Erbium Minion Isotope TRM

Document Status: Proprietery Internal Draft.

Created By

Vijayvithal
jvs@veevx.com
Veevx

For

Internal

Ref: 05be1de+post1
December 3, 2025

Contents

Overview

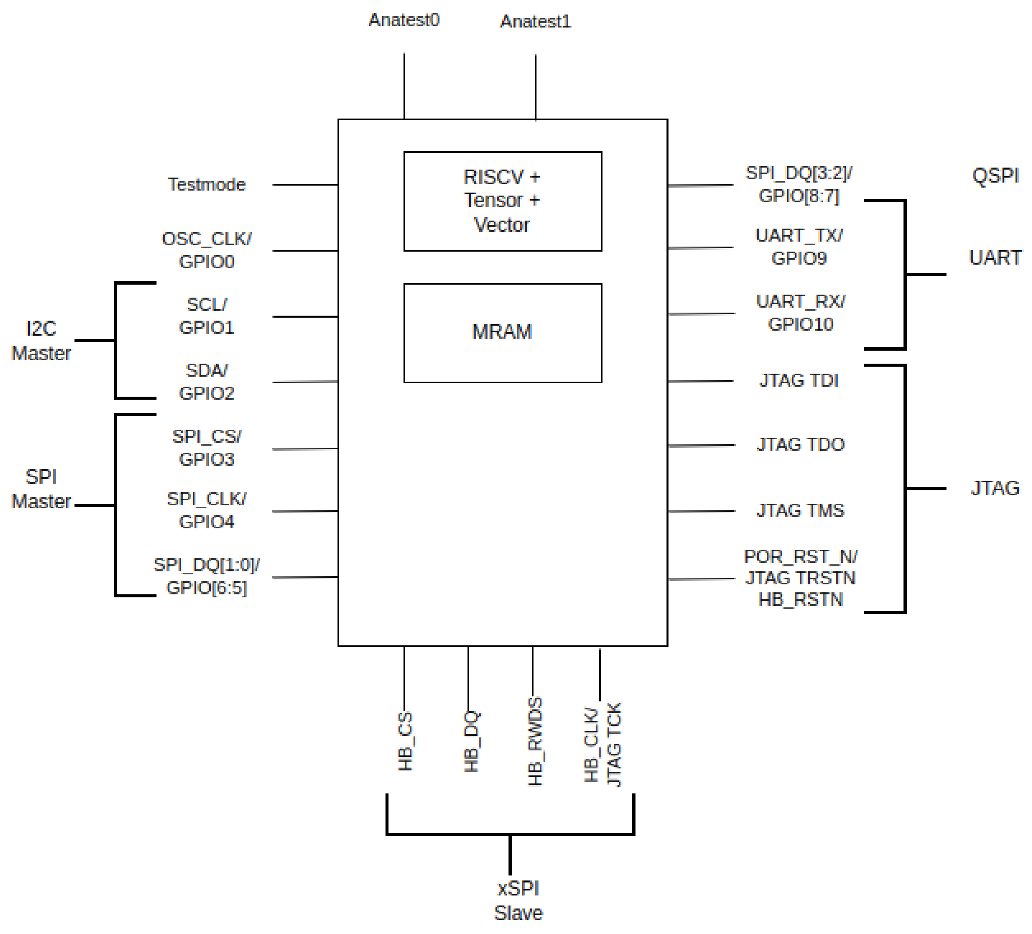


Figure 1: Erbuim RISCV Isotope

Features

- External Interfaces
  - JTAG
  - UART
  - xSPI Target,
  - I2C Controller
  - (Q)SPI Controller
  - 11 GPIO
  - Chiplet Interfaces
    - ⋆ AHB
    - ⋆ AXI
- CPU:
  - 8C,16T RISCV64(IMDFC +Tensor, +Vector)

xSPI Interface

- JESD251C and JESD216H compliant
- Supports
  - 1S-1S-1S
  - 4S-4D-4D
  - 4D-4D-4D
  - 8S-8S-8S
  - 8D-8D-8D (Octal)
  - 8D-8D-8D (Hyperbus)
- Operating frequency 200 Mhz
- 64 bit register and memory access
- Linear Burst capable.

Chiplet Interface

- 2 Interfaces AXI, AHB
- AXI and AHB in 32bit and 64 bit variants.
- Chiplet interface is not available in testchip tapeouts.

Note: While the chiplet interface is a part of the design it is not available(optimised away during PD) in testchips. i.e. erbium will not have the chiplet interface.

Signals

Test Signals

Signal	Direction	Note
ANATEST0	INOUT	Analog Test Pins
ANATEST1	INOUT	Analog Test Pins
TestMode	INPUT	Digital Testmode Enable. For internal use only. Tie to 0 in production.
OSC_CLK_OUT	OUTPUT	Oscillator clk div N

xSPI Target Interface

Port	Direction	Note
HB_CLK,	INPUT	Clock. Shared with JTAG
HB_RSTN	INPUT	Reset. Shared with JTAG
HB_CS	INPUT	Chip Select
HB_DQ,	INPUT	Data, Bidirectional
HB_RWDS,	INOUT	Data Strobe

UART Target Interface

Port	Direction	Note
Tx	OUTPUT	Transmit Pin
Rx	INPUT	Receive Pin

I2C Controller Interface

Port	Direction	Note
SCL	OUTPUT	Clock
SDA	INOUT	Data. Needs external pullup

QSPI Controller Interface

Port	Direction	Note
CLK	OUTPUT	Clock
DQ[3:0]	INOUT	Data
CS	OUTPUT	Chip select

GPIO Interface.

Port	Direction	Note
GPIO[10:0]	INOUT	All GPIO Pins are muxed with other IO protocols.

JTAG Signals

Port	Direction	Note
JTAG_TDI	INPUT	
JTAG_TDO	OUTPUT	
JTAG_TMS	INPUT	

Pin Muxing

A Port	B Port	A_SEL
OSC_CLK_OUT	GPIO[0]	TestMode PIN
SCL	GPIO[1]	system_register.SystemConfig.i2c_enable
SDA	GPIO[2]	system_register.SystemConfig.i2c_enable
SPI_CS	GPIO[3]	system_register.SystemConfig.spi_enable
SPI_CLK	GPIO[4]	system_register.SystemConfig.spi_enable
SPI_DQ[1:0]	GPIO[6:5]	system_register.SystemConfig.spi_enable
SPI_DQ[3:2]	GPIO[8:7]	system_register.SystemConfig.qspi_enable
UART_TX	GPIO[9]	system_register.SystemConfig.uart_enable
UART_RX	GPIO[10]	system_register.SystemConfig.uart_enable

Note At boot SPI_enable=1 I2C_enable=0, QSPI_enable=0;

Bring up sequence.

First boot on the tester.

- TestMode pin is held high. CPU is halted (in reset mode?)
- POR is toggled.
- OSC_CLK_OUT is measured, the PPM difference from expected frequency is calculated.
- Osc TRIM Parameters are calculated. (Index in Trim table?)
- Part is speed graded.
- Osc TRIM parameters and speed grade is written to OTP

Second Boot.

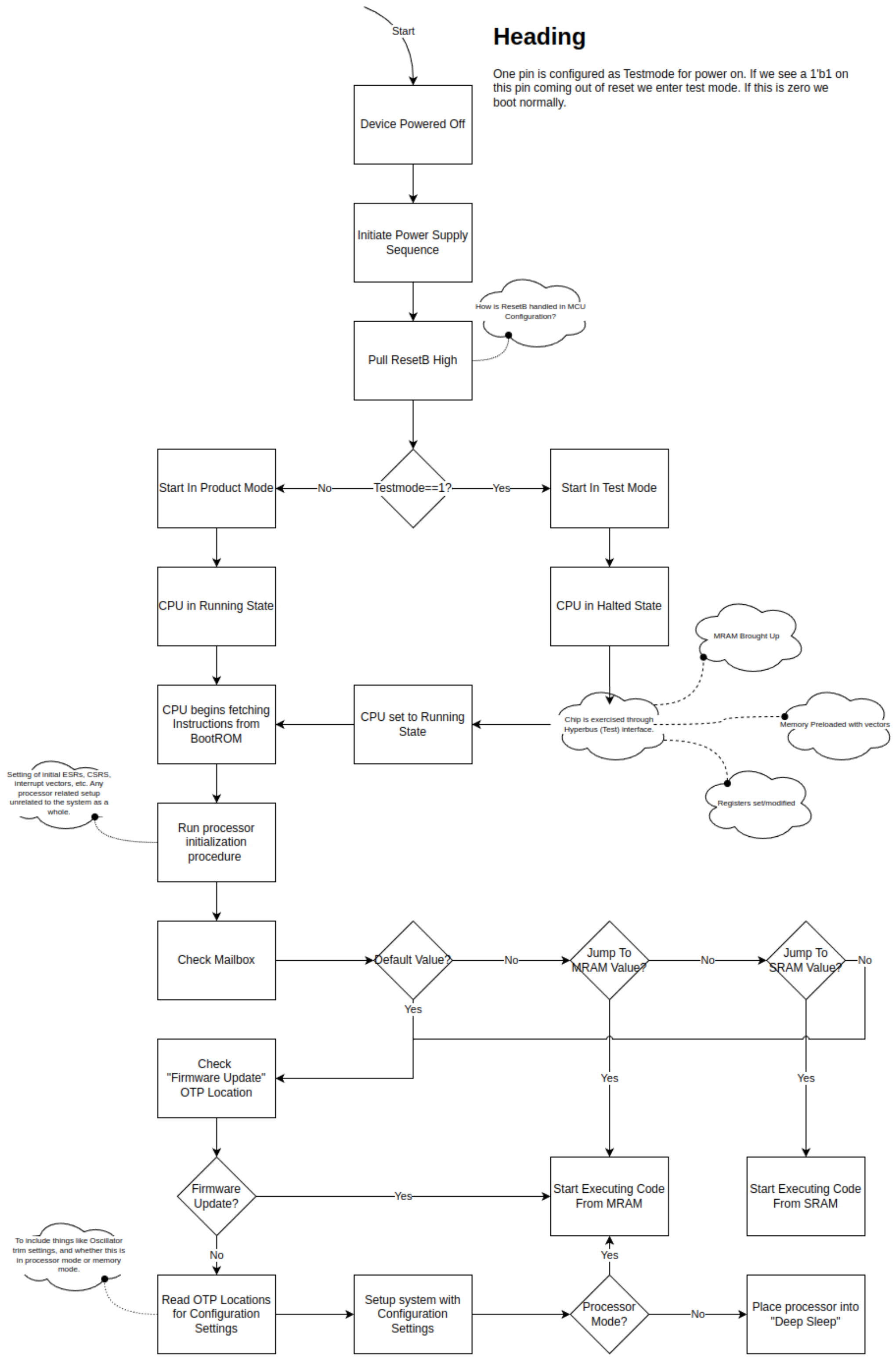
- TestMode is held low.
- CPU Starts executing bootrom code.
    - OTP is read(Trim Param, Speed grade, ROM Update).
    - If ROM Update bit set, jump to updated firmware in MRAM.
    - Trim Ring Oscillator.
    - Program RO-CLK Div for highest clk freq supported by device speed grade.
    - Calculate Clk Div for Periph and MRAM.
    - Program Periph clkdiv and mram clkdiv.
    - Read CHIPMODE.bootloader register to determine location of application code (TCM,MRAM(default))
    - Jump to application code.

Program Mode

- Bring MRAM out of deep sleep.
- Load MRAM/TCM with the application code
- Set CHIPMODE.bootloader register.

Application Code

- If MRAM is in deep sleep mode it needs to be brought out of sleep.
    - Write to `system_registers.PowerDomainReq.mram_dsleep_en` to bring MRAM out of deep sleep.


Start
Device Powered Off
Initiate Power Supply Sequence
How is ResetB handled in MCU Configuration?
Pull ResetB High
Start In Product Mode
No
Testmode==1?
Yes
Start In Test Mode
CPU in Running State
CPU in Halted State
MRAM Brought Up
CPU begins fetching Instructions from BootROM
CPU set to Running State
Chip is exercised through Hyperbus (Test) interface.
Memory Preloaded with vectors
Registers set/modified
Setting of initial ESRs, CSRS, interrupt vectors, etc. Any processor related setup unrelated to the system as a whole.
Run processor initialization procedure
Check Mailbox
Default Value?
No
Jump To MRAM Value?
No
Jump To SRAM Value?
No
Yes
Check "Firmware Update" OTP Location
Yes
Yes
Firmware Update?
Yes
Start Executing Code From MRAM
Start Executing Code From SRAM
To include things like Oscillator trim settings, and whether this is in processor mode or memory mode.
No
Yes
Read OTP Locations for Configuration Settings
Setup system with Configuration Settings
Processor Mode?
No
Place processor into "Deep Sleep"

Heading

One pin is configured as Testmode for power on. If we see a 1'b1 on this pin coming out of reset we enter test mode. If this is zero we boot normally.

Figure 2: Boot sequence

- Use `system_registers.Mailbox0.mbox0` and `system_registers.Mailbox1.mbox1` for communication from CPU to host chip and vice versa.

Power Reset & Clock Management.

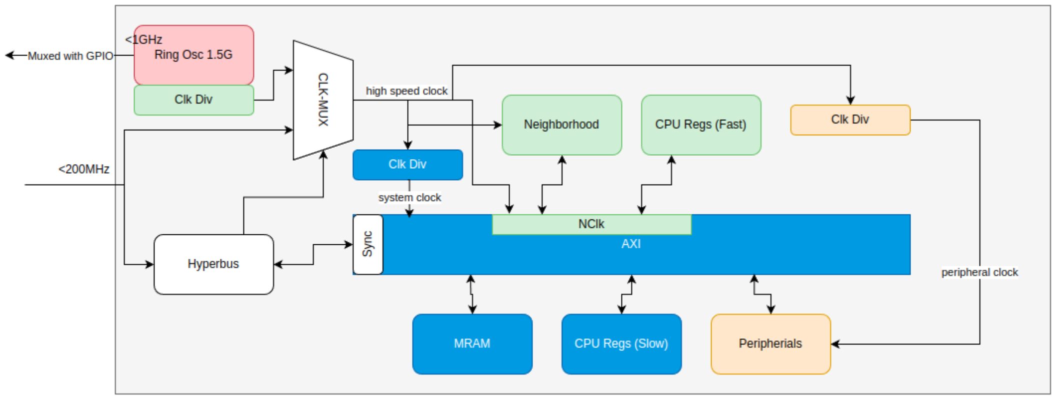


Figure 3: Clock Domains

Clock and reset signals.

System Clocks

There are dedicated clock and reset signals for each interface

Interface	Clock	Reset	Note
Chiplet	HCLK_ACLK	HRESETn_ARESETn	Active low Reset, Tied High
Hyperbus	HB_CLK	HB_RSTN	Active low Reset
Internal	OSC_CLK		Internal Ring Oscillator

Note In Erbium the Chiplet domain is not available and only hyperbus reset is used to reset the system.

OSC Clock

- This is a high speed clock generated internally from a ring oscillator.
- The raw ring oscillator(RO) frequency is process dependent and can vary by a few hundred MHz from the designed frequency.
- The first level clock divider is programmed to divide the output of the RO by 20 resulting in a <75MHz boot clock.

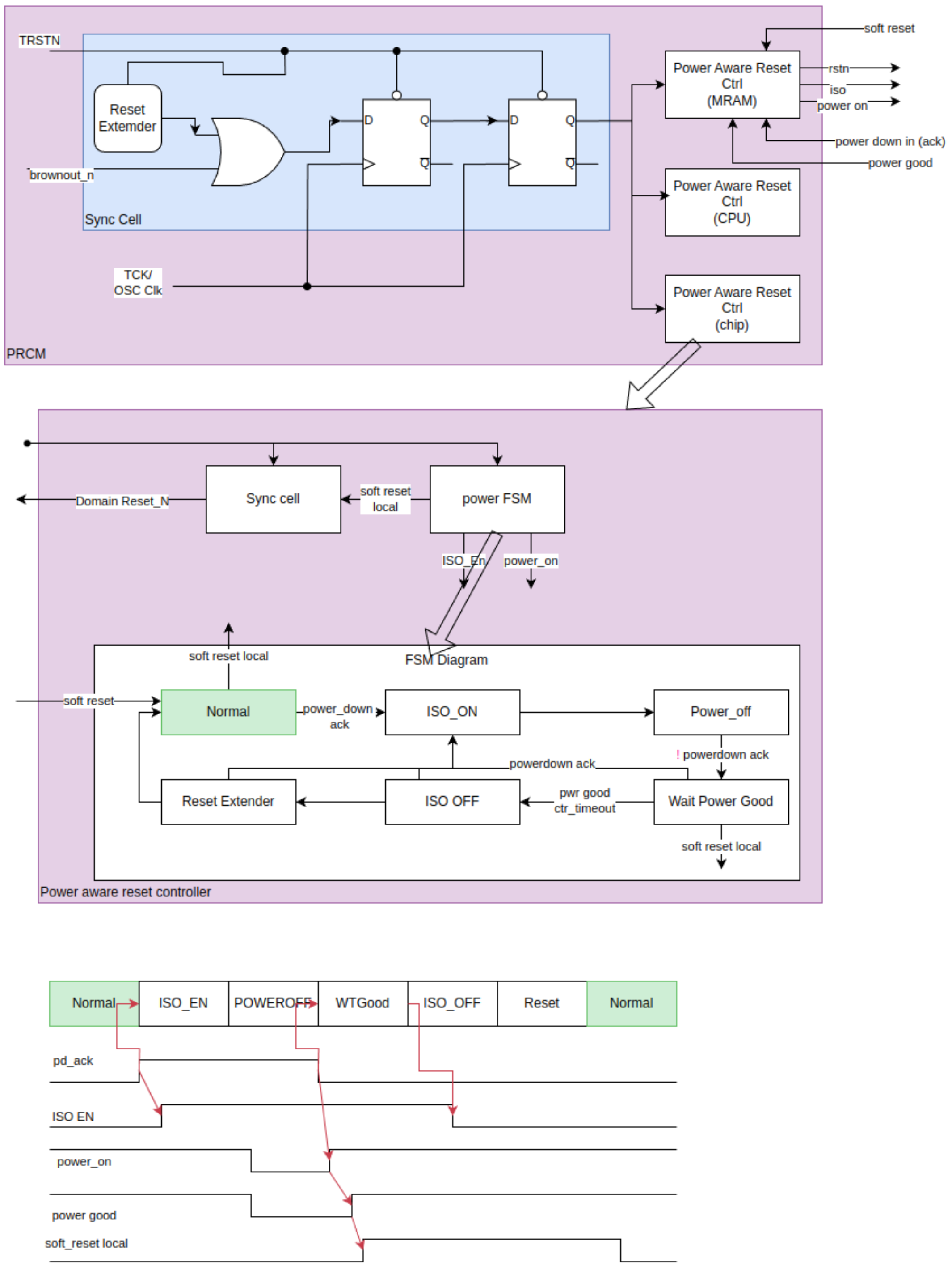


Figure 4: PRCM

- The tester measures the RO frequency, and process variation and writes it to the OTP. This is used to
  - Trim the RO to generate a 1GHz clock.
  - Program the clock divider to provide the required highspeed CPU Clock.
  - Clock dividers can be reprogrammed later as per the system load.
- MRAM and rest of the system operates at a 200MHz clock. A second clock divider divides the `high speed clock` to generate the `system clock`
- Serial protocols require a clock programmed to N times the required baud rate. The Peripheral clock is used as this sampling clock.

Hyperbus Clock

- This is an external clock with frequency upto 200MHz.
- If required a bit in the hyperbus register can be written to, to make this the system clock.

Shire External Resets

Reset	Description	Register Programming
reset_cold	Power on reset	
reset_warm	Retains state: Does not reset ESR, VC FIFOs etc.	SoftReset.cpu_warm_reset = 0
dmctrl.ndmreset	Resets all non-debug logic	Handled by CPU Registers
bpam_run_control.g	Internal to Shire	
dmctrl.dmactive	debug logic reset.	Handled by CPU Registers

Power Domains.

The device contains the following power domains.

- Always On.
- System Domain.

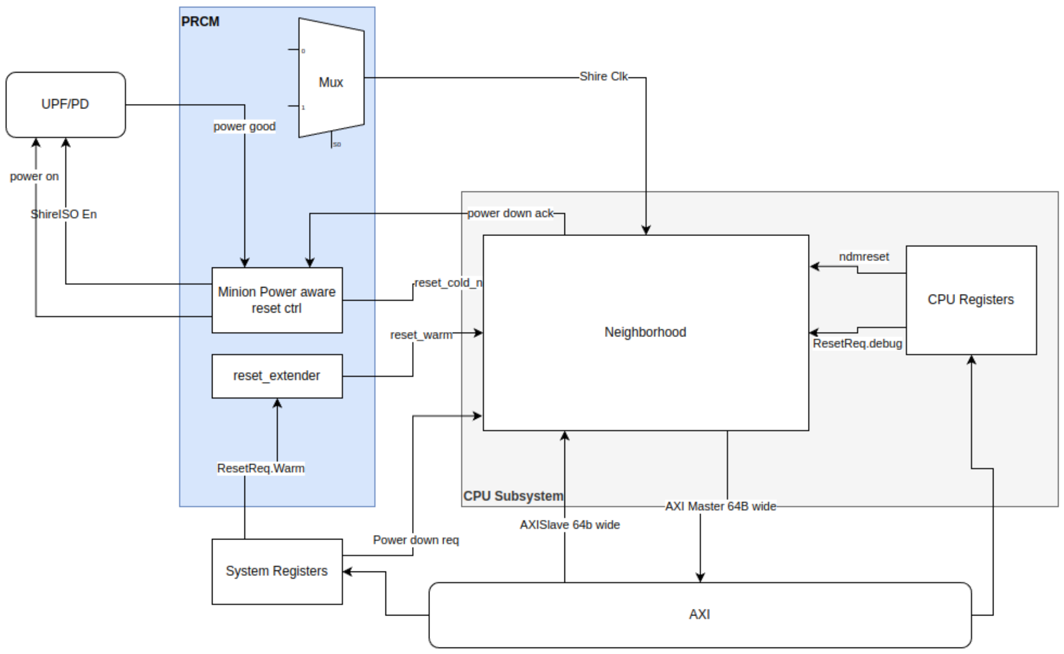


Figure 5: Shire Integration

- CPU Subsystem Domain.
- Per Minion Power Domain.
- MRAM Digital Domain
- Each Memory (Bootrom, SRAM and MRAM) has its own power domains.
- Each powerdomain is controlled via a power switch.
- At Boot every Digital Domain powerswitch is turned on.
- The boot routine will check the OTP register for domains that need to be powered down after boot and write to the corresponding system_registers.PowerDownReq bits. ## Valid Power State

Simple Power States

State	Sys PS	MRAM Digital PS	MRAM PS	cpu subsystem PS	ROM/ SRAM PS	minion PS[7:0]	Notes
All On	On	On	On	On	On	On	

State	Sys PS	MRAM Digital PS	MRAM PS	cpu subsystem PS	ROM/ SRAM PS	minion PS[7:0]	Notes
System Off	Off	Off	Off	Off	Off	Off	Extreme low power. Toggle TMS to wake the device
CPU Off	On	X	X	Off	X	Off	
MRAM Off	On	Off	Off	X	X	X	
1-8 Minions Off	On	X	X	On	On	On/Off	
Controlling PowerDomainReq register	system poweroff	mram pd	mram dsleep en	cpu pd	sram pd	minion pd_[0-7]	

Note:

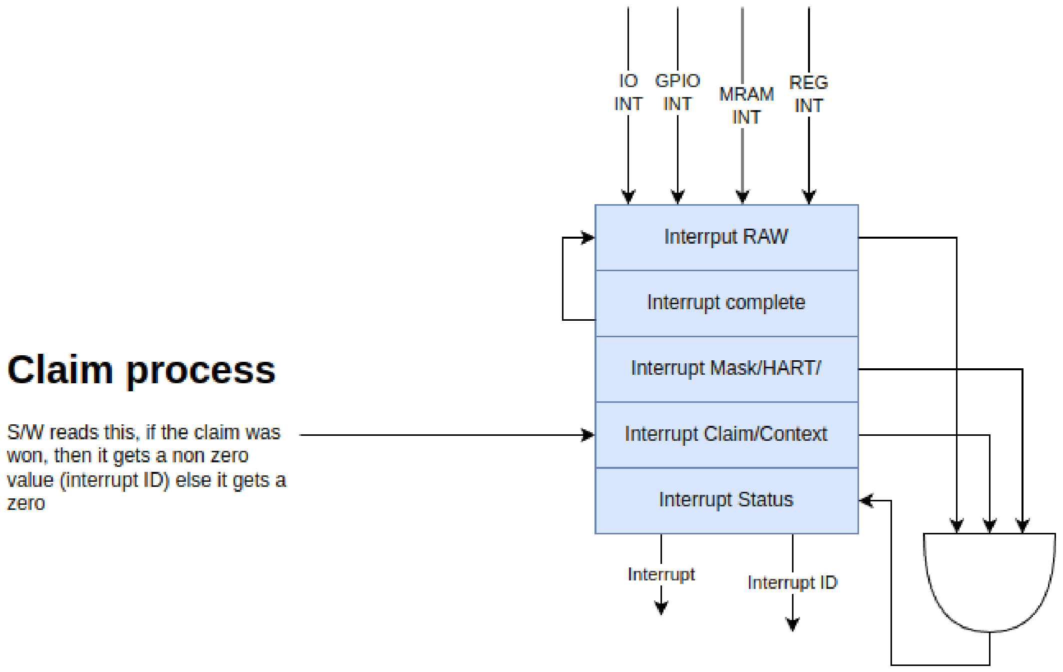
- X: Dont care. The domain is usually on. But can be off if a complex Power State is used.
- Other than the system off state all other power switches can be turned on by clearing the corresponding pd_req bit.
- Complex Power States are:
  - CPU OFF + MRAM OFF
  - Minion_x OFF + MRAM OFF

Power State Controller FSM.

- At boot the FSM is in normal state.
- Writing to pd_req initiates the power down sequence.
- The corresponding domain is informed about the request.
- The domain completes the current outstanding transaction and frees up the system bus and goes into a halt state.
- Domain generates the PD ack signal.

- PD ack signal triggers the Power State Controller which executes the following state changes:
  - ISO_EN State: drives the isolation enable signal to the isolation cells placed on the output wires of the power domain.
  - POWEROFF State: turns off the power switch and remains in this state until pd req goes low, This can be due to:
    - ⋆ Toggling of TMS signal in case of system power off
    - ⋆ Clearing the corresponding PD Register bit.
  - When the pd req goes low this state turns on the power switch and transitions to WTGood State.
  - WTGOOD State: It takes some time to ramp up the voltage from 0 to VDD. In this state:
    - ⋆ A counter is initialized with the system_registers.PowerGood.counter register value and counts down to zero.
    - ⋆ State changes to ISO_OFF when the counter is zero.
    - ⋆ soft_reset_req is asserted and stays asserted until we enter the NORMAL state.
  - ISO_OFF: the isolation enable signal to the isolation cells is de-asserted.
    - ⋆ Note: It is expected the reset values of the domain output signals do not initiate or accept any transaction.
  - RESET: We stay in this state for the number of cycles required to propagate the reset to all sub modules in the power domain.
  - NORMAL: Normal operation resumes.

Interrupts



Claim process

S/W reads this, if the claim was won, then it gets a non zero value (interrupt ID) else it gets a zero

Figure 6: Interrupts

There are two sets of interrupts, Timer and S/W interrupts are described in the minion specification.

This chapter describes the system Interrupts.

The system Interrupts are generated from the following endpoints

- MRAM Interrupt: Raised by MRAM on ECC failure, to trigger a wellness check
- GPIO Interrupt: Raised by the GPIO module when any of the input signal changes state.
- Register Interrupt: A specific bit in the system register that can be written to to trigger an interrupt,e.g. this can be used by xSPI to interrupt the CPU and process the mailbox.
- Periph Interrupt: These interrupts are generated by QSPI,UART, I2C and xSPI Hardware to indicate state changes like TxDone, RxDone, FIFO under/over flow etc.

CPU Handling of system interrupts

- The system interrupt hardware follow the RISC-V PLIC architecture. The CPU handles the interrupts in Machine mode.
- PLIC MMIO Space consists of the following registers

Register	Count	Purpose
Pending	1bit/interrupt source	Set when source generates an interrupt. input could be level or pulse. Output is level
Mask	1 Reg/HART	The Pending register is Anded with the mask and the result is or'ed and sent to the external interrupt pin of the corresponding core
Claim	1 Reg/HART	A read from this register returns the highest priority interrupt id available for this endpoint.
Complete	1bit/interrupt source	Write of interruptid to this register clears the corresponding interrupt
Priority	2bit/interrupt source	Sets the priority for the interrupt

Interrupt handling process.

This is a short overview. Refer to the RISCV privileged ISA for details

- A interrupt from a peripheral sets the corresponding bit in the Pending Register.
- This is ANDed with the per HART mask register and added to the interrupt queue of the HART.
- If the Queue is not empty interrupt signal is sent to to minion.
- If we want to pin down a specific interrupt to a specific minion, we will have to clear its mask bit in all other minions mask registers.
- Multiple interrupts can be outstanding at any given time.
- Each HART that received the interrupt will attempt to read its Claim register. There is one Claim register per HART.
- The PLIC will:
    - Check what is the highest outstanding interrupt in the Queue for this HART and return its ID during the claim read.
    - Remove this interrupt from the Queue of other HART's

- After servicing the interrupt the minion will write to the Complete register. On Write the PLIC will clear the Pending bit.

Memory Map

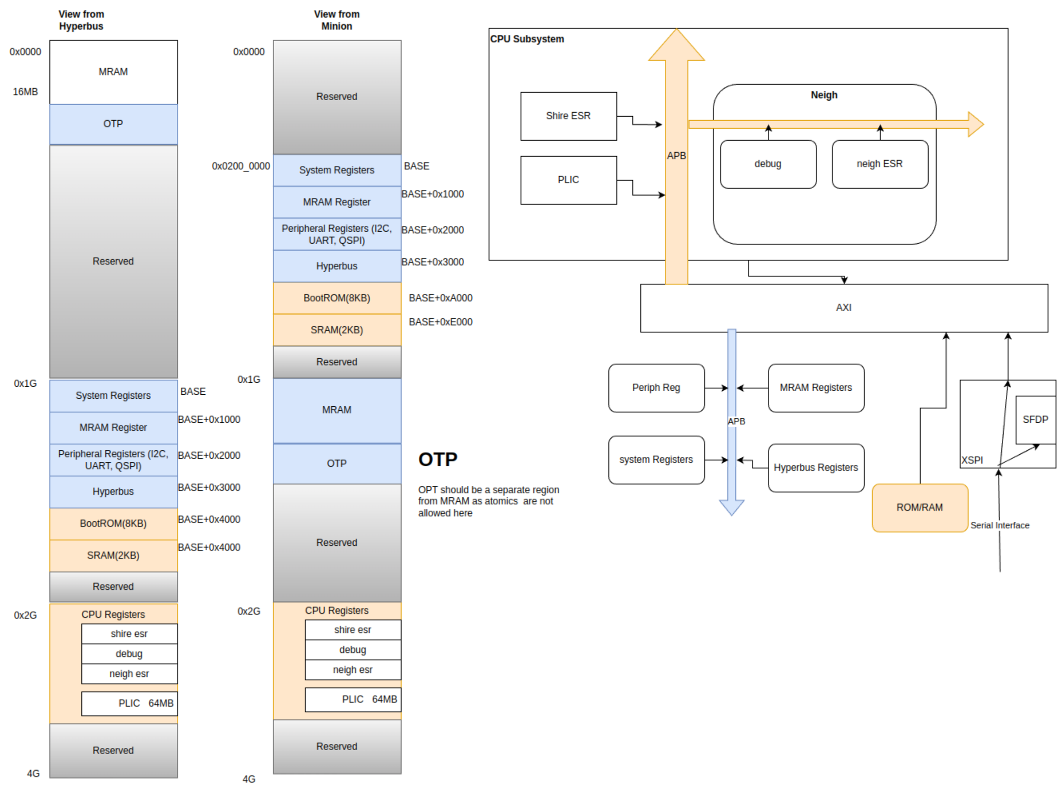


Figure 7: Busmatrix Pathways

Interconnect: Transaction Initiators and Targets

There are 3 Major Transaction Initiators in the System.

1. Hyperbus
2. CPU
3. Chiplet Interface(Not present in Erbium).

Initiator	Interface	Targets Accessible
Hyperbus	AXI4	MRAM, SRAM, System Registers, Peripheral Registers, HyperBusRegisters, CPURegisters, MRAMRegisters
CPU	AHB-Lite	MRAM, SRAM, Bootrom, System Registers, Peripheral Registers, HyperBusRegisters, CPURegisters, MRAMRegisters
Chiplet	AXI4	MRAM, SystemRegisters

This device can be used as either an Edge AI device or as a simple Flash replacement. The memory Map seen by CPU and UART (Edge AI Mode) is different than the memory map seen by xSPI(Flash replacement device)

CPU and UART Memory Map.

Target	Protocol at NIC	Data Width	Start Address	Size
SystemRegister	APB4	32	0x0200_0000	0x1000
MRAMRegiste	APB4	64	0x0200_1000	0x1000
PeriphRegister	APB4	32	0x0200_2000	0x1000
HyperbusRegis	APB4	32	0x0200_3000	0x1000
Bootrom	AXI4	64	0x0200_A000	0x2000
SRAM	AXI4	64	0x0200_E000	0x0800
MRAM + OTP	AXI4	512	0x4000_0000	0x4000_0000
CPURegisters	APB4	32	0x8000_0000	0x4000_0000

Hyperbus Memory Map.

Customers accessing Erbium through xSPI interface are expected to use it as a MRAM Memory device. In such a case it is natural to assume the memory starts from address 0. Hence in Hyperbus case the addressmap is rearranged to facilitate this world view.

Target	Protocol at NIC	Data Width	Start Address	Size
MRAM + OTP	AXI4	512	0x0000_0000	0x4000_0000
System Register	APB4	32	0x4200_0000	0x1000
MRAM Register	APB4	64	0x4200_1000	0x1000
Peripheral Register	APB4	32	0x4200_2000	0x1000
Hyperbus Register	APB4	32	0x4200_3000	0x1000
Bootrom	AXI4	64	0x4200_A000	0x2000
SRAM	AXI4	64	0x4200_E000	0x0800
CPU Registers	APB4	64	0x8000_0000	0x4000_0000

Note All Register address spaces are 64 bit aligned.

xSPI Datasheet

- xSPI bus interface supporting
  - 1S-1S-1S (SPI)
  - 4S-4D-4D
  - 4D-4D-4D
  - 8D-8D-8D (Octal profile 2)
  - 8D-8D-8D (Hyperbus)
- While other combinations are possible, only the above configurations are tested during verification. e.g.
  - 1S-4S-4S (Quad)
  - 4S-4S-4S (Quad)
- Boot configuration SPI, Quad, Octal, Hyperbus.
- Hardware Reset.
- Deep Powerdown mode
- Based on JESD251C, JESD215C-1, JESD216H

Overview.

Commands

Hyperbus Mode

- 8D-8D-8D mode.

SPI, Quad SPI, Octal SPI Mode

Commands	Code	1S-1S-1S	4S-4D-4D	8D-8D-8D(Octal)
Read SFDP	5Ah	0.E	1.B	1.B
Read (0L)	NA	NA	NA	NA
Read Register	65	0.D	1.B	1.B
Write Register	71	0.K	1.D	1.D
Enter Power Down	B9h	0.A	1.A	1.A
Exit Power Down	ABh	0.A	1.A	1.A
Reset Device and Enter default mode(1S)	99h	0.A	1.A	1.A
Read Memory	0Bh	0.F	1.B	1.B

Commands	Code	1S-1S-1S	4S-4D-4D	8D-8D-8D(Octal)
Write Memory	02h	0.K	1.D	1.D

Note

- The 1.X commands follow the pattern of CMD-Ext, 4B Address, X cycles of latency, Data
- The 0.X commands do not have Ext, Spec allows #b or 4B address, we implement only 4B address.
- For Register Transaction the Data is 32 bit.
- For Memory Transaction Data is 64 bit.
- JEDEC has messed up the QuadSPI Standard.
    - Section 4 specifies 1.X format and Section 5 Specifies 3.X format which is similar to 0.X formats.
    - We will go with 1.X format specified in Section 4.
- for 1S mode we will use the equivalent 0.X format,
- All address formats use 4B address.
- We will not implement read zero latency as that will not work with our pipeline.
- Supported(Tested) Modes are (ref 6.10.18 of SFDP)
    - 1S-1S-1S
    - 8D-8D-8D
    - 8S-8S-8S
    - 4S-4D-4D
    - 4S-4S-4S
- Other Modes are possible, e.g. 4S-1S-4D (Why?) but not tested. If there is interest in an SPI mode other than the 3 listed above, please discuss.

Not supported in Phase 1

Mode	Plan
Wrapped bursts	(Phase 2)
XIP	(Phase 2)
Write Enable	Not Supported

Mode	Plan
Page/Sector Erase/Program	Not Supported

Octal Mode

We have a requirement of supporting the following devices in the production chip

1. SPI
2. QuadSPI
3. OctalSPI
4. Hyperbus

System_Reg address map

- Absolute Address: 0x0
- Base Offset: 0x0
- Size: 0x8C

Register Controlling System Behavior

Offset	Identifier	Name
0x00	Version	—
0x08	SystemConfig	—
0x10	WATCHDOG_COUNT	—
0x18	Watchdog	—
0x20	SysInterrupt	—
0x28	SoftReset	—
0x30	ResetCause	—
0x38	PowerDomainReq	—
0x40	PowerDomainAck	—
0x48	PowerGood	—
0x50	SpinLock	—
0x58	ChipMode	—
0x60	Mailbox0	—
0x68	Mailbox1	—
0x70	GPIO_OE	—
0x78	GPIO_I	—
0x80	GPIO_O	—
0x88	GPIO_Interrupt_Enable	—

Version register

- Absolute Address: 0x0
- Base Offset: 0x0
- Size: 0x4

Device identifier, used by software to identify the device family (chipid) variant, and bugfix version

Bits	Identifier	Access	Reset	Name
7:0	respin	r	0x0	—
15:8	variation	r	0x0	—
31:16	chipid	r	0xEB68	—

SystemConfig register

- Absolute Address: 0x8
- Base Offset: 0x8
- Size: 0x4

System configuration fields. use to enable/disable various features

Bits	Identifier	Access	Reset	Name
0	sys_interrupt_enable	rw	0x0	—
1	mram_startup_bypass	rw	0x0	—
2	wdog_disable	rw	0x1	—
3	i2c_enable	rw	0x0	—
4	spi_enable	rw	0x1	—
5	qspi_enable	rw	0x0	—
6	uart_enable	rw	0x0	—

sys_interrupt_enable field reg interrupt:Writing to this bit generates an interrupt.

mram_startup_bypass field connected to mram_startup_bypass of mram_wrapper

wdog_disable field Watchdog Disable

i2c_enable field I2C Enable

spi_enable field SPI Enable

qspi_enable field QSPI Enable

uart_enable field UART Enable

WATCHDOG_COUNT register

- Absolute Address: 0x10
- Base Offset: 0x10
- Size: 0x4

The watchdog detects 'hang' conditions and resets the system. This feature is disabled by default. Clear cpu_config.wdog_disable to enable this. Once enabled, S/W should write to Watchdog.kick to reset the WATCHDOG_COUNT.count field. If the count reaches 0, it triggers a system reset.

Bits	Identifier	Access	Reset	Name
31:0	watchdog_count	rw	0xFFFF	—

watchdog_count field When the watchdog timer is enabled it counts down from this value

Watchdog register

- Absolute Address: 0x18
- Base Offset: 0x18
- Size: 0x4

Bits	Identifier	Access	Reset	Name
7	kick	rw	0x0	—

kick field Resets the watchdog timer, cpu needs to regularly write to this bit to aviod a watchdog timeout based reset of the device

SysInterrupt register

- Absolute Address: 0x20
- Base Offset: 0x20
- Size: 0x4

Bits	Identifier	Access	Reset	Name
0	interrupt	rw	0x0	—

interrupt field Write to this bit to generate an interrupt

SoftReset register

- Absolute Address: 0x28
- Base Offset: 0x28
- Size: 0x4

Bits	Identifier	Access	Reset	Name
0	soft_reset	rw	0x0	—
1	cpu_warm_reset	rw	0x0	—
2	mram_rst_b	rw	0x1	—

soft_reset field System Soft Reset, Active High

cpu_warm_reset field CPU Warm Reset, Active High

mram_rst_b field MRAM Resetn, Active Low

ResetCause register

- Absolute Address: 0x30
- Base Offset: 0x30
- Size: 0x4

This register reports the cause of reset. There are multiple reset sources. * Power on Reset, * Brownout Reset, and * Various S/W reset requests. These clear on read bits capture the reset cause since the last read Note if the por bit is set ignore the other cause registers, they will have random values until the first read.

Bits	Identifier	Access	Reset	Name
0	por	r, rclr	0x1	—
1	watchdog_timedout	r, rclr	—	—

Bits	Identifier	Access	Reset	Name
2	sysreset_req	r, rclr	—	—
3	brownout	r, rclr	—	—
4	softreset	r, rclr	—	—
5	hresetn	r, rclr	—	—

por field System POR was toggled

watchdog_timedout field Watchdog was enabled and CPU failed to clear the watchdog timer

sysreset_req field CPU detected an architecture level lockup and requested a system reset

brownout field The brownout detector triggered a reset

softreset field The soft reset bit was written to

hresetn field The cpu reset bit was written to

PowerDomainReq register

- Absolute Address: 0x38
- Base Offset: 0x38
- Size: 0x4

Writing to PD fields initiates the shutdown of the corresponding power domain. A shutdown request is sent to the domain, This domain waits until all outstanding transactions are completed and then generates a corresponding ack signal. Once the ack signal is generated the power controller will turn off power to that domain.

Bits	Identifier	Access	Reset	Name
0	cpu_pd	rw	0x0	—
1	sram_pd	r	0x0	—
2	cpu_ram_powerdown	rw	0x0	—
3	chiplet_pd	r	0x0	—
4	mram_pd	rw	0x0	—

Bits	Identifier	Access	Reset	Name
5	system_poweroff	w	0x0	—
6	hyperbus	r	0x0	—
15:8	minion_pd	rw	0x0	—
16	mram_dsleep_en	rw	0x1	—
17	cpu_sleep_en	rw	0x0	—

cpu_pd field Poweroff the ARM Domain

sram_pd field Not used, in future connected to TCM poweroff bit

cpu_ram_powerdown field Powerdown the CPU ram. this puts the ram in deepsleep mode

chiplet_pd field Not used; chiplet power domain is controlled via the mode bits

mram_pd field Power down for MRAM digital logic

system_poweroff field Power of the chip.only wakeup logic is powered on

hyperbus field Not used; hyperbus power domain is controlled via the mode bits.

minion_pd field Minion PowerDown Req

mram_dsleep_en field connected to dsleep pin on mram_wrapper

cpu_sleep_en field currently not used. In future this will be used for CPU sleep management.

PowerDomainAck register

- Absolute Address: 0x40
- Base Offset: 0x40
- Size: 0x4

This mirrors the Ack signal generated in response to power down request.

Bits	Identifier	Access	Reset	Name
0	cpu_pd_ack	r	0x0	—
1	sram_pd_ack	r	0x0	—
3	chiplet_pd_ack	r	0x0	—
4	mram_pd_ack	r	0x0	—
5	system_pd_ack	r	0x0	—
6	hyperbus_pd_ack	r	0x0	—
15:8	minion_pd_ack	rw	0x0	—

cpu_pd_ack field pd ack for cpu

sram_pd_ack field Not used;pd ack for TCM

chiplet_pd_ack field Not used;pd ack for chiplet

mram_pd_ack field pd ack for mram

system_pd_ack field pd ack for system

hyperbus_pd_ack field Not used;pd ack for hyperbus

minion_pd_ack field Minion PowerDown Req

PowerGood register

- Absolute Address: 0x48
- Base Offset: 0x48
- Size: 0x4

When a power domain is switched on, it takes time for the voltage to stabalize, this time is process dependent. The default value is sufficiently large to account for all variations.

Bits	Identifier	Access	Reset	Name
20:0	counter	rw	0xFFFFF	—

counter field Counter for powerGood

SpinLock register

- Absolute Address: 0x50
- Base Offset: 0x50
- Size: 0x4

initially locked=0, A read on this register will set the lock bit. Usage:

1. Read this field. If you get a value of zero you got the lock. Else a different processing element acquired the lock. Poll at fixed intervals(dont spam) until you get the lock.
2. If you acquired the lock, Once you have finished interacting with the locked resource write 0 to this register to release the lock.

Bits	Identifier	Access	Reset	Name
0	lock	rw, rset	0x0	—

ChipMode register

- Absolute Address: 0x58
- Base Offset: 0x58
- Size: 0x4

Bits	Identifier	Access	Reset	Name
1:0	chip_mode	r	—	—
2	ifc_width	r	—	—
4:3	bootload	rw	0x0	—
6:5	load_external	rw	0x0	—

chip_mode field The mode in which chip is working, hyperbus, axi,ahb,gci

ifc_width field If chip is axi/ahb mode datawidth

bootload field Jump to 00:no Jump, 01:TCM,10:MRAM

load_external field Jump to 00:no load, 01:Load TCM via Chiplet,10:Load MRAM via Chiplet

Mailbox0 register

- Absolute Address: 0x60
- Base Offset: 0x60
- Size: 0x4

Bits	Identifier	Access	Reset	Name
31:0	mbox0	rw	0x0	—

mbox0 field Mailbox0

Mailbox1 register

- Absolute Address: 0x68
- Base Offset: 0x68
- Size: 0x4

Bits	Identifier	Access	Reset	Name
31:0	mbox1	rw	0x0	—

mbox1 field Mailbox1

GPIO_OE register

- Absolute Address: 0x70
- Base Offset: 0x70
- Size: 0x4

Gpio Output enable. Write 1 to this bit to set the GPIO register in output mode.

Bits	Identifier	Access	Reset	Name
10:0	gpio_oe	rw	0x0	—

gpio_oe field Gpio output enable

GPIO_I register

- Absolute Address: 0x78
- Base Offset: 0x78
- Size: 0x4

Gpio Input For each bit in input mode this register captures the input value.

Bits	Identifier	Access	Reset	Name
10:0	gpio_i	r	0x0	—

gpio_i field Gpio input

GPIO_O register

- Absolute Address: 0x80
- Base Offset: 0x80
- Size: 0x4

Gpio Output For each bit in output mode the content of the corresponding bit in this register is reflected on the GPIO Pin

Bits	Identifier	Access	Reset	Name
10:0	gpio_o	r	0x0	—

gpio_o field Gpio input

GPIO_Interrupt_Enable register

- Absolute Address: 0x88
- Base Offset: 0x88
- Size: 0x4

Gpio Interrupt Enable For each bit in input mode if the input signal toggles and the content of the corresponding bit in this register is 1, then a GPIO interrupt is raised.

Bits	Identifier	Access	Reset	Name
10:0	gpio_interrupt_en	r	0x0	—

gpio_interrupt_en field Gpio input