used for data to be transmitted or received. Each descriptor embeds a Data pointer that indicates the location of its associated Data

Buffer. The following table describes the exchange memory content: data intends to cover advertising, scanning and link layer connection modes.

Field name	Size(byte)	Description
Exchange Table	64	Scheduling entry of the RW-BLE Core, read each 625µs
Control Structure	90	A single Control Structure contains all control parameters of advertising, scanning, and connection events. Different control structures are needed to handle efficiently all the event types
Tx Descriptor	6	Define next Tx Descriptor pointer value, as well as transmitted packet header content (according CS-FORMAT) and gives status about transmission of this packet (done or pending)
Tx Data Buffer	Up to 251	Sized exactly to the amount of data to be transmitted, as per the packet payload size.
Tx ISO Buffer	Up to 251	Size controlled by RW-BLE Software. Size is defined by ISOTRCNTL<0/1/2>- ISO<0/1/2>TXLEN register field. Tx ISO Buffer pointer are programmed through ISOCURRENTTXPTR<0/1/2> register
Rx Descriptor	10	Define next Rx Descriptor pointer value, as well as received packet header content (according to CS-FORMAT) and gives status about reception of this buffer (done or free)
Rx Data Buffer	Up to 251	Sized to the maximum possible amount of received data, as per the different packet payload size.
Rx ISO Buffer	Up to 251	Size controlled by RW-BLE Software. Size is defined by ISOTRCNTL<0/1/2>- ISO<0/1/2>RXLEN register field. Rx ISO Buffer pointer are programmed through ISOCURRENTRXPTR<0/1/2> register
RAL Structure	52	Resolving Address List Structure used during RPA resolution/renewal Note there may be several structures embedded in the Exchange Memory

35.3Interrupt

The interrupt generator creates the following interrupts, from the highest to the lowest priority:

- ble_error_irq: Error interrupt, generated when undesired behavior or bad programming occurs in the RWBLE Core. Error status in reported in ERRORTYPESTAT register.
- ble_cscnt_irq: 625µs base time reference interrupt, available in active modes
- ble_rx_irq: Receipt interrupt at the end of each received packets, or when received packets number equals CS-RXTHR value.

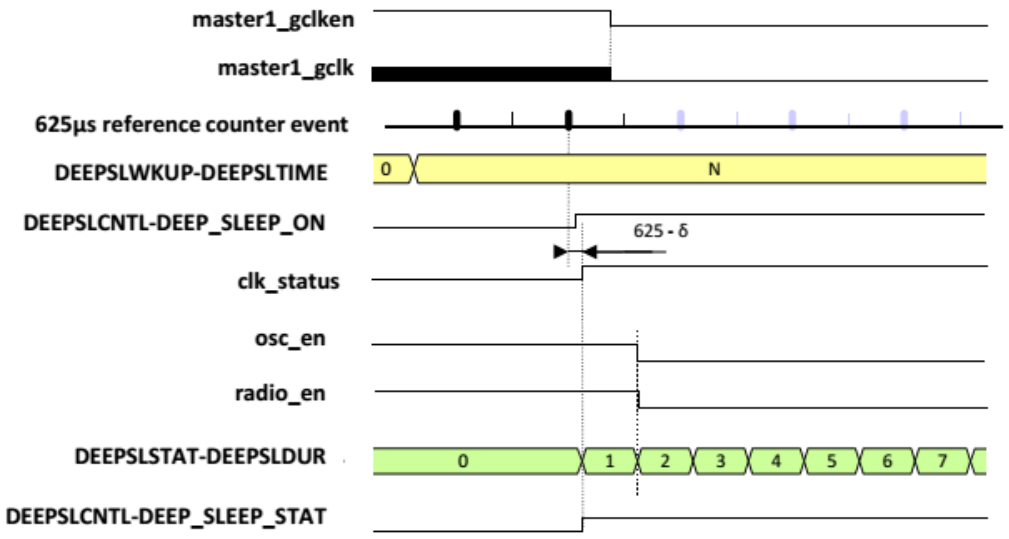
- ble_event_irq: End of Advertising / Scanning / Connection events interrupt. Generated on normal termination, or on anticipated pre-fetch mechanism request.
- ble_slp_irq: End of Sleep mode interrupt
- ble_crypt_irq: Encryption engine interrupt, generated either when AES-128 ciphering/deciphering process is finished
- ble_grosstgtim_irq: Gross Target Timer interrupt generated when Gross Target timer expired. Timer resolution is 10ms.
- ble_finetgtim_irq: Fine Target Timer interrupt generated when Fine Target timer expired. Timer resolution is 625µs.
- ble_sw_irq: SW triggered interrupt: generated on SW request through register access (i.e RWBLECNTLSWINT_REQ)

35.4Sleep mode

The BLE MAC supports sleep mode to save power during the protocol's inactive time. When entering the sleep mode, the BLE MAC can be powered off and the protocol's timer is running under 32KHz clock in the module "BLE MAC low power controller" described in chapter 14. Then at some preprogrammed time the BLE MAC low power controller will generate BLE low power interrupt to wake up the system if necessary. Then the system will prepare for the wake up of the BLE MAC, including set the frequency of the BLE clock and wake up the BLE power domain. Then the BLE MAC low power controller will inform BLE MAC that it is time to wake up from sleep mode. BLE MAC will take over the timing of the BLE protocol from the BLE MAC low power controller and then enter the active mode.

35.4.1 Switch from active mode to deep sleep mode

By activating the corresponding DEEPSLCNTL-DEEP_SLEEP_ON register bit, the software puts the RW-BLE Core into Deep Sleep mode. The RW-BLE Core then uses *low_power_clk* (usually 32.0kHz or 32.768 kHz) in order to maintain its internal 625µs timing reference(by module BLE MAC low power controller). Once the *low_power_clk* is used for base time reference, *master1_gclk* and *master2_gclk* clock can be gated. The Radio Module and the Oscillator can be powered down, driving low *osc_en* and *radio_en* outputs, according to their respective DEEPSLCNTL bit fields. The radio and oscillator enable signals are conveyed only if the corresponding DEEPSLCNTL bits are set. The DEEPSLWKUP register sets the time to stay in Deep Sleep before wake-up in number of *low_power_clk* clock cycles.
The following waveform shows the active mode to low power mode switch timing diagram:



On DEEPSLPCNT-DEEP_SLEEP_ON rising edge, on *low_power_clk* first rising edge, *clk_status* goes high, when *osc_en* and *radio_en* go low (depending on their respective DEEPSLPCNTL register bits). When *clk_status* goes high, Fine Counter value is blocked at δ value, and *master1_gclk* clock can be gated on next *low_power_clk* falling edge. Fine Counter resumes later at δ (on wake-up) in order maintaining the correct 625µs reference timing. The software must poll the value of the DEEPSLSTAT-DEEPSLDUR register to make sure the status switching has finished. After the BLE MAC has entered the deep sleep mode, the BLE power domain can be powered off and the system can also be powered off if necessary. Then if the power domain is powered off then before BLE MAC goes to active mode, the CPU power domain and the BLE power domain must be powered up and software must do the preparation for the running of the BLE MAC after the power domain is powered up.

35.4.2 Switch from deep sleep mode to active mode

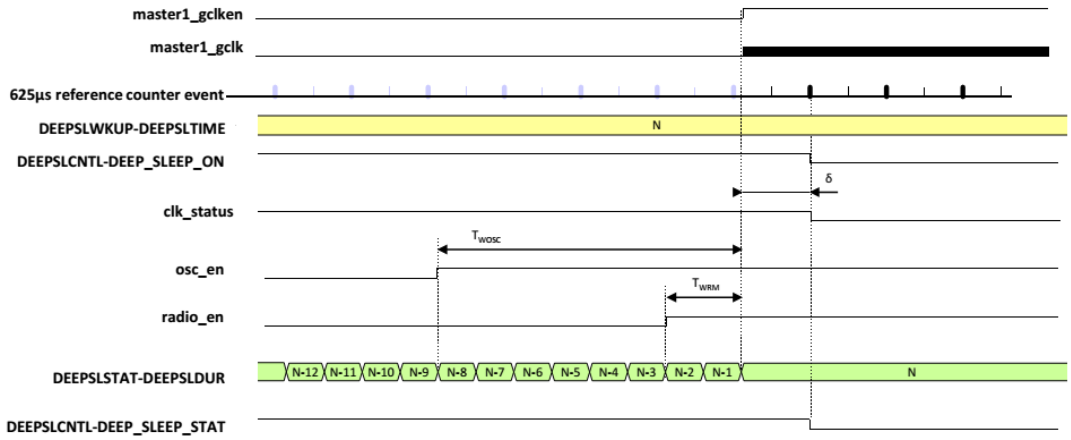
There are two possibilities for RW-BLE Core to terminate a Deep-Sleep phase:

- Termination at the end of a predetermined time
- Software wake-up request

In deep sleep mode, the BLE low power controller is waiting for deep sleep counter to equal DEEPSLWKUP–DEEPSLTIME. After a configurable time before wake up time (i.e. using ENBPRESET register fields), the BLE MAC low power controller pull up the radio_en and osc_en output which can be used by the system to start to do the preparation for BLE MAC running, according to their respective DEEPSLCNTL bit fields. The preparation includes powering up the CPU and BLE power domain, powering up the 32MHz crystal, configure the PLL, configure the clock and reset for BLE MAC and configure other related system registers. The time margin for the

preparation is programmable. At the end of the sleep phase the DEEPSLCNTL-DEEP_SLEEP_ON bit is reset, leading to the RW-BLE Core wake up, which switches back to the fast clock, and restore proper 625µs internal reference time. The output clk_status indicates that the master clock coming from the fast oscillator is available and stable.

The following waveform shows a typical deep sleep phase that is terminated at its predetermined time.



35.5Clock and reset

The clock for BLE MAC is clk_ble_mac and clk_hbus_ble(refer to chapter 8)

The reset for BLE MAC is rst_ble_hbus_n and rst_ble_mac_n(refer to chapter 9)

The clock gating control register of BLE MAC is:

sysc_awo	0x20201000	0x004	[11]	clkg_ble_clr
sysc_awo	0x20201000	0x004	[10]	clkg_ble_set

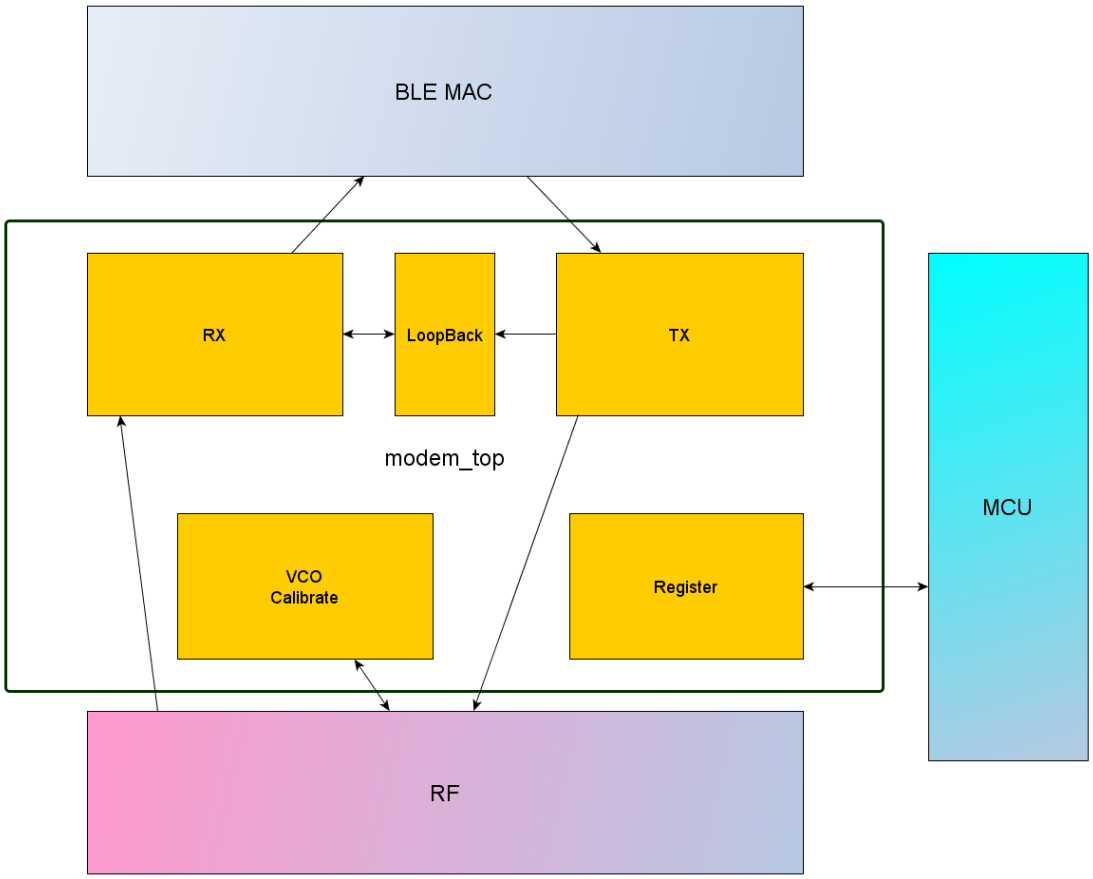
The software reset of BLE MAC is:

sysc_awo	0x20201000	0x040	[7]	srst_ble_n_clr
sysc_awo	0x20201000	0x040	[6]	srst_ble_n_set

36 BLE modem

36.1General description

The BLE modem modulates the transmit package and demodulates the receive package. The GFSK modulation is applied according to the BLE protocol. The block diagram of the BLE modem is shown below:



36.2RX

The RX module gets data from the BLE RF. The sample rate of the RX data is 32MHz or 48MHz which is programmable. The received data is mixed with an intermediate frequency waveform signal and then pass through several FIR low pass filters. The filtered signal is converted from the phase field to the frequency field and then the BLE package bit is got by comparing with an estimated carrier frequency offset. If the synchronization word is found in the package data, then the package is treated as a

valid package and output to the BLE MAC. If the synchronization word is not found then the data is dropped.

36.3TX

The GFSK modulation is applied in the BLE protocol. The bits in the data package from the BLE MAC represent the frequency value: 0 represents that the frequency is center frequency minus 250KHz and 1 represents that the frequency is center frequency plus 250KHz. In the TX module, the data package from the BLE MAC is over sampled and filtered by a Gaussian filter. The following process is designed in two modes: IQ mode and DM mode.
In the IQ mode, the filtered data is accumulated and converted to the phase field. The accumulated data now represents the phase information and then converted to IQ data by looking into a predefined SIN/COS table. The IQ data is output to the DAC in the RF.
In the DM mode, the filtered data is converted to the PLL control signal to make the high frequency PLL to output correct frequency signal.

36.4VCO calibration

36.4.1 Deviation calibration

This deviation calibration is used only in DM mode to get the correct VCO capacitance array value to deviate a precise frequency offset. The deviation calibration must be done once just after power on of BX2400. The result of deviation calibration is saved in the local register and called automatically.

36.4.2 Band calibration

Band calibration is used to provide a control signal for the VCO in the RF to increase the accuracy of the center frequency for each channel. The control signal is different for each channel. So the band calibration must be done before each transmission and reception. The Band calibration is enabled automatically each time the modem sends or receives data.

36.5Clock and reset

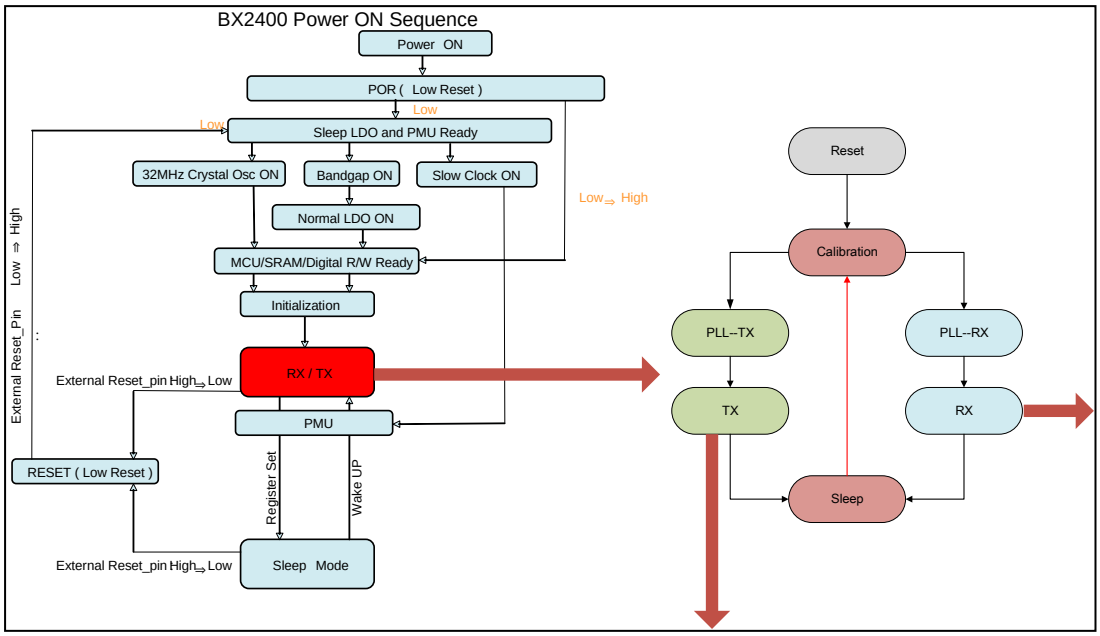
The clock of the BLE modem is clk_ble_mdm_rx, clk_32m_mdm_tx and clk_ble_mac.

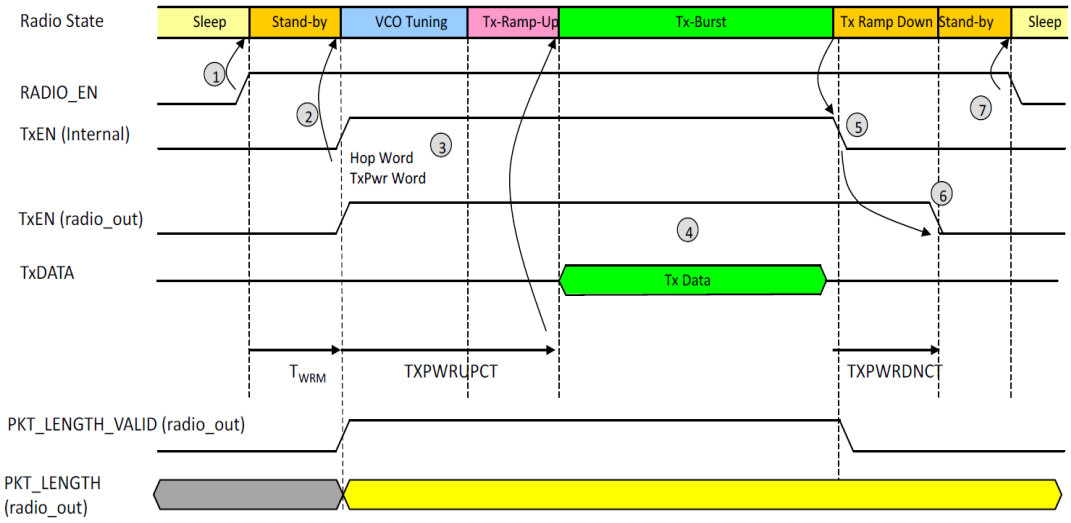
(refer to chapter 7)

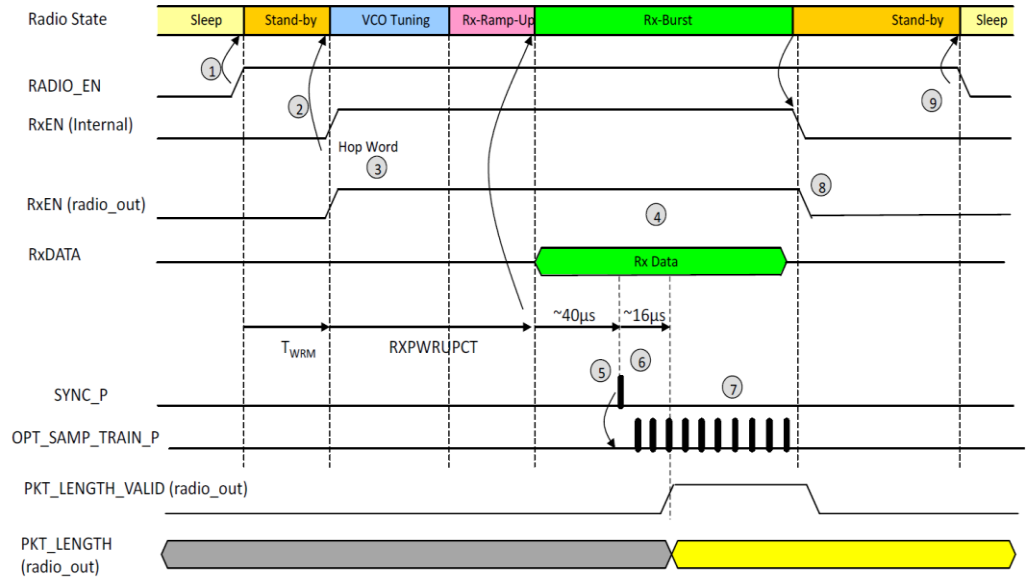
The reset of the BLE modem is rst_ble_mdm_rx_n, rst_ble_mdm_tx_n and rst_ble_mac_n (refer to chapter 8)

37 Power On Sequence

BX2400 power on sequence is illustrated in the figure below.







38 Power management

The power management unit (PMU) in BX2400 comprises a DC-DC Buck Converter, various LDOs for the different power domains of the system, a Constant Current Constant Voltage (CCCV) charger for battery recharging. The PMU is capable of supplying external devices even during BX2400 in sleep mode.

Features

- DC-DC Buck Converter with excellent 93% efficiency
- Programmable DC-DC converter output charging sequence
- One LDO output up to 3.6 V with up to 50 mA load capability
- DC-DC converter automatically on/off control during in sleep mode
- Active and Sleep mode current limited LDOs
- Use of small external components
- Supply of external rails (V33, VDD1V8) while in Sleep mode
- Linear charger with battery/die-temperature protection

39 DC/DC Buck Converter

Two DC/DC Buck Converter modes are designed for BX2400. One is DC/DC Converter ON and the other is DC/DC Converter Bypass. BX2400 will automatically enter DC/DC Converter ON mode within 150 us while CPU is active. And DC/DC Converter will be bypass mode during CPU in sleep mode.

(Ta = -40℃ ~ +85℃ , 2.3V ≦ VBAT ≦ 5V)

Item	Parameter	Conditions	MIN.	TYP.	MAX.	Unit
DC/DC Converter	Input Voltage		2.3		4.75	V
	Output Voltage	DC/DC Converter On (Default 1.4V)		1.4		V
				1.3		V
				1.2		V
				1.1		V
		DC/DC Converter Bypass		VBAT-0.3		V
	Quiescent Current	DC/DC Converter On		100		uA
	Start-up Time	DC/DC Converter On		120		usec

40 LDO

BX2400 provides several LDOs to external power supply, including two 3.3V LDO and one 1.8V LDO. The LDO for external use can be configured as active even in sleep mode. The LDO characteristics are listed below.

(Ta = -40℃ ~ +125℃ , 2.3V ≦ VBAT ≦5.5V)

Item	Parameter	MIN.	TYP.	MAX.	Unit
1.8V LDO	Output Current			40	mA
	Output Voltage at 96MHz > 10mA		1.8		V

(Ta = -40℃ ~ +125℃ , 2.3V ≦ VBAT ≦5.5V)

Item	Parameter	MIN.	TYP.	MAX.	Unit
3.0V LDO_1	Output Current			25	mA
	Output Voltage		3.0		V

(Ta = -40℃ ~ +125℃ , 2.3V ≦ VBAT ≦5.5V)

Item	Parameter	MIN.	TYP.	MAX.	Unit
3.0V LDO_2	Output Current			50	mA
	Output Voltage		3.0		V

41 Charger

BX2400 provides a linear charger for battery recharging, and charging detection circuit. The control circuit keeps the charge voltage or the charge current at the predefined values (whichever of the two is reached first). The charger current characteristic is listed below.

Parameter	Conditions	Min	Typ	Max	Unit
I_{CHARGE_0}	charge current 0000b		5		mA
I_{CHARGE_1}	charge current 0001b		10		mA
I_{CHARGE_2}	charge current 0010b		30		mA
I_{CHARGE_3}	charge current 0011b		45		mA
I_{CHARGE_4}	charge current 0100b		60		mA
I_{CHARGE_5}	charge current 0101b		90		mA
I_{CHARGE_6}	charge current 0110b		120		mA
I_{CHARGE_7}	charge current 0111b		150		mA
I_{CHARGE_8}	charge current 1000b		180		mA
I_{CHARGE_9}	charge current 1001b		210		mA
I_{CHARGE_10}	charge current 1010b		270		mA

The charge control circuit is initially supplied from pin VBUS. The complete charging circuit is powered down, when the VBUS voltage is low for more than 10 ms or by setting register.

42 32MHz Crystal Oscillator

32MHz Crystal Oscillator characteristics are listed below. Also, frequency compensation, programmable level of frequency compensation capacitors will be implemented to cover 32MHz Crystal variation, aging … over temperature range -40℃ ~ +125℃.

(Ta = -40℃ ~ +125℃ , 2.3V ≦ VBAT ≦ 5V)

Item	Parameter	MIN.	TYP.	MAX.	Unit
32MHz Crystal Oscillator	Oscillation frequency		32		MHz
	Frequency offset	-20		20	ppm
	Startup time			150	μs
	Shunt Capacitor (with NDK NX3225SA)			10	pF

43 32MHz RC Oscillator

32MHz RC Oscillator characteristics are listed below.

(Ta = -40℃ ~ +125℃ , 2.3V ≦ VBAT ≦ 5V)

Item	Parameter	MIN.	TYP.	MAX.	Unit
Internal 32MHz RC Oscillator	Oscillation frequency		32		MHz
	Startup time		-	20	μs

44 32KHz Crystal Oscillator

32KHz Crystal Oscillator characteristics are listed below.

(Ta = -40℃ ~ +125℃ , 2.3V ≦ VBAT ≦ 5V)

Item	Parameter	MIN.	TYP.	MAX.	Unit
32.768KHz Crystal Oscillator	Oscillation frequency		32.768		KHz
	Frequency offset	-20		20	ppm
	Series Resistor		10		MΩ
	Shunt Capacitor		33		pF

45 32KHz RC Oscillator

32KHz RC Oscillator characteristics are listed below

(Ta = -40℃ ~ +125℃ , 2.3V ≦ VBAT ≦ 5V)

Item	Parameter	MIN.	TYP.	MAX.	Unit
Internal 32KHz RC Oscillator	Oscillation frequency		32		KHz
	Frequency offset after calibrated by 32MHz Crystal Oscillator	-500		500	ppm
	Chip-to-chip Variation from -40℃ ~ +85℃ , 2.3V ~ 5V	10	-	100	KHz

46 PLL

BX2400 provides alternative 96/80/64/48/32/16 MHz system clock. Changing this system's clock can be done within 300 us without affecting the operation of the chip. This PLL dissipates additional 0.4 mA when operating at 96 MHz.

(Ta = -40℃ ~ +85℃ , 2.3V ≦ VBAT ≦ 5V)

Item	Parameter	MIN.	TYP.	MAX.	Unit
PLL	Output Clock (Default 96MHz)		96		MHz
			80		MHz
			64		MHz
			48		MHz
			32		MHz
			16		MHz
	Reference Clock		16		MHz
	Stable time			300	μs