M3 Processor Layer Family (Version 18) Documentation (PRCv18, PRCv18G, PREv18, PREv18A, PREv18E)

Revision 1.1

Yejoong Kim[*1], Zhiyoong Foo[†1], Gyouho Kim[‡1], Dongmin Yoon[§1], Ruochen Xie[¶1], Xun Sun[‖1], and Taekwang Jang[**1]

[1]Michigan Integrated Circuits Laboratory, University of Michigan, Ann Arbor

March 15, 2019

[*]yejoong@umich.edu
[†]zhiyoong@umich.edu
[‡]gyouhokim@umich.edu
[§]dmyoon@umich.edu
[¶]xierc@umich.edu
[‖]xusun@umich.edu
[**]tkjang@umich.edu

Contents

1 Layer Revision History

1.1 PRC/PREv9 Family

- PRCv9/PREv9 are bug-fix versions of the previous PRC/PRE layers.
- Die size: 1050um x 2080um (PRCv9), 1300um x 3600um (PREv9)

1.2 PRC/PREv10 Family

- PRCv10 has a 4kB SRAM (10T, logic-rule), instead of the 3kB SRAM (10T, logic-rule) used in previous PRC layers.
- Everything else remains same as in PRCv9.
- There is no PREv10.
- Die size: 1050um x 2352um (PRCv10)

1.3 PRC/PREv11 Family

- For MBus CIN pad, a Schmitt-Trigger Digital Input pad (PAD_50x60_DIN_SCHTRG_TSMC180) replaces the previous simple Digital Input pad (PAD_50x60_DIN_TSMC180).
- Missing double-latches between MBus_node and layer_ctrl have been added.
- Layer Controller checks whether TX_ACK becomes 0, before asserting TX_REQ.
- Die size: 1050um x 2350um (PRCv11), 1300um x 3600um (PREv11)
- Tape-out Date: July 8, 2015

1.4 PRC/PREv12 Family

- First PRC/PRE versions using CPF flow.
- Previous PMU and BOD have been removed. GOC_CLK_GEN and DSLP_CLK_GEN have been newly made. DSLP_CLK_GEN has 1 tuning bit to support a faster sleep clock.
- A 16-bit timer, a 32-bit timer, and a watchdog timer have been added. Watchdog timer triggers PMU hard-reset; PAD_WD_IRQ must be connected to a reset pad in a separate PMU layer.
- Memory Map has been entirely changed and mostly NOT backward-compatible.
- 8 MBus registers reside in Register File, which replace the previous message registers. All of them can be configured to generate an interrupt when written. Previous message register addresses (`0x14` `0x17`) can be still used if configured properly.
- Arbiter has been designed to support arbitrary-length MBus messages as well as a low-latency & low-overhead 32-bit MBus message Tx. More IRQ and CPU_HALT options have been implemented.
- MPQ registers have been added to support 4 Memory Streaming channels and Memory Bulk Write. (Action Registers are not implemented)
- 8kB SRAM (7T, pushed-rule) replaces the previous 4kB SRAM.

- GOC and MBus flag registers have been added.
- (PREv12 only) A Clock Driver for driving Crystal Oscillator is added.
- (PREv12 only) 2 more Power Switches have been added, resulting in total 3 Power Switches.
- (PREv12 only) The number of bits in GPIO has been reduced to 8.
- Die size: 1050um x 2080um (PRCv12), 1300um x 2500um (PREv12)
- Tape-out Date: July 8, 2015

1.5 PRC/PREv13 Family

- Various bug fixed versions
- Arbiter has been newly designed to provide a unified memory map to GOC, CPU, MBus.
- (PREv13 only) The number of bits in XO_TIMER increases to 32-bits.
- (PREv13 only) PAD_WD_IRQ can be configured to output a divided crystal clock.
- Die size: 1050um x 2080um (PRCv13), 1300um x 2500um (PREv13)
- Tape-out Date: December 23, 2015

1.6 PRC/PREv14 Family

- MBus Release 03
- Various bug fixed versions
- Metal stack option changed from 1p6m4x1u40k2ff (40k UTM Top-Metal) to 1p6m4x1n8k2ff (8k Standard Top-Metal)
- Added MBus watchdog timer
- MBus Full Prefix has been updated according to the new rules.
- GOC parity has been replaced with checksum.
- (PREv14 only) 8-bit GPIO pads can be individually enabled/disabled.
- (PREv14 only) Separated XO clock output pad and Watchdog output pad
- Die size: 1050um x 2080um (PRCv14), 1300um x 2500um (PREv14)
- Tape-out Date: June 20, 2016

1.7 PRC/PREv15 Family

- MBus Release 04
- Added a new layer, PRCv15G, which has a new GOC AFE, designed by Wootaek Lim.
- Made a transition in APR tool: Encounter 10.1 $\rightarrow$ Innovus 15.21
- MBus CIN glitch fixed (changed the logic in mbus_master_ctrl regarding mbus_busy_b signal)

- Added MBus Flag to indicate whether it has been interrupted by GOC/EP
- Added HALT_UNTIL_MBUS_TRX option
- Added GOC/EP interrupt port in Cortex-M0
- Reduced the number of MBus memory streaming channels to 2 (from 4)
- Layer Controller can now access Timers and GPIO/SPI (PREv15 only) as well
- (PRCv15, PREv15 only) Changed GOC/EP default clock frequency
- Die size: 1050um x 2080um (PRCv15, PRCv15G), 1300um x 2500um (PREv15)
- Tape-out Date: December 28, 2016

1.8 PRC/PREv16 Family

- MBus Release 04
- Made fiducials exposed so that they can be clearly seen during automated assembly process
- MBus Full Prefix is same as in PRC/PREv15 Family.
- There is no PRCv16G.
- Die size: 1050um x 2080um (PRCv16), 1300um x 2500um (PREv16)
- Tape-out Date: June 12, 2017

1.9 PRC/PREv17 Family

- MBus Release 04p1
  - Soft Reset support in Layer Controller
  - MBUS_BUSY isolation moved into mbus_master_ctrl
  - Explicit isolation of SLEEP_REQ
  - Added FORCE_IDLE_WHEN_DONE mode in mbus_mater_node_ctrl to prevent any timing issue between the master node's MBus clock and a member node's Layer Controller clock.
- Used a new Pad library (ESD_PAD_TSMC180_rev1)
- Cortex-M0 interrupt port re-mapped with 2 new IRQs added (Soft Reset, Wake-up)
- Added FLAGS in MBus Register File
- Replaced the DSLP_CLK_GEN. Now it is running at VDD_1P2.
- Re-designed Arbitration (ARB)
  - Now offers detailed waveform descriptions and testbench
  - Optimized MBus IRQ Generation (MBUS_TX_IRQ, MBUS_RX_IRQ, etc) to minimize the number of waiting cycles
  - Added MBUS_TX_IRQ_AFTER_CLR_INT setting for more conservative CPU halt/resume
  - Added Soft Reset
  - Added WAKEUP_IRQ and WAKEUP_SOURCE identifier; Now CPU provides a synchronized reset going into ARB (M0_HRESETn) to be used for generating WAKEUP_IRQ.

  - Added double-latches to support Read Access for WUP_TSTAMP_VAL and XOT_TIMER_VAL
  - Now use 32-bit address space (2 LSBs are fixed at 0) when the Layer Controller accesses the memory, to support full MMIO access
  - Layer Controller can access MBus Register File through MMIO
  - LC_MEM_ACK is delayed by 1 cycle when read from SPI to handle SPI_HREADY=0
  - Added PERI_HREADY which is always 1 when not serving CPU
  - Removed second_arb_cpu
  - Added SYS_CTRL_CMD_RESUME_CPU and made it accessible by others (CPU, MBus, GOC/EP)
  - Added a system register file (ARB_SYS_REG) module which includes CONFIG_HALT_CPU, MBUS_TX_IRQ_AFTER_CLR_INT, and SET_WUP_SRC.

- Re-designed mbus_isolation
  - Used generate blocks to simplify the behavioral Verilog
  - Added GPIO_WAKEUP_REQ and corresponding logic
  - Added flip-flops to store wake-up history
  - Added more explicit reset for the always-on flip-flops
  - Added a flip-flop to store RUN_M0 at the start of GOC/EP

- Re-designed GOCEP
  - GOC_CLK and GOC_DATA are double-latched; they were single-latched in previous versions.
  - RUN_M0 is replaced by RUN_M0_SAMPLED (coming from mbus_isolation)
  - Added M0_HRESETn input so that GEN_IRQ can wait until CPU starts running before generating an M0 IRQ.

- Re-designed TIMER
  - In TIMER32, cnt_reg keeps changing beyond cmp_reg, rather than staying at cmp_reg
  - In TIMER32 and TIMER16, IRQ is changed to become high only for one clock cycle

- (PRCv17G only) GOC_AFE Base clock speed became faster by default.
- Die size: 1050um x 2080um (PRCv17, PRCv17G), 1300um x 2500um (PREv17)
- Tape-out Date: June 12, 2017

1.10 PRC/PREv18 Family

- MBus Release 05
  - Now supports ”pending wakeup request”
  - Now Layer Controller handles RX_REQ only when RX_BROADCAST is 0. It provides RX_REQ_SYNC regardless of RX_BROADCAST.

- Added a new layer, PREv18E, which has two copies of 8kB SRAMs (16kB in total).
- Added a new layer, PREv18A, which is identical to PREv18 but with a new pad library.
- Removed SRAM_USE_VREF_0P6 signal; now VREF is fixed at VDD_1P2.
- Updated SRAM version (RSRAM7Tv775s8kBx32m2Mar2018)

- Moved 'System Configuration' MBus Register from `0x1A` to `0x1B`
- Moved SRAM into ARB for better clock tree synthesis
- Uses a wrapper module for SRAM ISOL LC for better drivability
- Moved custom blocks into m3_custom_tsmc180 and m3_custom_PRC_tsmc180
- Increased the size of WUP_TIMER counter (15 bits $\rightarrow$ 22 bits); bit assignments of WUP_TIMER MBus Register has been also changed.
- Updated the LC/MBus Clock Generator so it has glitch-free dividers and more wide-range of frequencies.
  MASTER_CLK_GEN_V1_TSMC180 $\rightarrow$ MASTER_CLK_GEN_V2_TSMC180
- Added power gating for the new XO Driver
- Added PUF Chip ID
- (PRCv18G only) Replaced the previous GOC_AFE with GOC_AFE_V2, designed by Seok Hyeon Jeong.
- (PRCv18G only) Added GOC_DBE & GOCEP Error Handling (MAIN_FAIL, MAIN_FAIL_TYPE, MAIN_FAIL_ACK)
- (PREv18, PREv18A, PREv18E only) Moved 'XO Driver' MBus Register from `0x0C` to `0x19` and `0x1A`
- (PREv18, PREv18E only) Updated XO Driver.
  XO_DRV_V1_TSMC180 $\rightarrow$ XO_DRV_V2A_TSMC180
- (PREv18A only) Updated XO Driver.
  XO_DRV_V1_TSMC180 $\rightarrow$ XO_DRV_V3_TSMC180
- (PREv18, PREv18A, PREv18E only) Added support for N2N layer's clock requirement (XO_TIMER)
- (PREv18, PREv18A, PREv18E only) VDD_COTS and COTS power switches' pad locations were changed for better routability on PCB.
- Die size: 1050um x 2080um (PRCv18, PRCv18G), 1300um x 2500um (PREv18), 1350um x 3100um (PREv18E)
- Tape-out Date: June 20, 2018 (PRCv18, PRCv18G, PREv18, PREv18E)
- Tape-out Date: July 25, 2018 (PREv18A)

2 Layer Description

The Processor Layer Family (PRC/PREv18 Family) is the MBus master layer, and contains an ARM Cortex-M0, a 16-bit timer, a 32-bit timer, and Watchdog timer.

PRC/PREv18 Family includes the following five layers. All layers have 8kB SRAM, except PREv18E, which has two 8kB SRAMs (16kB in total).

PREv18A is an incremental update from PREv18. PREv18 uses ESD_PAD_TSMC180_rev1 for its pads, while PREv18A uses ESD_PAD_TSMC180_rev2. ESD_PAD_TSMC180_rev2 fixes all latch-up and ESD errors occurred in ESD_PAD_TSMC180_rev1 and replaces an SVT SLC level converter with an HVT interrupted-DCVS, while circuit functionalities remain same. PREv18A also has an improved version of XO Driver, XO_DRV_V3_TSMC180. XO_DRV_V3_TSMC180 fixes some of the latchup errors found in the previous version (XO_DRV_V2A_TSMC180).

Table 1: PRC/PREv18 Family

Layer Name	Description
PRCv18	Processor Layer with the standard GOC AFE.
PRCv18G	Processor Layer with the new ambient-light-compensating GOC AFE.
PREv18	Extended processor layer: PRCv18 + SPI, GPIO, COTS switches, XO driver.
PREv18A	Extended processor layer: PREv18 with new pads and new XO driver layout
PREv18E	Extended processor layer with more memory: PREv18 + add. 8kB SRAM (total 16kB).

- Designed in TSMC 180nm (CR018G 1p6m4x1n8k2ff).
- Taped-out on June 20, 2018.
- MBus Full Prefix
  - PRCv18: `0x01012`
  - PRCv18G: `0x01212`
  - PREv18: `0x01112`
  - PREv18A: `0x01112`
  - PREv18E: `0x01112`
- Top-Level layout
  - PRCv18: `m3_hdk/virtuoso/TSMC180/PRCv18_TOP/PRCv18/layout`
  - PRCv18G: `m3_hdk/virtuoso/TSMC180/PRCv18G_TOP/PRCv18G/layout`
  - PREv18: `m3_hdk/virtuoso/TSMC180/PREv18_TOP/PREv18/layout`
  - PREv18A: `m3_hdk/virtuoso/TSMC180/PREv18A_TOP/PREv18A/layout`
  - PREv18E: `m3_hdk/virtuoso/TSMC180/PREv18E_TOP/PREv18E/layout`
- Top-Level LVS netlist
  - PRCv18: `m3_hdk/layer/PRC/PRCv18/cdl/PRCv18.cdl`
  - PRCv18G: `m3_hdk/layer/PRC/PRCv18G/cdl/PRCv18G.cdl`
  - PREv18: `m3_hdk/layer/PRE/PREv18/cdl/PREv18.cdl`
  - PREv18A: `m3_hdk/layer/PRE/PREv18A/cdl/PREv18A.cdl`
  - PREv18E: `m3_hdk/layer/PRE/PREv18E/cdl/PREv18E.cdl`
- Top-Level Spice netlist

  - PRCv18: `m3_hdk/layer/PRC/PRCv18/ckt/PRCv18.ckt`
  - PRCv18G: `m3_hdk/layer/PRC/PRCv18G/ckt/PRCv18G.ckt`
  - PREv18: `m3_hdk/layer/PRE/PREv18/ckt/PREv18.ckt`
  - PREv18A: N/A
  - PREv18E: `m3_hdk/layer/PRE/PREv18E/ckt/PREv18E.ckt`
- C header file
  - PRCv18: `m3_hdk/layer/PRC/PRCv18/verilog/genRF/PRCv18_RF.h`
  - PRCv18G: `m3_hdk/layer/PRC/PRCv18G/verilog/genRF/PRCv18G_RF.h`
  - PREv18: `m3_hdk/layer/PRE/PREv18/verilog/genRF/PREv18_RF.h`
  - PREv18A: N/A (identical to PREv18)
  - PREv18E: `m3_hdk/layer/PRE/PREv18E/verilog/genRF/PREv18E_RF.h`

3 Memory Map

This section describes the Memory Map of PRC/PREv18 Family. See Table 2. Every address is word-aligned, meaning that the 2 LSBs are always 0. Unspecified addresses are reserved, and accessing those addresses may lead to undefined behavior.

Table 2: Memory Map in PRC/PREv18 Family

Address	Description	Related Section	Remarks
0x00000000 ~ 0x00001FFC	8kB Retentive SRAM	Section 7	
0x00002000 ~ 0x00003FFC	8kB Retentive SRAM	Section 7	PREv18E Only
0x00004000	Reserved for internal use		
0x6C840000	Soft Reset	Section 6	
0xA0000000 ~ 0xA0000FFC	MBus Register File	Section 4	
0xA0001000 ~ 0xA00010FC	16-bit Timer	Section 10	
0xA0001100 ~ 0xA00011FC	32-bit Timer	Section 11	
0xA0001200 ~ 0xA00012FC	CPU Watchdog Timer	Section 5.6	
0xA0001300 ~ 0xA00013FC	Wakeup Timer	Section 13	
0xA0001400 ~ 0xA00014FC	XO Timer	Section 14	PREv18(A/E) Only
0xA0001500 ~ 0xA00015FC	MBus Watchdog Timer	Section 8.5.1	
0xA0002000 ~ 0xA0002FFC	MBus	Section 8	
0xA0003000 ~ 0xA0003FFC	Quick MBus	Section 8.5.3	
0xA0004000 ~ 0xA0004FFC	SPI	Section 20	PREv18(A/E) Only
0xA0005000 ~ 0xA0005FFC	GPIO	Section 21	PREv18(A/E) Only
0xA000A000 ~ 0xA000AFFC	System Register File	Section 5	
0xAFFFF000 ~ 0xAFFFFFFC	System Control	Section 5	
0xE000E100 ~ 0xE000E41C	Nested Vectored Interrupt Control (NVIC)	See Cortex-M0 Devices Generic User Guide	

4 MBus Register File

4.1 MBus Register File Mapping

Table 4 shows MBus Register File mapping information.

Some of the register signals and default values are valid only in a specific set of PRC/PREv18 Family. The 'Remark' section shows this information. See Table 3.

Table 3: Remark Keyword Definition

Keyword	Definition
Empty	Valid in all PRC/PREv18 Family (PRCv18, PRCv18G, PREv18, PREv18A, PREv18E)
PRCv18G	Valid only in PRCv18G.
PREv18E	Valid only in PREv18E.
PREv18(A/E)	Valid only in PREv18, PREv18A, and PREv18E.
Non-PRCv18G	Valid only in PRCv18, PREv18, PREv18A, and PREv18E.
Non-PREv18E	Valid only in PRCv18, PRCv18G, PREv18, and PREv18A.

Reg Addr	Bit Field	Reg Name	Property	Size & Reset	Remark
Register 0 (0x00, 0xA0000000) Default: 24'h000000					
0x00	[23:0]	MBUS_R0	W/R	24'h000000	
Register 1 (0x01, 0xA0000004) Default: 24'h000000					
0x01	[23:0]	MBUS_R1	W/R	24'h000000	
Register 2 (0x02, 0xA0000008) Default: 24'h000000					
0x02	[23:0]	MBUS_R2	W/R	24'h000000	
Register 3 (0x03, 0xA000000C) Default: 24'h000000					
0x03	[23:0]	MBUS_R3	W/R	24'h000000	
Register 4 (0x04, 0xA0000010) Default: 24'h000000					
0x04	[23:0]	MBUS_R4	W/R	24'h000000	
Register 5 (0x05, 0xA0000014) Default: 24'h000000					
0x05	[23:0]	MBUS_R5	W/R	24'h000000	
Register 6 (0x06, 0xA0000018) Default: 24'h000000					
0x06	[23:0]	MBUS_R6	W/R	24'h000000	
Register 7 (0x07, 0xA000001C) Default: 24'h000000					
0x07	[23:0]	MBUS_R7	W/R	24'h000000	
Register 8 (0x08, 0xA0000020) Default: 24'h000000					
0x08	[15:0]	GOCEP_CHIP_ID	W/R	16'h0000	
Register 9 (0x09, 0xA0000024) Default: 24'h880008					
0x09	[23]	MBUS_IGNORE_RX_FAIL	W/R	1'h1	
	[19:0]	MBUS_NUM_BITS_THRESHOLD	W/R	20'h80008	
Register 10 (0x0A, 0xA0000028) Default: 24'h000000					
0x0A	[23:0]	FLAGS	W/R	24'h000000	
Register 11 (0x0B, 0xA000002C) **(Non-PRCv18G Only)** Default: 24'h0D2788					
0x0B Non-PRCv18G	[21]	CLK_GEN_HIGH_FREQ	W/R	1'h0	Non-PRCv18G
	[20:18]	CLK_GEN_DIV_CORE	W/R	3'h3	Non-PRCv18G
	[17:15]	CLK_GEN_DIV_MBC	W/R	3'h2	Non-PRCv18G
	[14:13]	CLK_GEN_RING	W/R	2'h1	Non-PRCv18G
	[11:10]	GOC_CLK_GEN_SEL_DIV	W/R	2'h1	Non-PRCv18G
	[9:7]	GOC_CLK_GEN_SEL_FREQ	W/R	3'h7	Non-PRCv18G
	[6]	GOC_ONECLK_MODE	W/R	1'h0	Non-PRCv18G
	[5:4]	GOC_SEL_DLY	W/R	2'h0	Non-PRCv18G
	[3:0]	GOC_SEL	W/R	4'h8	Non-PRCv18G
Register 11 (0x0B, 0xA000002C) **(PRCv18G Only)** Default: 24'h0D2515					
Continued on next page					

Continued from previous page

Reg Addr	Bit Field	Reg Name	Property	Size & Reset	Remark
0x0B PRCv18G	[23]	GOC_DBG_ENABLE	W/R	1'h0	PRCv18G
	[22]	GOC_DBG_SEL	W/R	1'h0	PRCv18G
	[21]	CLK_GEN_HIGH_FREQ	W/R	1'h0	PRCv18G
	[20:18]	CLK_GEN_DIV_CORE	W/R	3'h3	PRCv18G
	[17:15]	CLK_GEN_DIV_MBC	W/R	3'h2	PRCv18G
	[14:13]	CLK_GEN_RING	W/R	2'h1	PRCv18G
	[11:10]	GOC_R_OFFSET	W/R	2'h1	PRCv18G
	[9:8]	GOC_MAIN_CLK_TUNE_DIV	W/R	2'h1	PRCv18G
	[7:6]	GOC_BASE_CLK_TUNE_DIV	W/R	2'h0	PRCv18G
	[5:4]	GOC_TRAIN_MAX_ERROR	W/R	2'h1	PRCv18G
	[3:2]	GOC_MAVG_THRESHOLD	W/R	2'h1	PRCv18G
	[1:0]	GOC_MF_THRESHOLD	W/R	2'h1	PRCv18G
Register 12 (0x0C, 0xA0000030) **(PRCv18G Only)** Default: 24'h4D0200					
0x0C	[23]	GOC_BASE_SEL_OSC_STAGE	W/R	1'h0	PRCv18G
	[22:21]	GOC_SEL_DIV_AZ	W/R	2'h2	PRCv18G
	[20]	GOC_AZ_ENB	W/R	1'h0	PRCv18G
	[19:18]	GOC_SIG_GAIN_B	W/R	2'h3	PRCv18G
	[17:15]	GOC_R_REF	W/R	3'h2	PRCv18G
	[14:12]	GOC_I_VBIAS	W/R	3'h0	PRCv18G
	[11:9]	GOC_I_AMBIENT	W/R	3'h1	PRCv18G
	[8:6]	GOC_I_SIG_REF	W/R	3'h0	PRCv18G
	[5:3]	GOC_I_CMP_1ST	W/R	3'h0	PRCv18G
	[2:0]	GOC_I_CMP_2ND	W/R	3'h0	PRCv18G
Register 13 (0x0D, 0xA0000034) Default: 24'h000000					
0x0D	[23:0]	PUF_CHIP_ID	R	24'h000000	
Register 14 (0x0E, 0xA0000038) Default: 24'h1E8480					
0x0E	[23:0]	GOCEP_TIMEOUT	W/R	24'h1E8480	
Register 15 (0x0F, 0xA000003C) Default: 24'h000000					
0x0F	[23]	GOCEP_FREEZE_RUN_CPU	W/R	1'h0	
	[14:13]	GOC_DBE_MAIN_FAIL_TYPE	W/R	2'h0	PRCv18G
	[12]	GOC_DBE_MAIN_FAIL	W/R	1'h0	PRCv18G
	[8]	GOCEP_CPU_WAS_RUNNING	W/R	1'h0	
	[7]	GOCEP_FAIL_PREMATURE	W/R	1'h0	
	[6]	GOCEP_FAIL_TIMEOUT	W/R	1'h0	
	[5]	GOCEP_FAIL_MEM_CHECKSUM	W/R	1'h0	
	[4]	GOCEP_FAIL_HEADER_CHECKSUM	W/R	1'h0	
	[3]	GOCEP_FAIL_CHIP_ID	W/R	1'h0	
	[2]	GOCEP_FAIL	W/R	1'h0	
	[1:0]	GOCEP_PASS	W/R	2'h0	
Register 16 (0x10, 0xA0000040) Default: 24'h000000					
0x10	[0]	RUN_CPU	W/R	1'h0	
Register 17 (0x11, 0xA0000044) Default: 24'h000000					
0x11	[23]	WUP_WREQ_EN	W/R	1'h0	
	[21:0]	WUP_CNT_SAT	W/R	22'h000000	
Register 18 (0x12, 0xA0000048) Default: 24'h000000					
0x12	[21:0]	WUP_CNT_VAL	R	22'h000000	
Register 19 (0x13, 0xA000004C) **(PREv18(A/E) Only)** Default: 24'h000000					
0x13	[23]	XOT_ENABLE	W/R	1'h0	PREv18(A/E)
	[22]	XOT_MODE	W/R	1'h0	PREv18(A/E)
	[21]	XOT_WREQ_EN	W/R	1'h0	PREv18(A/E)
	[20]	XOT_IRQ_EN	W/R	1'h0	PREv18(A/E)
	[19]	XOT_FORCE_COUT	W/R	1'h0	PREv18(A/E)
	[15:0]	XOT_CNT_SAT_LOWER	W/R	16'h0000	PREv18(A/E)
Register 20 (0x14, 0xA0000050) **(PREv18(A/E) Only)** Default: 24'h000000					
0x14	[15:0]	XOT_CNT_SAT_UPPER	W/R	16'h0000	PREv18(A/E)

Continued on next page

Continued from previous page

Reg Addr	Bit Field	Reg Name	Property	Size & Reset	Remark
Register 21 (0x15, 0xA0000054) **(PREv18(A/E) Only)** Default: 24'h000000					
0x15	[23]	XOT_CNT_EN	R	1'h0	PREv18(A/E)
	[22]	XOT_COUT_EN	R	1'h0	PREv18(A/E)
	[15:0]	XOT_CNT_VAL_LOWER	R	16'h0000	PREv18(A/E)
Register 22 (0x16, 0xA0000058) **(PREv18(A/E) Only)** Default: 24'h000000					
0x16	[15:0]	XOT_CNT_VAL_UPPER	R	16'h0000	PREv18(A/E)
Register 23 (0x17, 0xA000005C) **(PREv18(A/E) Only)** Default: 24'h000000					
0x17	[2:0]	CPS_ON	W/R	3'h0	PREv18(A/E)
Register 24 (0x18, 0xA0000060) **(PREv18(A/E) Only)** Default: 24'h000000					
0x18	[23]	SPI_FREEZE_OUTPUT	W/R	1'h0	PREv18(A/E)
	[21]	SPI_PAD_ENABLE_INPUT	W/R	1'h0	PREv18(A/E)
	[20]	SPI_PAD_ENABLE_OUTPUT	W/R	1'h0	PREv18(A/E)
	[16]	GPIO_FREEZE_OUTPUT	W/R	1'h0	PREv18(A/E)
	[15:12]	GPIO_WIRQ_POSEDGE_ENABLE	W/R	4'h0	PREv18(A/E)
	[11:8]	GPIO_WIRQ_NEGEDGE_ENABLE	W/R	4'h0	PREv18(A/E)
	[7:0]	GPIO_PAD_ENABLE	W/R	8'h00	PREv18(A/E)
Register 25 (0x19, 0xA0000064) **(PREv18(A/E) Only)** Default: 24'h602701					
0x19	[22]	XO_SLEEP	W/R	1'h1	PREv18(A/E)
	[21]	XO_ISOLATE	W/R	1'h1	PREv18(A/E)
	[20]	XO_EN_DIV	W/R	1'h0	PREv18(A/E)
	[19:17]	XO_S	W/R	3'h0	PREv18(A/E)
	[16]	XO_SEL_CP_DIV	W/R	1'h0	PREv18(A/E)
	[15]	XO_EN_OUT	W/R	1'h0	PREv18(A/E)
	[14:11]	XO_PULSE_SEL	W/R	4'h4	PREv18(A/E)
	[10:8]	XO_DELAY_EN	W/R	3'h7	PREv18(A/E)
	[7]	XO_DRV_START_UP	W/R	1'h0	PREv18(A/E)
	[6]	XO_DRV_CORE	W/R	1'h0	PREv18(A/E)
	[5]	XO_RP_LOW	W/R	1'h0	PREv18(A/E)
	[4]	XO_RP_MEDIA	W/R	1'h0	PREv18(A/E)
	[3]	XO_RP_MVT	W/R	1'h0	PREv18(A/E)
	[2]	XO_RP_SVT	W/R	1'h0	PREv18(A/E)
	[1]	XO_SCN_CLK_SEL	W/R	1'h0	PREv18(A/E)
	[0]	XO_SCN_ENB	W/R	1'h1	PREv18(A/E)
Register 26 (0x1A, 0xA0000068) **(PREv18(A/E) Only)** Default: 24'h000000					
0x1A	[11:0]	XO_CAP_TUNE	W/R	12'h000	PREv18(A/E)
Register 27 (0x1B, 0xA000006C) Default: 24'h000060					
0x1B	[6]	PUF_CHIP_ID_SLEEP	W/R	1'h1	
	[5]	PUF_CHIP_ID_ISOLATE	W/R	1'h1	
	[4]	ENABLE_SOFT_RESET	W/R	1'h0	
	[3:0]	WAKEUP_ON_PEND_REQ	W/R	4'h0	
Register 28 (0x1C, 0xA0000070) Default: 24'h000044					
0x1C	[13:9]	SRAM0_TUNE_ASO_DLY	W/R	5'h00	
	[8:5]	SRAM0_TUNE_WL_WIDTH	W/R	4'h2	
	[4:1]	SRAM0_TUNE_DECODER_DLY	W/R	4'h2	
	[0]	SRAM0_USE_INVERTER_SA	W/R	1'h0	
Register 29 (0x1D, 0xA0000074) **(PREv18 Only)** Default: 24'h000044					
0x1D	[13:9]	SRAM1_TUNE_ASO_DLY	W/R	5'h00	PREv18E
	[8:5]	SRAM1_TUNE_WL_WIDTH	W/R	4'h2	PREv18E
	[4:1]	SRAM1_TUNE_DECODER_DLY	W/R	4'h2	PREv18E
	[0]	SRAM1_USE_INVERTER_SA	W/R	1'h0	PREv18E
Register 30 (0x1E, 0xA0000078) Default: 24'h000000					
0x1E	[0]	MBUS_MSG_INTERRUPTED	R	1'h0	
Register 31 (0x1F, 0xA000007C) Default: 24'h16E360					
0x1F	[23:0]	MBUS_WD_THRESHOLD	W/R	24'h16E360	

Continued on next page

Continued from previous page					
Reg Addr	**Bit Field**	**Reg Name**	**Property**	**Size & Reset**	**Remark**
Register 41 (0x29, 0xA00000A4) Default: 24'h000000					
0x29	[23:16]	STR_WR_CH1_ALT_ADDR	W/R	8'h00	
	[15:0]	STR_WR_CH1_WR_BUF_LOWER	W/R	16'h0000	
Register 42 (0x2A, 0xA00000A8) Default: 24'h000000					
0x2A	[23:16]	STR_WR_CH1_ALT_REG_WR	W/R	8'h00	
	[15:0]	STR_WR_CH1_WR_BUF_UPPER	W/R	16'h0000	
Register 43 (0x2B, 0xA00000AC) Default: 24'h8007FF (Non-PREv18E), 24'h800FFF (PREv18E)					
0x2B	[23]	STR_WR_CH1_EN	W/R	1'h1	
	[22]	STR_WR_CH1_WRP	W/R	1'h0	
	[21]	STR_WR_CH1_DBLB	W/R	1'h0	
	[19:0]	STR_WR_CH1_BUF_LEN	W/R	20'h007FF	Non-PREv18E
				20'h00FFF	PREv18E
Register 44 (0x2C, 0xA00000B0) Default: 24'h000000					
0x2C	[19:0]	STR_WR_CH1_BUF_OFF	W/R	20'h00000	
Register 45 (0x2D, 0xA00000B4) Default: 24'h000000					
0x2D	[23:16]	STR_WR_CH0_ALT_ADDR	W/R	8'h00	
	[15:0]	STR_WR_CH0_WR_BUF_LOWER	W/R	16'h0000	
Register 46 (0x2E, 0xA00000B8) Default: 24'h000000					
0x2E	[23:16]	STR_WR_CH0_ALT_REG_WR	W/R	8'h00	
	[15:0]	STR_WR_CH0_WR_BUF_UPPER	W/R	16'h0000	
Register 47 (0x2F, 0xA00000BC) Default: 24'h8007FF (Non-PREv18E), 24'h800FFF (PREv18E)					
0x2F	[23]	STR_WR_CH0_EN	W/R	1'h1	
	[22]	STR_WR_CH0_WRP	W/R	1'h0	
	[21]	STR_WR_CH0_DBLB	W/R	1'h0	
	[19:0]	STR_WR_CH0_BUF_LEN	W/R	20'h007FF	Non-PREv18E
				20'h00FFF	PREv18E
Register 48 (0x30, 0xA00000C0) Default: 24'h000000					
0x30	[19:0]	STR_WR_CH0_BUF_OFF	W/R	20'h00000	
Register 51 (0x33, 0xA00000CC) Default: 24'h800000					
0x33	[23]	BLK_WR_EN	W/R	1'h1	
	[22]	BLK_WR_CACT	W/R	1'h0	
	[19:0]	BLK_WR_LENGTH_LIMIT	W/R	20'h00000	
Register 63 (0x3F, 0xA00000FC) Default: 24'h000000					
0x3F	[23:0]	CHIP_ID	W/R	24'h000000	
Register 64 (0x40, 0xA0000100) Default: 24'h000000					
0x40	[23]	ACT_RST	W/R	1'h0	

Table 4: PRCv18 MBus Register File Mapping

4.2 MBus Register Descriptions

4.2.1 Register 27 (0x1B, 0xA000006C)

WAKEUP_ON_PEND_REQ Reg 0x1B (0xA000006C), Bit Field: [3:0], Default: 4'h0, W/R

5 Cortex-M0 and Arbitration

5.1 Power and Clock

- CPU and Arbitration is operating at VDD_0P6_LC, which is a power-gated 0.6V, shared with Layer Controller.
- CPU and Arbitration use the same clock as Layer Controller (clk_lc).
- CPU and Arbitration are reset by the Layer Controller Reset signal (lc_reset_b).

5.2 Memory-Mapped Addresses

Table 5 shows memory mapped address for Cortex-M0 and Arbitration. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

Table 5: Memory Map for Cortex-M0 and Arbitration

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
0xA0000000	0x00	MBUS_R0	W/R	24'h000000	[23:0]
0xA0000004	0x01	MBUS_R1	W/R	24'h000000	[23:0]
0xA0000008	0x02	MBUS_R2	W/R	24'h000000	[23:0]
0xA000000C	0x03	MBUS_R3	W/R	24'h000000	[23:0]
0xA0000010	0x04	MBUS_R4	W/R	24'h000000	[23:0]
0xA0000014	0x05	MBUS_R5	W/R	24'h000000	[23:0]
0xA0000018	0x06	MBUS_R6	W/R	24'h000000	[23:0]
0xA000001C	0x07	MBUS_R7	W/R	24'h000000	[23:0]
0xA0000028	0x0A	FLAGS	W/R	24'h000000	[23:0]
0xA0000040	0x10	RUN_CPU	W/R	1'h0	[0]
0xA0001200	-	TIMERWD_GO	W/R	1'h1	[0]
0xA0001204	-	TIMERWD_CNT	W/R	32'h0016E360	[31:0]
0xA000A000	-	CONFIG_HALT_CPU	W/R	4'h9	[3:0]
0xA000A004	-	MBUS_TX_IRQ_AFTER_CLR_INT	W/R	1'h1	[0]
0xA000A008	-	WAKEUP_SOURCE	W/R	12'h000	[11:0]
0xAFFFF000	-	SYS_CTRL_HALT_CPU	W	N/A	[31:0]
0xAFFFF004	-	SYS_CTRL_RESUME_CPU	W	N/A	[31:0]
0xAFFFF008	-	SYS_CTRL_CLR_WUP_PEND_REQ	W	N/A	[31:0]
0xAFFFF00C	-	SYS_CTRL_CLR_WUP_SRC	W	N/A	[31:0]
0xE000E100	-	ISER	W/R	32'h00000000	[31:0]
0xE000E180	-	ICER	W/R	32'h00000000	[31:0]
0xE000E200	-	ISPR	W/R	32'h00000000	[31:0]
0xE000E280	-	ICPR	W/R	32'h00000000	[31:0]
0xE000E400	-	IPR0-7	W/R	32'h00000000	[31:0]
0xE000E404	-		W/R	32'h00000000	[31:0]
0xE000E408	-		W/R	32'h00000000	[31:0]
0xE000E40C	-		W/R	32'h00000000	[31:0]
0xE000E410	-		W/R	32'h00000000	[31:0]
0xE000E414	-		W/R	32'h00000000	[31:0]
0xE000E418	-		W/R	32'h00000000	[31:0]
0xE000E41C	-		W/R	32'h00000000	[31:0]

5.3 Retentive General-Purpose Registers

These are general purpose 24-bit retentive registers.

5.3.1 `0xA0000000` - `0xA000001C`

MBUS_R0 Reg `0x00` (`0xA0000000`), Bit Field: [23:0], Default: `24'h000000`, W/R

MBUS_R1 Reg `0x01` (`0xA0000004`), Bit Field: [23:0], Default: `24'h000000`, W/R

MBUS_R2 Reg `0x02` (`0xA0000008`), Bit Field: [23:0], Default: `24'h000000`, W/R

MBUS_R3 Reg `0x03` (`0xA000000C`), Bit Field: [23:0], Default: `24'h000000`, W/R

MBUS_R4 Reg `0x04` (`0xA0000010`), Bit Field: [23:0], Default: `24'h000000`, W/R

MBUS_R5 Reg `0x05` (`0xA0000014`), Bit Field: [23:0], Default: `24'h000000`, W/R

MBUS_R6 Reg `0x06` (`0xA0000018`), Bit Field: [23:0], Default: `24'h000000`, W/R

MBUS_R7 Reg `0x07` (`0xA000001C`), Bit Field: [23:0], Default: `24'h000000`, W/R

5.4 Flag Register

This is a 24-bit retentive register reserved to store flags. However, it could be also used as a general purpose register.

5.4.1 `0xA0000028`

FLAGS Reg `0x0A` (`0xA0000028`), Bit Field: [23:0], Default: `24'h000000`, W/R

5.5 CPU Control, Reset, and Halt

5.5.1 `0xA0000040`

RUN_CPU Reg `0x10` (`0xA0000040`), Bit Field: [0], Default: `1'h0`, W/R

This is the 'RESETn' signal for CPU. If set to 1, CPU starts running after the double-latching delay. If set to 0, CPU stops running.

Note that 'Reset Release (RUN_CPU: 0 $\rightarrow$ 1)' is double-latched within CPU hence it can be regarded as synchronous. However, 'Reset Assert (RUN_CPU: 1 $\rightarrow$ 0)' is asynchronous, and it is not double-latched.

It is NOT recommended that a software directly changes this value since it may lead to permanent system halt.

If the system goes into sleep while RUN_CPU=1, CPU will stop right before the system goes into sleep. RUN_CPU will still hold the value 1 in the sleep mode, since it is a retentive register. In this situation, if the system wakes up later, CPU will start running right after the system goes into active (after the double-latch delay).

CONFIG_HALT_CPU (`0xA000A000`), Bit Field: [3:0], Default: `4'h9`, W/R

PRC/PRE provides an option to halt CPU until a user-specified event is occurred. This is useful for forcing sequential operations. This option can be disabled or enabled/configured using `CONFIG_HALT_CPU`.

If set to other than `4'hF`, CPU becomes halted once MBux Tx is initiated (i.e, right after it raises INT_VECTOR), until the specified event is occurred.

See Table 6 for available events.

Table 6: `CONFIG_HALT_CPU`

`CONFIG_HALT_CPU`	**Description**
`4'h0`	Halt until `MBUS_R0` is written
`4'h1`	Halt until `MBUS_R1` is written
`4'h2`	Halt until `MBUS_R2` is written
`4'h3`	Halt until `MBUS_R3` is written
`4'h4`	Halt until `MBUS_R4` is written
`4'h5`	Halt until `MBUS_R5` is written
`4'h6`	Halt until `MBUS_R6` is written
`4'h7`	Halt until `MBUS_R7` is written
`4'h8`	Halt until SRAM is written through Layer Ctrl
`4'h9`	Halt until PRC/PRE finishes transmitting an MBus message (MBus Tx)
`4'hA`	Halt until PRC/PRE finishes receiving an MBus message (MBus Rx)
`4'hB`	Halt until PRC/PRE finishes forwarding an MBus message (MBus Fwd)
`4'hC`	Halt until PRC/PRE finished MBus Rx following MBus Tx (MBus TRx)
`4'hD`	Reserved
`4'hE`	Reserved
`4'hF`	Do not halt (Halt Disabled)

This is effective only for MBus Tx initiated by CPU through the Layer Ctrl interrupt. More precisely, CPU becomes halted when Arbitration unit sends an interrupt to the Layer Ctrl to initiate the MBus Tx. Thus, there may be some delay until the first MBus clock transition for the message occurs.

MBus Tx, that is not done by the Layer Ctrl interrupt, does not halt CPU even if `CONFIG_HALT_CPU` is set to do so. Examples that ignore `CONFIG_HALT_CPU` setting include, but are not limited to:

- Response to REG READ message from other layers
- Fake Wake-Up message triggered by the internal Wake-up Timer

MBUS_TX_IRQ_AFTER_CLR_INT (`0xA000A004`), Bit Field: [0], Default: `1'h1`, W/R

If 0, MBus Tx interrupt is generated when the Layer Ctrl asserts CLR_INT (Clear Interrupt): 0 $\rightarrow$ 1 which then goes into Arbitration Unit.

If 1, MBus Tx interrupt is generated when the Layer Ctrl resets CLR_INT: 1 $\rightarrow$ 0.

”Resetting CLR_INT” always happens after ”Asserting CLR_INT”, hence waiting until the ”Resetting CLR_INT” is more conservative and provide more safety (i.e., more deterministic behavior).

This setting affects the Cortex-M0 IRQ[5], and `CONFIG_HALT_CPU` (`4'h9`).

SYS_CTRL_HALT_CPU (`0xAFFFF000`), Write-Only

Writing `0xBAADF00D` in this register makes CPU stall until an event specified in `CONFIG_HALT_CPU` happens.

SYS_CTRL_RESUME_CPU (`0xAFFFF004`), Write-Only

Writing `0xCAFEF00D` in this register makes CPU resume its operation regardless of `CONFIG_HALT_CPU` setting.

5.6 CPU Watchdog Timer

CPU Watchdog Timer is a decremending counter. Upon its expiration (`TIMERWD_CNT`=0), it generates an interrupt (WD_IRQ), which is a digital outptu signal, active-high at VDD_1P2 level. It is user's responsibility to configure the system-level hard-reset using WD_IRQ.

5.6.1 `0xA00012XX`

TIMERWD_GO (`0xA0001200`), Bit Field: [0], Default: `1'h1`, W/R

Go Register. Starts the CPU Watchdog timer. It is reset to 1, thus the CPU Watchdog timer starts running by default once CPU starts running, unless specified otherwise by the user.

TIMERWD_CNT (`0xA0001204`), Bit Field: [31:0], Default: `32'h0016E360`, W/R

CPU Watchdog timer's counter register. It indicates what count CPU Watchdog timer is currently at. Once TIMERWD_CNT reaches 0, CPU Watchdog timer generates an interrupt (WD_IRQ), and TIMERWD_CNT stays 0 until System Reset or controlled by the user.

The default value (`32'h0016E360`) corresponds to approximately 20 seconds at 75kHz Core Clock frequency.

5.7 CPU Wakeup Behavior

5.7.1 `0xA000A008`

WAKEUP_SOURCE (`0xA000A008`), Bit Field: [11:0], Default: `12'h000`, W/R

Whenever the system wakes up, `WAKEUP_SOURCE` is automatically updated to indicate the source of the wake-up. See Table 7 for details.

If user writes data in this address, the data will overwrite any previous value stored in `WAKEUP_SOURCE`. Then, `WAKEUP_SOURCE` may lose its meaning and may not be valid anymore, until it gets updated by the system at the next wakeup event.

5.7.2 `0xAFFFF00X`

SYS_CTRL_CLR_WUP_PEND_REQ (`0xAFFFF008`), Write-Only

Writing 1 in specific bit in this register clears the corresponding "pending wakeup request". See Table 8.

Table 7: `WAKEUP_SOURCE`

Bit Field	Description	Related Section
[31:12]	Reserved	
[11]	GPIO[3] has triggered the wakeup (valid only when `WAKEUP_SOURCE`[3]=1)	Section 21
[10]	GPIO[2] has triggered the wakeup (valid only when `WAKEUP_SOURCE`[3]=1)	Section 21
[9]	GPIO[1] has triggered the wakeup (valid only when `WAKEUP_SOURCE`[3]=1)	Section 21
[8]	GPIO[0] has triggered the wakeup (valid only when `WAKEUP_SOURCE`[3]=1)	Section 21
[7:5]	Reserved	
[4]	MBus message has triggered the wakeup (e.g., Flash Auto Boot-up)	Section 8
[3]	One of GPIO[3:0] has triggered the wakeup	Section 21
[2]	XO Timer has triggered the wakeup	Section 14
[1]	Wakeup Timer has triggered the wakeup	Section 13
[0]	GOC/EP has triggered the wakeup	Section 9

Table 8: `SYS_CTRL_CLR_WUP_PEND_REQ`

Bit	Pending Wakeup Request to be Cleared	Related Section	Remark
[3]	GPIO Wakeup Request	Section 21	PREv18(A/E) Only
[2]	XOT_TIMER Wakeup Request	Section 14	PREv18(A/E) Only
[1]	WUP_TIMER Wakeup Request	Section 13	
[0]	GOC/EP Wakeup Request	Section 21	

SYS_CTRL_CLR_WUP_SRC (`0xAFFFF00C`), Write-Only

Writing any value in this register clears `WAKEUP_SOURCE` register.

5.8 Cortex-M0 Interrupts and NVIC (Nested Vectored Interrupt Control)

PRC/PRE support the following interrupt ports. User should implement appropriate interrupt service routines. All the interrupts shown in Table 9 can be configured using Cortex-M0's NVIC (Nested Vectored Interrupt Controller) configuration registers. MMIO address of those registers are shown in Table 5, which is repeated in Table 10. See ARM Cortex-M0 Devices Generic User Guide for more details.

5.8.1 `0xE000EXXX`

ISER (`0xE000E100`), Bit Field: [31:0], Default: `32'h00000000`, W/R

Interrupt Set-Enable Register. For detailed information, see *Cortex-M0 Devices Generic User Guide*.

ICER (`0xE000E180`), Bit Field: [31:0], Default: `32'h00000000`, W/R

Interrupt Clear-Enable Register. For detailed information, see *Cortex-M0 Devices Generic User Guide*.

ISPR (`0xE000E200`), Bit Field: [31:0], Default: `32'h00000000`, W/R

Interrupt Set-Pending Register. For detailed information, see *Cortex-M0 Devices Generic User Guide*.

ICPR (`0xE000E280`), Bit Field: [31:0], Default: `32'h00000000`, W/R

Interrupt Clear-Pending Register. For detailed information, see *Cortex-M0 Devices Generic User Guide*.

Table 9: Cortex-M0 IRQ

Interrupt Port	Description	Related Section
IRQ[0]	Interrupt right after system wakes up	Section 13
IRQ[1]	Interrupt when Soft Reset is done	Section 6
IRQ[2]	Interrupt when GOC/EP finishes	Section 9
IRQ[3]	Interrupt from 32-bit Timer	Section 11
IRQ[4]	Interrupt from 16-bit Timer	Section 10
IRQ[5]	Interrupt when PRC/PRE finishes MBus Tx	Section 8
IRQ[6]	Interrupt when PRC/PRE finishes MBus Rx	Section 8
IRQ[7]	Interrupt when PRC/PRE finishes MBus Forwarding	Section 8
IRQ[8]	Interrupt when `MBUS_R0` is written	Section 5
IRQ[9]	Interrupt when `MBUS_R1` is written	Section 5
IRQ[10]	Interrupt when `MBUS_R2` is written	Section 5
IRQ[11]	Interrupt when `MBUS_R3` is written	Section 5
IRQ[12]	Interrupt when `MBUS_R4` is written	Section 5
IRQ[13]	Interrupt when `MBUS_R5` is written	Section 5
IRQ[14]	Interrupt when `MBUS_R6` is written	Section 5
IRQ[15]	Interrupt when `MBUS_R7` is written	Section 5
IRQ[16]	Interrupt when SRAM is written through Layer Ctrl	Section 7
IRQ[17]	[PREv18(A/E) Only] Interrupt from GPIO	Section 21
IRQ[18]	[PREv18(A/E) Only] Interrupt from SPI	Section 20
IRQ[19]	[PREv18(A/E) Only] Interrupt from XO Timer	Section 14

Table 10: Cortex-M0 NVIC Registers

MMIO ADDR	Register Name	W/R	Default	Bit Field
`0xE000E100`	`ISER`	W/R	`32'h00000000`	[31:0]
`0xE000E180`	`ICER`	W/R	`32'h00000000`	[31:0]
`0xE000E200`	`ISPR`	W/R	`32'h00000000`	[31:0]
`0xE000E280`	`ICPR`	W/R	`32'h00000000`	[31:0]
`0xE000E400`	`IPR0-7`	W/R	`32'h00000000`	[31:0]
`0xE000E404`		W/R	`32'h00000000`	[31:0]
`0xE000E408`		W/R	`32'h00000000`	[31:0]
`0xE000E40C`		W/R	`32'h00000000`	[31:0]
`0xE000E410`		W/R	`32'h00000000`	[31:0]
`0xE000E414`		W/R	`32'h00000000`	[31:0]
`0xE000E418`		W/R	`32'h00000000`	[31:0]
`0xE000E41C`		W/R	`32'h00000000`	[31:0]

IPR0-7 (`0xE000E400` ∼ `0xE000E41C`), Bit Field: [31:0], Default: `32'h00000000`, W/R

Interrupt Priority Registers. For detailed information, see *Cortex-M0 Devices Generic User Guide*.

6 Soft Reset

This is a CPU reset mechanism triggered by an MBus Memory Write message.

6.1 Memory-Mapped Addresses

Table 11 shows memory mapped address for Soft Reset. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

PUF_* signals and `WAKEUP_ON_PEND_REQ` are not related to Soft Reset. See Section 12 and Section 4 for their detailed information.

Table 11: Memory Map for Soft Reset

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
`0xA0000034`	`0x0D`	`PUF_CHIP_ID`	R	`24'h000000`	[23:0]
`0xA000006C`	`0x1B`	`PUF_CHIP_ID_SLEEP`	W/R	`1'h1`	[6]
		`PUF_CHIP_ID_ISOLATE`	W/R	`1'h1`	[5]
		`ENABLE_SOFT_RESET`	W/R	`1'h0`	[4]
		`WAKEUP_ON_PEND_REQ`	W/R	`4'h0`	[3:0]
`0xA00000FC`	`0x3F`	`CHIP_ID`	W/R	`24'h000000`	[23:0]

6.2 Retentive Soft Reset Configuration Registers

6.2.1 `0xA000006C`

ENABLE_SOFT_RESET Reg `0x1B` (`0xA000006C`), Bit Field: [4], Default: `1'h0`, W/R

If 1, Soft Reset is enabled. See Section 6.3.

6.3 Operation

CPU will start from the beginning of the program (SRAM Address#0) if all the following conditions are met.

- `ENABLE_SOFT_RESET` is set to 1
- There is an incoming MBus Memory Write message targeting PRC/PRE, where the memory start address is set to `0x6C840000`.
- The length of the data in such MBus Memory Write message must be 64 bits or longer.

If all of these conditions are met, the incoming data will be written from SRAM Address#0. CPU will re-start from the beginning (SRAM Address#0) once the MBus message ends.

6.4 Interrupt

Soft Reset generates a Cortex-M0 IRQ once it is done. The interrupt can be further masked by Cortex-M0 NVIC registers.

Table 12: Soft Reset-related Cortex-M0 IRQ

Interrupt Port	Description
IRQ[1]	Interrupt when Soft Reset is done

7 Retentive SRAM

PRCv18/PRCv18G/PREv18 have a copy of 8kB retentive SRAM, and the SRAM macro name is RSRAM7Tv775s8kBx32m2Mar2018_V.

PREv18E has two copies of 8kB retentive SRAM (total 16kB). The macro names are RSRAM7Tv775s8kBx32m2Mar2018_V (first 8kB section) and RSRAM7Tv775s8kBx32m2Mar2018_H (second 8kB section).

7.1 SRAM Specification

- Macro Name: RSRAM7Tv775s8kBx32m2Mar2018_V, RSRAM7Tv775s8kBx32m2Mar2018_H
- Capacity: 8kB, 32-bit I/O, 2:1 Column-Muxing
- Array Configuration: (128 WLs x 64 BLs)/Array. 2 Arrays/Bank. Total 4 Banks.
- Bitcell: 7T Bitcell (HVT 6T part+ 1 SVT Read Device) built with Custom Pushed-Rules. Bitcell measures 4.12um x 1.88um (7.75um^2)
- 8kB Macro Layout Size: 1299.76um x 638.98um (0.831mm^2)
- Powers
  - VDD_1P2: Used for NWELL biasing and WL voltage during sleep. Default = 1.2V.
  - VDD_0P6: Used for cell VDD (data storage). Default = 0.6V.
  - VDD_1P2PG: Used in WL and BL drivers. Default = 1.2V Power Gated.
  - VDD_0P6PG: Used for main logic and control. Default = 0.6V Power Gated.
- Auto-Shut-Off Sensing: By default, entire WLs become shuts-off once there is an enough voltage development on Read BL. This is to eliminate the excessive READ power caused by the 7T topology.
- Retention Power: 3.35fW/bit @ 0.95V/0.60V (measured)
- Energy: 0.39pJ/bit for READ, 0.41pJ/bit for WRITE @ 0.95V/0.60V (measured)
- Frequency: Up to 4.6MHz (measured)
- V_{MIN} = 320mV @ 50kHz (measured)

7.2 Memory-Mapped Addresses

Table 13 shows memory mapped address for the retentive SRAMs.

7.3 Input Signals

By default, all input signals must be at 0.6V or higher, except ISOLATEn which must be $>$2.5V.

- CLK: Clock signal.
- RESETn: Reset (Active-Low). Usually this is driven by a POR Reset Detector.
- MEM_EN: Memory Enable (Active-High). SRAM performs READ or WRITE only when MEM_EN is high.

Table 13: Memory Map for Retentive SRAM

MMIO ADDR	MBus Reg ID	Name	W/R	Default	Bit Field	Remark
`0x00000000` ∼ `0x00001FFC`	-	8kB SRAM (SRAM0)	W/R	-	-	
`0x00002000` ∼ `0x00003FFC`	-	8kB SRAM (SRAM1)	W/R	-	-	PREv18E Only
`0xA0000070`	`0x1C`	`SRAM0_TUNE_ASO_DLY`	W/R	`5'h00`	[13:9]	
		`SRAM0_TUNE_WL_WIDTH`	W/R	`4'h2`	[8:5]	
		`SRAM0_TUNE_DECODER_DLY`	W/R	`4'h2`	[4:1]	
		`SRAM0_USE_INVERTER_SA`	W/R	`1'h0`	[0]	
`0xA0000074`	`0x1D`	`SRAM1_TUNE_ASO_DLY`	W/R	`5'h00`	[13:9]	PREv18E Only
		`SRAM1_TUNE_WL_WIDTH`	W/R	`4'h2`	[8:5]	PREv18E Only
		`SRAM1_TUNE_DECODER_DLY`	W/R	`4'h2`	[4:1]	PREv18E Only
		`SRAM1_USE_INVERTER_SA`	W/R	`1'h0`	[0]	PREv18E Only

- READ0_WRITE1: READ/WRITE select. SRAM performs READ if MEM_EN=1 and READ0_WRITE1=0. SRAM performs WRITE if MEM_EN=1 and READ0_WRITE1=1.
- ISOLATEn: Isolate (Active-Low). This signal resets all WLs and cuts off all WBLs for an ultra-low leakage sleep mode. It must be >2.5V during active mode for correct operation. (>3.5V is recommended).
- ADDR[10:0]: Address
- DATAIN[31:0]: Data for WRITE

7.4 Output Signals

All output signal is at 0.6V by default.

- DATAOUT[31:0]: READ Data.

7.5 Tuning Bits

Following is a list of the tuning bits. These signals must be at 1.2V or higher. In order to set these tuning bits, PRC/PREv18 Family uses MBus Register File. See *Section 7.6. MBus Register File*.

- SA_SEL: If 0, SRAM uses Coarse & Fine Sense Amplifiers. This is the recommended setting for low-power operation. If 1, SRAM uses a simple inverter for sensing. This is for debugging-purpose only and may consume extra power. In PRC/PREv18 Family, this is mapped to `SRAM0_USE_INVERTER_SA` or `SRAM0_USE_INVERTER_SA`.
- TUNE_DLY1[3:0]: This controls a delay circuit that measures the decoder delay. Recommended default value is `4'h2`. In PRC/PREv18 Family, this is mapped to `SRAM0_TUNE_DECODER_DLY` or `SRAM0_TUNE_DECODER_DLY`.
- TUNE_DLY2[3:0]: This controls a delay circuit that determines the WL width. Recommended default value is `4'h2`. In PRC/PREv18 Family, this is mapped to `SRAM0_TUNE_WL_WIDTH` or `SRAM0_TUNE_WL_WIDTH`.
- TUNE_RDRSB_EN: If 1, it adds extra delay to Auto-Shut-Off. The extra delay is determined by TUNE_RDRSB tuning bits. If 0, the Auto-Shut-Off does not use the extra delay. TUNE_RDRSB_EN is provided for debugging-purpose, and the recommended default value is 0. In PRC/PREv18 Family, this is mapped to `SRAM0_TUNE_ASO_DLY`[4] or `SRAM0_TUNE_ASO_DLY`[4].

- TUNE_RDRSB[3:0]: It controls the extra delay added to Auto-Shut-Off, with `4'h0` being the shortest and `4'hF` being the longest. This becomes effective only when TUNE_RDRSB_EN is 1. In PRC/PREv18 Family, this is mapped to `SRAM0_TUNE_ASO_DLY`[3:0] or `SRAM0_TUNE_ASO_DLY`[3:0].
- VREF_SEL: If 0, the Fine Sense Amplifier uses VDD_1P2PG for reference. If 1, the Fine Sense Amplifier uses VDD_0P6PG for reference. Recommended default value is 0. It may need to be set 1 if the process corner is at extremely Fast NMOS. In PRC/PREv18 Family, this is hard-wired to 0.

7.6 MBus Register File

7.6.1 First 8kB SRAM (SRAM0)

SRAM0_TUNE_ASO_DLY Reg `0x1C` (`0xA0000070`), Bit Field: [13:9], Default: `5'h00`, W/R

Auto-Shut-Off Delay tuning for the first SRAM (the first 8kB).

If the MSB is 0 (i.e., SRAM0_TUNE_ASO_DLY[4]=0), this is not effective and thus ignored.

Only if the MSB is 1 (i.e., SRAM0_TUNE_ASO_DLY[4]=1), SRAM adds an extra delay to its Auto-Shut-Off delay element. The amount of delay is determined by SRAM0_TUNE_ASO_DLY[3:0], where `4'h0` being the shortest, and `4'hF` being the longest.

By default, it is recommended to set SRAM0_TUNE_ASO_DLY[4]=0, since the SRAM was designed to operate correctly without any extra delay added to Auto-Shut-Off, and the less delay means lower power. However, if SRAM has READ errors in normal operation, it would be helpful to add some more delay to Auto-Shut-Off.

SRAM0_TUNE_WL_WIDTH Reg `0x1C` (`0xA0000070`), Bit Field: [8:5], Default: `4'h2`, W/R

Wordline Width tuning for the first SRAM (the first 8kB).

This sets the Wordline (WL) pulse width of SRAM, with `4'h0` being the shortest, and `4'hF` being the longest. The Sense Amplifiers are fired at the end of this WL pulse.

A larger value would improve the robustness at the cost of performance and READ power consumption.

SRAM0_TUNE_DECODER_DLY Reg `0x1C` (`0xA0000070`), Bit Field: [4:1], Default: `4'h2`, W/R

Decoder canary circuit delay tuning for the first SRAM (the first 8kB).

This sets the delay between the clock's positive edge and the WL activation. Theoretically, this must be longer than the decoder's worst-case delay.

A larger value would improve the robustness at the cost of performance.

SRAM0_USE_INVERTER_SA Reg `0x1C` (`0xA0000070`), Bit Field: [0], Default: `1'h0`, W/R

Sense Amplifier selection for the first SRAM (the first 8kB).

If 1, SRAM uses a simple inverter for sensing READ Bitline (BL) voltages.

If 0, SRAM uses Coars/Fine Sense Amplifier for sensing READ BL voltages.

By default, SRAM uses the Coarse/Fine Sense Amplifier for sensing, since it is more efficient in terms of power and speed. The use of the simple inverter may result in better robustness but at the cost of (probably) lots of power overhead.

SRAM0_USE_INVERTER_SA is provided only for debugging-purpose, so it rarely has to be set 1 to use the simple inverter sensing.

7.6.2 Second 8kB SRAM (SRAM1) - PREv18E Only

SRAM1_TUNE_ASO_DLY Reg `0x1D` (`0xA0000074`), Bit Field: [13:9], Default: `5'h00`, W/R

Auto-Shut-Off Delay tuning for the second SRAM (the second 8kB).

If the MSB is 0 (i.e., SRAM1_TUNE_ASO_DLY[4]=0), this is not effective and thus ignored.

Only if the MSB is 1 (i.e., SRAM1_TUNE_ASO_DLY[4]=1), SRAM adds an extra delay to its Auto-Shut-Off delay element. The amount of delay is determined by SRAM1_TUNE_ASO_DLY[3:0], where `4'h0` being the shortest, and `4'hF` being the longest.

By default, it is recommended to set SRAM1_TUNE_ASO_DLY[4]=0, since the SRAM was designed to operate correctly without any extra delay added to Auto-Shut-Off, and the less delay means lower power. However, if SRAM has READ errors in normal operation, it would be helpful to add some more delay to Auto-Shut-Off.

SRAM1_TUNE_WL_WIDTH Reg `0x1D` (`0xA0000074`), Bit Field: [8:5], Default: `4'h2`, W/R

Wordline Width tuning for the second SRAM (the second 8kB).

This sets the Wordline (WL) pulse width of SRAM, with `4'h0` being the shortest, and `4'hF` being the longest. The Sense Amplifiers are fired at the end of this WL pulse.

A larger value would improve the robustness at the cost of performance and READ power consumption.

SRAM1_TUNE_DECODER_DLY Reg `0x1D` (`0xA0000074`), Bit Field: [4:1], Default: `4'h2`, W/R

Decoder canary circuit delay tuning for the second SRAM (the second 8kB).

This sets the delay between the clock's positive edge and the WL activation. Theoretically, this must be longer than the decoder's worst-case delay.

A larger value would improve the robustness at the cost of performance.

SRAM1_USE_INVERTER_SA Reg `0x1D` (`0xA0000074`), Bit Field: [0], Default: `1'h0`, W/R

Sense Amplifier selection for the second SRAM (the second 8kB).

If 1, SRAM uses a simple inverter for sensing READ Bitline (BL) voltages.

If 0, SRAM uses Coars/Fine Sense Amplifier for sensing READ BL voltages.

By default, SRAM uses the Coarse/Fine Sense Amplifier for sensing, since it is more efficient in terms of power and speed. The use of the simple inverter may result in better robustness but at the cost of (probably) lots of power overhead.

SRAM1_USE_INVERTER_SA is provided only for debugging-purpose, so it rarely has to be set 1 to use the simple inverter sensing.

7.7 Cortex-M0 Interrupts

Table 14 shows Cortex-M0 interrupts that can be triggered by the retentive SRAMs. The interrupt can be further masked by Cortex-M0 NVIC registers.

7.7.1 List of Interrupts

IRQ[1] IRQ[1] is generated if an incoming MBus memory streaming message triggers the Soft Reset. For details about the Soft Reset, see Section 6. The interrupt is generated once the Soft Reset is done.

Table 14: Cortex-M0 Interrupts from Retentive SRAM

Interrupt Port	**Description**	**Related Section**
IRQ[1]	Interrupt when Soft Reset is done	Section 6
IRQ[16]	Interrupt when SRAM is written through Layer Controller	-

IRQ[16] IRQ[16] is generated when the SRAM is written through Layer Controller. This implies that there is an incoming MBus memory write message (either bulk or streaming) that writes into the SRAM. The interrupt is generated at the end of the MBus message.

8 MBus

8.1 Power and Clock

8.2 Memory-Mapped Addresses

Table 15 shows memory mapped address for MBus. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

Table 15: Memory Map for MBus

MMIO ADDR	MBus Reg ID	Name	W/R	Default	Bit Field	Remark
0xA0000024	0x09	MBUS_IGNORE_RX_FAIL	W/R	1'h1	[23]	
		MBUS_NUM_BITS_THRESHOLD	W/R	20'h80008	[19:0]	
0xA0000078	0x1E	MBUS_MSG_INTERRUPTED	R	1'h0	[0]	
0xA000007C	0x1F	MBUS_WD_THRESHOLD	W/R	24'h16E360	[23:0]	
0xA00000A4	0x29	STR_WR_CH1_ALT_ADDR	W/R	8'h00	[23:16]	
		STR_WR_CH1_WR_BUF_LOWER	W/R	16'h0000	[15:0]	
0xA00000A8	0x2A	STR_WR_CH1_ALT_REG_WR	W/R	8'h00	[23:16]	
		STR_WR_CH1_WR_BUF_UPPER	W/R	16'h0000	[15:0]	
0xA00000AC	0x2B	STR_WR_CH1_EN	W/R	1'h1	[23]	
		STR_WR_CH1_WRP	W/R	1'h0	[22]	
		STR_WR_CH1_DBLB	W/R	1'h0	[21]	
		STR_WR_CH1_BUF_LEN	W/R	20'h007FF	[19:0]	Non-PREv18E
			W/R	20'h00FFF	[19:0]	PREv18E
0xA00000B0	0x2C	STR_WR_CH1_BUF_OFF	W/R	20'h00000	[19:0]	
0xA00000B4	0x2D	STR_WR_CH0_ALT_ADDR	W/R	8'h00	[23:16]	
		STR_WR_CH0_WR_BUF_LOWER	W/R	16'h0000	[15:0]	
0xA00000B8	0x2E	STR_WR_CH0_ALT_REG_WR	W/R	8'h00	[23:16]	
		STR_WR_CH0_WR_BUF_UPPER	W/R	16'h0000	[15:0]	
0xA00000BC	0x2F	STR_WR_CH0_EN	W/R	1'h1	[23]	
		STR_WR_CH0_WRP	W/R	1'h0	[22]	
		STR_WR_CH0_DBLB	W/R	1'h0	[21]	
		STR_WR_CH0_BUF_LEN	W/R	20'h007FF	[19:0]	Non-PREv18E
			W/R	20'h00FFF	[19:0]	PREv18E
0xA00000C0	0x30	STR_WR_CH0_BUF_OFF	W/R	20'h00000	[19:0]	
0xA00000CC	0x33	BLK_WR_EN	W/R	1'h1	[23]	
		BLK_WR_CACT	W/R	1'h0	[22]	
		BLK_WR_LENGTH_LIMIT	W/R	20'h00000	[19:0]	
0xA0000100	0x40	ACT_RST	W/R	1'h0	[23]	
0xA0001500	-	MBUS_WD_RESET_REQ	W	1'h0	[0]	
0xA0001504	-	MBUS_WD_RESET_OVERRIDE	W	1'h0	[0]	
0xA0002000	-	MBUS_CMD0	W	32'h0	[31:0]	
0xA0002004	-	MBUS_CMD1	W	32'h0	[31:0]	
0xA0002008	-	MBUS_CMD2	W	32'h0	[31:0]	
0xA000200C	-	MBUS_FUID_LEN	W	6'h00	[5:0]	
0xA0003XX0	-	QUICK_MBUS	W	32'h0	[31:0]	

8.3 Retentive MBus Registers

8.3.1 MBus Configuration

MBUS_IGNORE_RX_FAIL Reg 0x09 (0xA0000024), Bit Field: [23], Default: 1'h1, W/R

If 1, Layer Ctrl ignores RX_FAIL and proceeds to the next state. This slightly reduces the performance overhead as it does not trigger the error handling.

If 0, Layer Ctrl triggers the error handling if RX_FAIL is 1.

In this version of Layer Ctrl, the error handler is just another empty state, so the performance increase with `MBUS_IGNORE_RX_FAIL`=1 is only marginal.

If there is a long MBus message, such as Memory Streaming, with `MBUS_IGNORE_RX_FAIL`=1, and if there is RX_FAIL, then the system may hang. If this happens, the system should be eventually reset and resumed by the MBus Watchdog Timer. However, in order to prevent this situation, it is recommended to set `MBUS_IGNORE_RX_FAIL`=0 whenever the system expects longer-than-32-bit MBus messages.

MBUS_NUM_BITS_THRESHOLD Reg `0x09` (`0xA0000024`), Bit Field: [19:0], Default: `20'h80008`, W/R

MBus message threshold.

If the number of bits in ADDR and DATA reaches `MBUS_NUM_BITS_THRESHOLD`, PRC/PRE generates TX_FAIL and terminates the message.

By default, it is set to `20'h80008`, which corresponds to 8-bit ADDR + 64kB DATA.

MBUS_WD_THRESHOLD Reg `0x1F` (`0xA000007C`), Bit Field: [23:0], Default: `24'h16E360`, W/R

MBus watchdog threshold. If there is no MBus message until the internal MBus watchdog timer reaches the value specified in `MBUS_WD_THRESHOLD`, then PRC/PRE sends out the MBus sleep message to put the entire system to sleep.

The MBus watchdog timer is running off the internal MBus clock (CLK_MBC). The MBus watchdog timer is reset to 0 when one of the following events happens:

- There is an activity at MBus CIN/COUT.
- CPU accesses MMIO to request the MBus watchdog timer reset.

By default, it is set to `24'h16E360`, which corresponds to 10 seconds with 150kHz MBus clock (CLK_MBC) speed.

MBUS_MSG_INTERRUPTED Reg `0x1E` (`0xA0000078`), Bit Field: [0], Default: `1'h0`, R

It becomes set to 1 when GOC/EP becomes activated while there is an on-going MBus message, only if:

- The MBus message is NOT the first message (usually the first message is a wake-up message)
- The MBus message is either Tx or Rx for PRC/PRE
  - If the MBus message is a forwarding message, the flag is not set. However, since the 'forward' message cannot be identified until it receives the whole MBus ADDR section, it is possible that the flag becomes set even if the MBus message is a forwarding message. This could happen if GOC/EP becomes activated while PRC/PRE is receiving the ADDR section of the forwarding message.
- The MBus message finishes, but the Guard Band counter is not zero
  - This is to provide some margin to cover the last RX_REQ/RX_ACK in a long MBus message.
- There is an un-cleared TX_REQ.

This flag becomes automatically reset to 0 after being read.

8.3.2 Memory Streaming Configuration - Channel 1

STR_WR_CH1_ALT_ADDR Reg `0x29` (`0xA00000A4`), Bit Field: [23:16], Default: `8'h00`, W/R

Alert Address (8-bit) for Memory Streaming (Channel 1). Alerts are sent whenever the end of the buffer is reached. If `STR_WR_CH1_DBLB` is active, an alert is also sent when the halfway point of the buffer is reached.

See *MBus Specification* document for more details.

STR_WR_CH1_WR_BUF_LOWER Reg `0x29` (`0xA00000A4`), Bit Field: [15:0], Default: `16'h0000`, W/R

Pointer to the beginning of a buffer in memory for Memory Streaming (Channel 1). The pointer itself is 32-bit, and `STR_WR_CH1_WR_BUF_LOWER` represents the lower 16-bit. The 2 LSBs are always 0 (word-aligned).

See *MBus Specification* document for more details.

STR_WR_CH1_ALT_REG_WR Reg `0x2A` (`0xA00000A8`), Bit Field: [23:16], Default: `8'h00`, W/R

When an alert message is sent for Memory Streaming (Channel 1), `STR_WR_CH1_ALT_REG_WR` populates [31:24] in the MBus Data.

See *MBus Specification* document for more details.

STR_WR_CH1_WR_BUF_UPPER Reg `0x2A` (`0xA00000A8`), Bit Field: [15:0], Default: `16'h0000`, W/R

Pointer to the beginning of a buffer in memory for Memory Streaming (Channel 1). The pointer itself is 32-bit, and `STR_WR_CH1_WR_BUF_UPPER` represents the upper 16-bit.

See *MBus Specification* document for more details.

STR_WR_CH1_EN Reg `0x2B` (`0xA00000AC`), Bit Field: [23], Default: `1'h1`, W/R

If 1, Memory Streaming (Channel 1) is enabled.

If 0, Memory Streaming (Channel 1) is disabled.

See *MBus Specification* document for more details.

STR_WR_CH1_WRP Reg `0x2B` (`0xA00000AC`), Bit Field: [22], Default: `1'h0`, W/R

If 1, the Write Address Counter for Memory Streaming (Channel 1) is reset to `STR_WR_CH1_BUF_OFF` when the end of the buffer is reached.

If 0, the Write Address Counter for Memory Streaming (Channel 1) is un-changed when the end of the buffer is reached. Also, `STR_WR_CH1_EN` is changed to 0 at this point.

See *MBus Specification* document for more details.

STR_WR_CH1_DBLB Reg `0x2B` (`0xA00000AC`), Bit Field: [21], Default: `1'h0`, W/R

Double-buffering mode for Memory Streaming (Channel 1).

If 1, it generates an alert message halfway through the buffer in addition to at the end of the buffer.

If 0, it generates only when the end of the buffer is reached.

See *MBus Specification* document for more details.

STR_WR_CH1_BUF_LEN Reg `0x2B` (`0xA00000AC`), Bit Field: [19:0], Default: `20'h007FF` (Non-PREv18E), `20'h00FFF` (PREv18E), W/R

It defines the length of the buffer for Memory Streaming (Channel 1). It is 'Num of Words in Buffer - 1'

STR_WR_CH1_BUF_OFF Reg `0x2C` (`0xA00000B0`), Bit Field: [19:0], Default: `20'h00000`, W/R

It defines the buffer offset for Memory Streaming (Channel 1).

8.3.3 Memory Streaming Configuration - Channel 0

STR_WR_CH0_ALT_ADDR Reg `0x2D` (`0xA00000B4`), Bit Field: [23:16], Default: `8'h00`, W/R

Alert Address (8-bit) for Memory Streaming (Channel 0). Alerts are sent whenever the end of the buffer is reached. If `STR_WR_CH0_DBLB` is active, an alert is also sent when the halfway point of the buffer is reached.

See *MBus Specification* document for more details.

STR_WR_CH0_WR_BUF_LOWER Reg `0x2D` (`0xA00000B4`), Bit Field: [15:0], Default: `16'h0000`, W/R

Pointer to the beginning of a buffer in memory for Memory Streaming (Channel 0). The pointer itself is 32-bit, and `STR_WR_CH0_WR_BUF_LOWER` represents the lower 16-bit. The 2 LSBs are always 0 (word-aligned).

See *MBus Specification* document for more details.

STR_WR_CH0_ALT_REG_WR Reg `0x2E` (`0xA00000B8`), Bit Field: [23:16], Default: `8'h00`, W/R

When an alert message is sent for Memory Streaming (Channel 0), `STR_WR_CH0_ALT_REG_WR` populates [31:24] in the MBus Data.

See *MBus Specification* document for more details.

STR_WR_CH0_WR_BUF_UPPER Reg `0x2E` (`0xA00000B8`), Bit Field: [15:0], Default: `16'h0000`, W/R

Pointer to the beginning of a buffer in memory for Memory Streaming (Channel 0). The pointer itself is 32-bit, and `STR_WR_CH0_WR_BUF_UPPER` represents the upper 16-bit.

See *MBus Specification* document for more details.

STR_WR_CH0_EN Reg `0x2F` (`0xA00000BC`), Bit Field: [23], Default: `1'h1`, W/R

If 1, Memory Streaming (Channel 0) is enabled.

If 0, Memory Streaming (Channel 0) is disabled.

See *MBus Specification* document for more details.

STR_WR_CH0_WRP Reg `0x2F` (`0xA00000BC`), Bit Field: [22], Default: `1'h0`, W/R

If 1, the Write Address Counter for Memory Streaming (Channel 0) is reset to `STR_WR_CH0_BUF_OFF` when the end of the buffer is reached.

If 0, the Write Address Counter for Memory Streaming (Channel 0) is un-changed when the end of the buffer is reached. Also, `STR_WR_CH0_EN` is changed to 0 at this point.

See *MBus Specification* document for more details.

STR_WR_CH0_DBLB Reg `0x2F` (`0xA00000BC`), Bit Field: [21], Default: `1'h0`, W/R

Double-buffering mode for Memory Streaming (Channel 0).

If 1, it generates an alert message halfway through the buffer in addition to at the end of the buffer.

If 0, it generates only when the end of the buffer is reached.

See *MBus Specification* document for more details.

STR_WR_CH0_BUF_LEN Reg `0x2F` (`0xA00000BC`), Bit Field: [19:0], Default: `20'h007FF` (Non-PREv18E), `20'h00FFF` (PREv18E), W/R

It defines the length of the buffer for Memory Streaming (Channel 0). It is 'Num of Words in Buffer - 1'

STR_WR_CH0_BUF_OFF Reg `0x30` (`0xA00000C0`), Bit Field: [19:0], Default: `20'h00000`, W/R

It defines the buffer offset for Memory Streaming (Channel 0).

8.3.4 Memory Bulk Write Configuration

BLK_WR_EN Reg `0x33` (`0xA00000CC`), Bit Field: [23], Default: `1'h1`, W/R

If 1, Memory Bulk Write is enabled.

If 0, Memory Bulk Write is disabled.

BLK_WR_CACT Reg `0x33` (`0xA00000CC`), Bit Field: [22], Default: `1'h0`, W/R

If 1, `BLK_WR_LENGTH_LIMIT` acts as a limit for the maximum permitted bulk message length. A bulk message is allowed to write until this message limit is reached. If more data comes, the message becomes NAK'd.

If 0, this feature is disabled.

BLK_WR_LENGTH_LIMIT Reg `0x33` (`0xA00000CC`), Bit Field: [19:0], Default: `20'h00000`, W/R

If `BLK_WR_CACT` is 1, `BLK_WR_LENGTH_LIMIT` acts as a limit for the maximum permitted bulk message length.

If `BLK_WR_CACT` is 0, `BLK_WR_LENGTH_LIMIT` has no meaning.

8.3.5 Action Register

ACT_RST Reg `0x40` (`0xA0000100`), Bit Field: [23], Default: `1'h0`, W/R

Action Register Reset. This is NOT implemented in the current version of Layer Ctrl.

8.4 Non-Retentive MBus Registers

8.4.1 `0xA00015XX`

MBUS_WD_RESET_REQ (`0xA0001500`), Default: `1'h0`, W

When CPU writes any value in `MBUS_WD_RESET_REQ`, it sends a one-time reset request to the MBus watchdog timer. MBus watchdog timer will reset its value to 0 and re-start from 0.

MBUS_WD_RESET_OVERRIDE (`0xA0001504`), Default: `1'h0`, W

If CPU writes 1 in `MBUS_WD_RESET_OVERRIDE`, MBus watchdog timer will be reset to 0. It will remain 0 until CPU writes 0 in `MBUS_WD_RESET_OVERRIDE`.

8.4.2 `0xA00020XX`

MBUS_CMD0 (`0xA0002000`), Default: `32'h00000000`, W

Sets INT_CMD[95:64]. See Section 8.5.2.

MBUS_CMD1 (`0xA0002004`), Default: `32'h00000000`, W

Sets INT_CMD[63:32]. See Section 8.5.2.

MBUS_CMD2 (`0xA0002008`), Default: `32'h00000000`, W

Sets INT_CMD[31:0]. See Section 8.5.2.

MBUS_FUID_LEN (`0xA000200C`), Default: `6'h00`, W

Sets INT_CMD_LEN and INT_FU_ID and then asserts INT_VECTOR. See Section 8.5.2 for more details.

- `MBUS_FUID_LEN`[5:4]: Sets INT_CMD_LEN.
- `MBUS_FUID_LEN`[3:0]: Sets INT_FU_ID.

8.4.3 `0xA0003XX0`

QUICK_MBUS (`0xA0003XX0`), Default: `32'h00000000`, W

Sends a 32-bit MBus message. See Section 8.5.3.

8.5 Operation

8.5.1 MBus Watchdog Timer

MBus Watchdog Timer starts running by default once the internal MBus clock starts running. If there is no MBus message until the internal MBus watchdog timer reaches the value specified in `MBUS_WD_THRESHOLD`, then PRC/PRE sends out the MBus sleep message to put the entire system to sleep.

Here, the resulting sleep message is the ”Selective Sleep By Short Prefix” as defined in MBus Specification. Its Address is `0x01`, and the Data is `0x2FFFF000`.

The MBus watchdog timer is running off the internal MBus clock (CLK_MBC). The MBus watchdog timer is reset to 0 when one of the following events happens:

- There is an activity at MBus CIN/COUT.
- CPU accesses MMIO to request the MBus watchdog timer reset.

By default, it is set to `24'h16E360`, which corresponds to 10 seconds with 150kHz MBus clock (CLK_MBC) speed.

To disable the MBus Watchdog Timer, set `MBUS_WD_THRESHOLD`=0.

Writing any value in `MBUS_WD_RESET_REQ` temporarily resets the counter, so the counter starts from 0 again.

Writing 1 in `MBUS_WD_RESET_OVERRIDE` immediately resets the counter to 0. The counter remains 0 until `MBUS_WD_RESET_OVERRIDE` becomes 0.

8.5.2 MBus Transactions

This section explains how to send an arbitrary-length MBus messages.

Layer Ctrl Interrupt Layer Ctrl can accept an interrupt, which can initiate any MBus-related operations, including Register Read/Write and Memory Read/Write. In order to implement this functionality, Layer Ctrl has the following inputs.

- INT_VECTOR: 1-bit interrupt signal
- INT_FU_ID[3:0]: MBus Functional ID
- INT_CMD[95:0]: Interrupt Command.
- INT_CMD_LEN[1:0]: Specifies which portion of INT_CMD is valid. See

Table 16: INT_CMD_LEN Setting

INT_CMD_LEN	Description
2'h0	Do nothing.
2'h1	Use INT_CMD[95:64]. INT_CMD[63:0] is ignored.
2'h2	Use INT_CMD[95:32]. INT_CMD[31:0] is ignored.
2'h3	Use INT_CMD[95:0].

Once INT_VECTOR is asserted, Layer Ctrl behaves as if it received an MBus message shown in Figure 1.

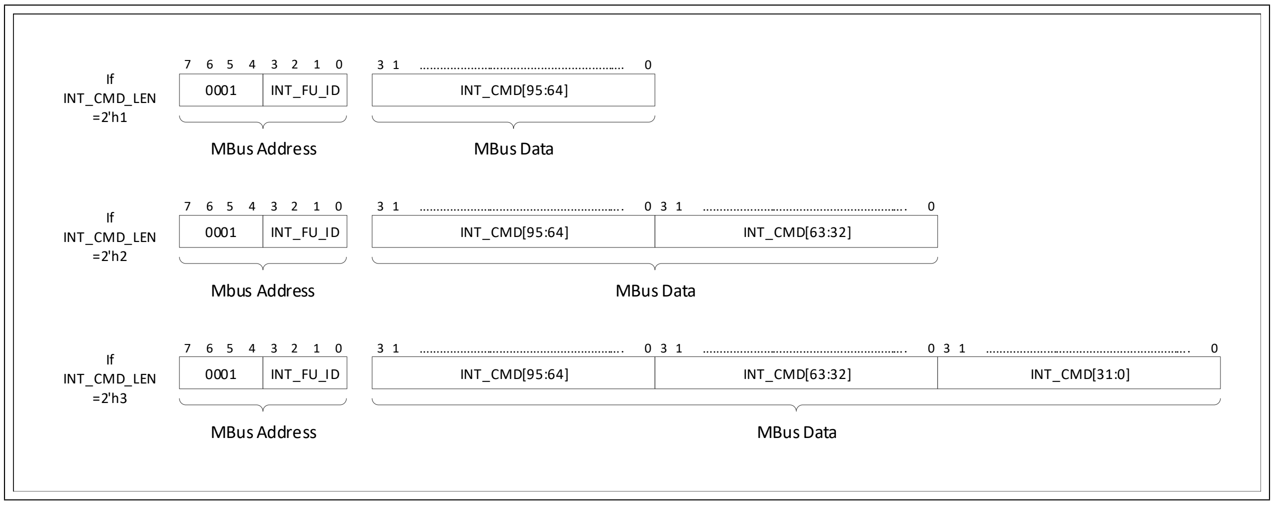

Figure 1: Layer Ctrl Interrupt Behavior

Sending an Arbitrary-Length MBus Message User can send an arbitrary-length MBus message by utilizing MEM_READ operation (INT_FU_ID = `4'h3`). Follow the steps below:

1. Store the MBus DATA (of any length) to be sent in SRAM. Suppose that an MBus DATA of 32×N bits is stored in SRAM where the memory address range is [MEM_START_ADDR] to [MEM_START_ADDR + 4×(N-1)]
2. Set MBUS_CMD0 = MBUS_TARGET_ADDR[7:0], `2'h0`, LENGTH[19:0] , where LENGTH = N - 1.
3. Set MBUS_CMD1 = MEM_START_ADDR[31:0]
4. Set MBUS_CMD2 = `32'h0` (not used)
5. Set MBUS_FUID_LEN = `2'h2`, `4'h3`

Once this procedure is completed, PRC/PRE will send out an MBus message shown in Figure 2.

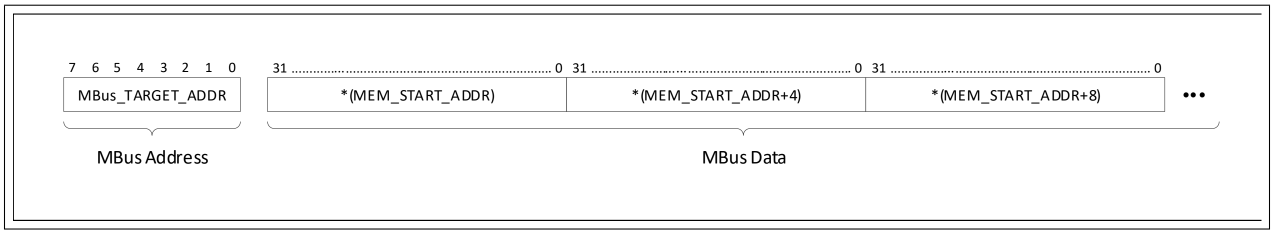

Figure 2: Arbitrary-Length MBus Tx

PRC/PRE provides a simpler way to send a 32-bit MBus message. See Section 8.5.3.

Other MBus Operations By setting INT_FU_ID differently, user can make Layer Ctrl do any MBus-related operation. See *MBus Specification* document for more details.

8.5.3 Quick MBus Transaction (32-bit MBus Tx)

This section explains how to quickly send a 32-bit MBus message.

QUICK_MBUS (`0xA0003XX0`), Default: `32'h00000000`, W

Triggers the Quick MBus transaction. Bit[11:4] in Address (marked as XX) will be the MBus Target Address. Any data written into this address will be sent out as MBus DATA.

Operation Details For example, if user wants to send out an MBus message where MBus Target ADDR = `0x40`, MBus Data = `0x12345678`, the user shall write `0x12345678` at `0xA0003400`.

In C code, this can be written as shown in Figure 3.

```
uint32_t quick_MBus_tx (uint8_t addr, uint32_t data) {
    uint32_t MBus_addr = \texttt{0xA0003000} | (addr << 4);
    *((volatile uint32_t *) MBus_Addr) = data;
    return 1;
}
```

Figure 3: C Code Example for Quick MBus

9 GOC/EP

This is a Global Optical Communication (GOC) / External Programming (EP) (GOC/EP) digital backend that translates GOC and EP inputs into MBus messages or Memory Writes.

9.1 Power and Clock

- GOC/EP is operating at VDD_0P6_LC, which is a power-gated 0.6V, shared with Layer Controller and CPU.
- GOC/EP use the same clock as Layer Controller (CLK_LC) to sample the incoming GOC/EP Clock..
- GOC/EP are reset by the Layer Controller Reset signal (lc_reset_b).

9.2 Memory-Mapped Addresses

Table 17 shows memory mapped address for GOC/EP. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

GOC_DBE_* signals are used for AMB_GOC. See Section 18.

Table 17: Memory Map for GOC/EP

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
0xA0000020	0x08	GOCEP_CHIP_ID	W/R	16'h0000	[15:0]
0xA0000038	0x0E	GOCEP_TIMEOUT	W/R	24'h1E8480	[23:0]
0xA000003C	0x0F	GOCEP_FREEZE_RUN_CPU	W/R	1'h0	[23]
		GOC_DBE_MAIN_FAIL_TYPE	W/R	2'h0	[14:13]
		GOC_DBE_MAIN_FAIL	W/R	1'h0	[12]
		GOCEP_CPU_WAS_RUNNING	W/R	1'h0	[8]
		GOCEP_FAIL_PREMATURE	W/R	1'h0	[7]
		GOCEP_FAIL_TIMEOUT	W/R	1'h0	[6]
		GOCEP_FAIL_MEM_CHECKSUM	W/R	1'h0	[5]
		GOCEP_FAIL_HEADER_CHECKSUM	W/R	1'h0	[4]
		GOCEP_FAIL_CHIP_ID	W/R	1'h0	[3]
		GOCEP_FAIL	W/R	1'h0	[2]
		GOCEP_PASS	W/R	2'h0	[1:0]

9.3 Retentive GOC/EP Configuration Registers

9.3.1 0xA00000XX

GOCEP_CHIP_ID Reg 0x08 (0xA0000020), Bit Field: [15:0], Default: 16'h0000, W/R

Chip ID. By default, it is set to 16'h0000.

GOCEP_TIMEOUT Reg 0x0E (0xA0000038), Bit Field: [23:0], Default: 24'h1E8480, W/R

Timeout value for GOC/EP transactions. If there is no activity until the internal timer reaches the value specified in GOCEP_TIMEOUT, the GOC/EP transaction is regarded as fail. The internal timer is running off

Core Clock (CLK_CORE). If set to 0, there is no time-out check during GOC/EP transactions. By default, it is set to `24'h1E8480`, which corresponds to 25 seconds with 80kHz Core Clock (CLK_CORE) speed.

GOCEP_FREEZE_RUN_CPU Reg `0x0F` (`0xA000003C`), Bit Field: [23], Default: `1'h0`, W/R

Set if RUN_CPU is frozen due to a failure during multi-word GOC/EP programming. Once it is set, RUN_CPU cannot be changed, until `GOCEP_FREEZE_RUN_CPU` is reset to 0, See Section 9.4.

GOCEP_CPU_WAS_RUNNING Reg `0x0F` (`0xA000003C`), Bit Field: [8], Default: `1'h0`, W/R

Set if the CPU had been already running before a GOC or EP message started.

GOCEP_FAIL_PREMATURE Reg `0x0F` (`0xA000003C`), Bit Field: [7], Default: `1'h0`, W/R

This flag is not used in this PRC/PRE version.

GOCEP_FAIL_TIMEOUT Reg `0x0F` (`0xA000003C`), Bit Field: [6], Default: `1'h0`, W/R

Set if GOC or EP Active is high without seeing a positive clock edge from GOC_CLK or EP_CLK for the number of Core Clock (CLK_CORE) cycles specified in `GOCEP_TIMEOUT`. See Section 9.4.

GOCEP_FAIL_MEM_CHECKSUM Reg `0x0F` (`0xA000003C`), Bit Field: [5], Default: `1'h0`, W/R

Set if Memory parity of GOC/EP's packet is incorrect. See Section 9.4.

GOCEP_FAIL_HEADER_CHECKSUM Reg `0x0F` (`0xA000003C`), Bit Field: [4], Default: `1'h0`, W/R

Set if HEADER_CHECKSUM of GOC/EP's packet is incorrect. See Section 9.4.

GOCEP_FAIL_CHIP_ID Reg `0x0F` (`0xA000003C`), Bit Field: [3], Default: `1'h0`, W/R

Set if CHIP_ID section of the CONTROL section of GOC/EP's packet does not match the PRC/PRE/s `GOCEP_CHIP_ID`. See Section 9.4.

GOCEP_FAIL Reg `0x0F` (`0xA000003C`), Bit Field: [2], Default: `1'h0`, W/R

Set if a GOC or EP message fails by any reason above. See Section 9.4.

GOCEP_PASS Reg `0x0F` (`0xA000003C`), Bit Field: [1:0], Default: `2'h0`, W/R

Set if a GOC or EP message passes. See Section 9.4.

9.4 Operation

See Figure 4 for GOC/EP's state diagram, and see Figure 5 for its timing diagram.

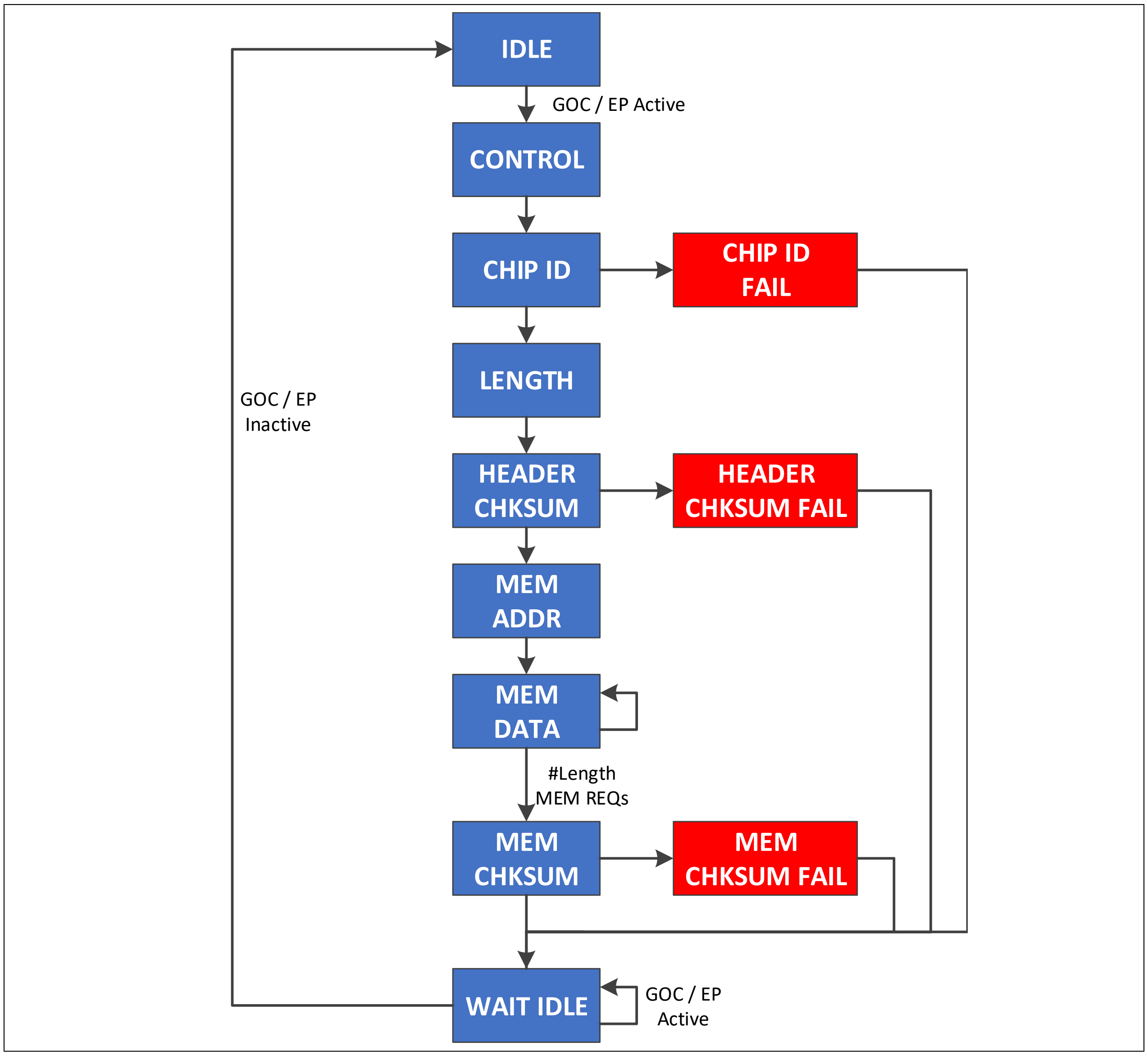

Figure 4: GOC/EP State Diagram

9.4.1 GOC and EP

GOC/EP is a digital backend that translates GOC and EP inputs into MBus messages or Memory Writes. GOC/EP requires three input signals: GOCEP_ACTIVE, GOCEP_CLK, GOCEP_DATA.

GOC uses a separate GOC module (Standard GOC in PRCv18/PREv18/PREv18E, and Ambient GOC in PRCv18G), which receives light pattern through a solar cell connected to PRC/PRE's VSOL pad. This GOC module decodes the received light pattern and generates GOC_ACTIVE, GOC_CLK, GOC_DATA, which are then fed into GOC/EP; GOC_ACTIVE becomes GOCEP_ACTIVE, GOC_CLK becomes GOCEP_CLK, and GOC_DATA becomes GOCEP_DATA. For more detaild information on Standard GOC or Ambient GOC, see M3 Processor Layer Family (Version 18) Documentation.

EP uses three pads: EP_ACTIVE, EP_CLK, EP_DATA. These three signals are directly fed into GOC/EP. EP_ACTIVE becomes GOCEP_ACTIVE, EP_CLK becomes GOCEP_CLK, and EP_DATA becomes GOCEP_DATA. User is responsible for correctly driving these three pads according to the timing diagram shown in Figure 5

Upon GOCEP_ACTIVE signal going high, GOC/EP wakes up the system. In order for the system to be fully stable, it is recommended to wait for >2 CPU clock cycles ($\sim 20\mu$s) before driving the first bit of GOCEP_DATA.

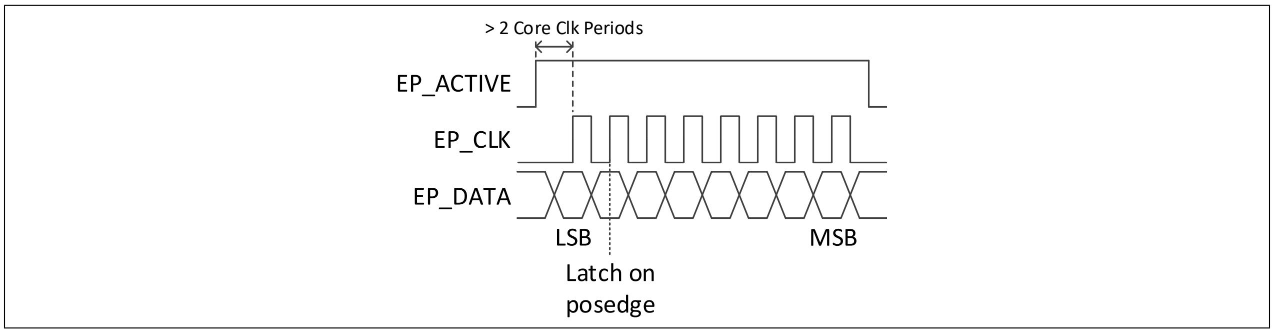

Figure 5: EP Timing Diagram

9.4.2 Packet

- Each GOC/EP PACKET is composed of HEADER section and DATA section as seen in Figure 6.
- In order to wake up the system without affecting SRAM data, use the following:
  - RUN_CPU in CONTROL must be set to 1. See Table 18.
  - GEN_IRQ in CONTROL must be set appropriately. See Table 18.
  - LENGTH = `16'h0`
  - MEM_ADDR = `32'h0`
  - MEM_DATA#0 = `32'h00002000`
  - MEM_CHECKSUM = `8'h20`

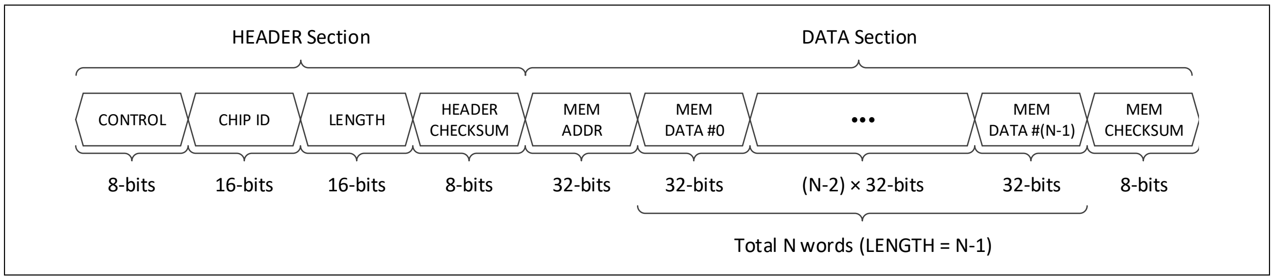

Figure 6: GOC/EP Packet

HEADER Section

- HEADER section is further composed of four sub-sections (CONTROL, CHIP_ID, LENGTH, HEADER_CHECKSUM) as shown in Figure 6, and must be sent by this order.
- In each sub-section, it is sent from LSB to MSB. Thus, for the 48-bit HEADER section, GOCEP_DATA[0] is CONTROL[0], GOCEP_DATA[7] is CONTROL[7], GOCEP_DATA[8] is CHIP_ID[0], GOCEP_DATA[23] is CHIP_ID[15], GOC_DATA[24] is LENGTH[0], GOC_DATA[39] is LENGTH[15], GOC_DATA[40] is HEADER_CHECKSUM[0], and GOC_DATA[47] is HEADER_CHECKSUM[7].
- See Table 18 for CONTROL sections's description.

- If Cortex-M0 was already running, GOC/EP ignores the RUN_CPU bit.
- CHIP_ID section is sued to match PRC/PRE's `GOCEP_CHIP_ID`.
- LENGTH indicated how many MEM data sections will be sent (LENGTH = 'Num of Words' - 1)
- HEADER Checksum must be calculated as shown below.

$$\begin{aligned}\mathtt{HEADER}_{\mathtt{CHECKSUM}}[7:0] = \mathtt{CONTROL}[7:0] &+ \left(\sum_{\mathtt{i}=0}^{1}(\mathtt{CHIP}_{\mathtt{ID}}[8(\mathtt{i}+1)-1:8\mathtt{i}])\right) \\ &+ \left(\sum_{\mathtt{i}=0}^{1}(\mathtt{LENGTH}[8(\mathtt{i}+1)-1:8\mathtt{i}])\right)\end{aligned} \tag{1}$$

Table 18: GOC/EP Control Section's Description

Bit #	Name	Description
0	CHIP_ID MASK[0]	Mask Byte 0 of CHIP_ID (1 = Mask)
1	CHIP_ID MASK[1]	Mask Byte 1 of CHIP_ID (1 = Mask)
2	CHIP_ID MASK[2]	Mask Byte 2 of CHIP_ID (1 = Mask)
3	CHIP_ID MASK[3]	Mask Byte 3 of CHIP_ID (1 = Mask)
4	RSVD	Reserved
5	RSVD	Reserved
6	GEN_IRQ	Set the behavior upon GOC/EP completion when RUN_CPU=1: 1: Generate IRQ to M0 without affecting CPU reset state 0: Reset CPU and re-start code execution without generating IRQ
7	RUN_CPU	Set the behavior upon GOC/EP completion: 1: Run CPU. User should set GEN_IRQ (Bit#6) appropriately 0: Send the MBus sleep message.

DATA Section

- The first 32-bit of DATA Section is MEM_ADDR, which is the Memory Start Address to which MEM_DATA#0 is written. MEM_DATA#n is written to 'MEM_ADDR + (4×n)'
- Each 32-bit DATA section is sent from LSB to MSB.
- MEM Checksum must be calculated as shown below.

$$\mathtt{MEM}_{\mathtt{CHECKSUM}}[7:0] = \left(\sum_{\mathtt{i}=0}^{3}(\mathtt{MEM}_{\mathtt{ADDR}}[8(\mathtt{i}+1)-1:8\mathtt{i}])\right) + \left(\sum_{\mathtt{j}=0}^{\mathtt{LENGTH}}\left(\sum_{\mathtt{i}=0}^{3}(\mathtt{MEM}_{\mathtt{DATA}}[\mathtt{j}][8(\mathtt{i}+1)-1:8\mathtt{i}])\right)\right) \tag{2}$$

9.4.3 Message Types

GOC/EP messages have two types: Program and Trigger. It has different pass/fail behaviors depending on the message type.

Program Message GOC/EP message are regarded as 'Program' if LENGTH in HEADER is not 0, i.e., the message has more than 32-bit data. Actual memory write operations happen immediately once it receives each 32-bit memory data. Thus it could corrupt memory if it has a MEM Checksum error or Timeout error while receiving MEM_DATA.

See Table 19 for pass/fail behaviors.

Table 19: GOC/EP Program Message Pass/Fail Behaviors

	HEADER has RUN_CPU=0	**HEADER has RUN_CPU=1**
Pass	Make `GOCEP_FREEZE_RUN_CPU`=0, Send MBus sleep message	Make `GOCEP_FREEZE_RUN_CPU`=0 Start running CPU
Chip ID Fail / HEADER Checksum Fail / Timeout in HEADER	Send MBus flag message, followed by MBus sleep message	
MEM Checksum Fail / Timeout in MEM_DATA	Make `GOCEP_FREEZE_RUN_CPU`=1, Send MBus flag message, followed by MBus sleep message	

Trigger Message GOC/EP messages are regarded as 'Trigger' if LENGTH in HEADER is 0, i.e., the message has only 32-bit data. Actual memory write operation is pending until it confirms that the received MEM_ADDR and MEM_DATA has no Checksum error.

See Table 20 for pass/fail behaviors.

Table 20: GOC/EP Trigger Message Pass/Fail Behaviors

	If `GOCEP_FREEZE_RUN_CPU`=1	**If `GOCEP_FREEZE_RUN_CPU`=0**
Pass	Write MEM_DATA in MEM_ADDR. Send MBus flag message, followed by MBus sleep message.	Write MEM_DATA in MEM_ADDR. If HEADER has RUN_CPU=1, then start running CPU. If HEADER has RUN_CPU=0, then send MBus sleep message.
Chip ID Fail / HEADER Checksum Fail / Timeout in HEADER	Send MBus flag message, followed by MBus sleep message	
MEM Checksum Fail / Timeout in MEM_DATA	Does NOT write MEM_DATA in MEM_ADDR. Send MBus flag message, followed by MBus sleep message	

9.4.4 GOC/EP Flags

GOC/EP has PASS/FAIL flag registers in MBus Register File.

PASS In case where a GOC/EP transaction is successfully completed, `GOCEP_PASS` is set. See Table 21.

Table 21: `GOCEP_PASS` Register

`GOCEP_PASS`**[1:0]**	**Description**
2'b0X	GOC/EP reset timer(s) and then sent the MBus sleep message
2'b01	GOC/EP reset the CPU and re-started the program execution
2'b11	GOC/EP generated an IRQ to Cortex-M0

FAIL In case where a GOC/EP transaction is terminated due to an error, the following flag registers are set.

- `GOCEP_FAIL` is set.
- `GOCEP_FAIL_CHIP_ID` is set if the CHIP_ID section of the CONTROL section does not match the PRC/PRE's `GOCEP_CHIP_ID`. GOC/EP ignores all subsequent transmissions until EP_ACTIVE/GOC_ACTIVE is pulled low and high again.
- `GOCEP_FAIL_HEADER_CHECKSUM` is set if HEADER Checksum is not correct. GOC/EP ignores all subsequent transmissions until EP_ACTIVE/GOC_ACTIVE is pulled low and high again.
- `GOCEP_FAIL_MEM_CHECKSUM` is set if MEM Checksum is not correct. GOC/EP ignores all subsequent transmissions until EP_ACTIVE/GOC_ACTIVE is pulled low and high again.
- `GOCEP_FAIL_TIMEOUT` is set if GOC or EP Active is high without a seeing a positive clock edge from GOC_CLK or EP_CLK for the number of Core Clock cycles specified in `GOCEP_TIMEOUT`. GOC/EP ignores all subsequent transmission and goes back to either its previous sleep or active state. The default value of `GOCEP_TIMEOUT` is `24'h1E8480`, which corresponds to 25 seconds with 80kHz Core Clock speed. If `GOCEP_TIMEOUT` is set to 0, there is no timeout check during GOC/EP transactions.

9.4.5 Hard Reset

GOC/EP can trigger the hard-reset of the system. Once triggered, PRC/PRE behaves as if the watch-dog timer expired, as described in Section 5.6. The system will lose any data and configuration, and the separate PMU layer will re-boot the system. In order to trigger the hard-reset using GOC/EP, use the following:

- CONTROL = `8'h3F`
- CHIP_ID = `16'hDEAD`
- LENGTH = `16'h0000`
- MEM_ADDR = `32'hDEADBEEF`
- MEM_DATA#0 = `32'hDEADBEEF`

9.5 Interrupt

GPIO/EP can be configured to generate a Cortex-M0 IRQ shown in Table 22, at the end of the GOC/EP transaction if the HEADER section is configured to do so. The interrupt can be further masked by Cortex-M0 NVIC registers.

Table 22: GOC/EP-related Cortex-M0 IRQ

Interrupt Port	Description
IRQ[2]	Interrupt when GOC/EP finishes

10 16-bit Timer

10.1 Power and Clock

- 16-bit Timer is operating at VDD_0P6_LC, which is a power-gated 0.6V, shared with Layer Ctrl.
- 16-bit Timer uses the same clock as Layer Ctrl (clk_lc).
- 16-bit Timer are reset by the Layer Ctrl Reset signal (lc_reset_b).

10.2 Memory-Mapped Addresses

Table 23 shows memory mapped address for 16-bit Timer. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

Table 23: Memory Map for 16-bit Timer

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
0xA0001000	-	GO	W/R	1'h0	[0]
0xA0001004	-	CMP0	W/R	16'hFFFF	[15:0]
0xA0001008	-	CMP1	W/R	16'hFFFF	[15:0]
0xA000100C	-	IRQEN	W/R	2'h0	[1:0]
0xA0001010	-	CNT	W/R	16'h0000	[15:0]
0xA0001014	-	STAT	W/R	6'h00	[5:0]

10.3 Non-Retentive 16-bit Timer Configuration Registers

10.3.1 0xA00010XX

GO (0xA0001000), Default: 1'h0, W/R

Go Register. Starts 16-bit Timer's CNT Register counting.

CMP0 (0xA0001004), Default: 16'hFFFF, W/R

Compare Register#0. Value to count up to interrupt.

CMP1 (0xA0001008), Default: 16'hFFFF, W/R

Compare Register#1. Value to count up to interrupt.

IRQEN (0xA000100C), Default: 2'h0, W/R

Interrupt Enable.

If IRQEN[1] is 1, 16-bit Timer generates an interrupt when CNT becomes equal to CMP1.

If IRQEN[0] is 1, 16-bit Timer generates an interrupt when CNT becomes equal to CMP0.

CNT (`0xA0001010`), Default: `16'h0000`, W/R

Counter Register. Indicates what count 16-bit Timer is currently at. When `CNT` reaches its maximum (`16'hFFFF`), it will wrap around, meaning that `CNT` starts from 0 again.

STAT (`0xA0001014`), Default: `6'h00`, W/R

Status Register. It indicates whether `CNT` has equaled `CMP0` or `CMP1`. Must be cleared by user.

`STAT`[5:2] is reserved.

`STAT`[1] is set 1 when `CNT` becomes equal to `CMP1`, only if `IRQEN`[1] is 1.

`STAT`[0] is set 1 when `CNT` becomes equal to `CMP0`, only if `IRQEN`[0] is 1.

10.4 Operation

Set the desired registers correctly, and assert `GO`.

10.5 Interrupt

16-bit Timer can be configured to generate a Cortex-M0 IRQ shown in Table 24, if enabled by `IRQEN`, when `CNT` becomes `CMP0` or `CMP1`. The interrupt can be further masked by Cortex-M0 NVIC registers.

Table 24: 16-bit Timer-related Cortex-M0 IRQ

Interrupt Port	Description
IRQ[4]	Interrupt from 16-bit Timer

11 32-bit Timer

11.1 Power and Clock

- 32-bit Timer is operating at VDD_0P6_LC, which is a power-gated 0.6V, shared with Layer Ctrl.
- 32-bit Timer uses the same clock as Layer Ctrl (clk_lc).
- 32-bit Timer are reset by the Layer Ctrl Reset signal (lc_reset_b).

11.2 Memory-Mapped Addresses

Table 25 shows memory mapped address for 32-bit Timer. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

Table 25: Memory Map for 32-bit Timer

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
0xA0001100	-	GO	W/R	1'h0	[0]
0xA0001104	-	CMP	W/R	32'hFFFFFFFF	[31:0]
0xA0001108	-	ROI	W/R	1'h0	[0]
0xA000110C	-	CNT	W/R	32'h00000000	[31:0]
0xA0001110	-	STAT	W/R	1'h0	[0]

11.3 Non-Retentive 32-bit Timer Configuration Registers

11.3.1 0xA00011XX

GO (0xA0001100), Default: 1'h0, W/R

Go Register. Starts 32-bit Timer's CNT Register counting.

CMP (0xA0001104), Default: 32'hFFFFFFFF, W/R

Compare Register. Value to count up to interrupt.

ROI (0xA0001108), Default: 1'h0, W/R

Reset-on-Interrupt Register. Automatically resets CNT when CNT has equaled CMP and starts counting again.

CNT (0xA000110C), Default: 32'h00000000, W/R

Counter Register. Indicates what count 32-bit Timer is currently at.

If ROI is set, it will automatically reset to 0 and start counting again when CNT has equaled CMP, otherwise it stays at CMP when CNT has equaled CMP.

STAT (0xA0001114), Default: 6'h00, W/R

Status Register. It indicates whether CNT has equaled CMP. Must be cleared by user.

11.4 Operation

Set the desired registers correctly, and assert `GO`.

11.5 Interrupt

32-bit Timer generates a Cortex-M0 IRQ shown in Table 26 when `CNT` becomes `CMP`. The interrupt can be further masked by Cortex-M0 NVIC registers.

Table 26: 32-bit Timer-related Cortex-M0 IRQ

Interrupt Port	Description
IRQ[3]	Interrupt from 32-bit Timer

12 PUF

12.1 Power and Clock

- PUF is operating at VDD_1P2, which is an always-on 1.2V. However, it can be power-gated using `PUF_CHIP_ID_SLEEP`.
- PUF is clockless. See the operations in Section 12.4.
- PUF does not have a dedicated reset signal. However, the sleep signal (`PUF_CHIP_ID_SLEEP`) also resets the PUF state.

12.2 Memory-Mapped Addresses

Table 27 shows memory mapped address for PUF. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

In the table, `ENABLE_SOFT_RESET` and `WAKEUP_ON_PEND_REQ` are not related to PUF. See Section 6 and Section 4 for their detailed information.

Table 27: Memory Map for PUF

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
0xA0000034	0x0D	PUF_CHIP_ID	R	24'h000000	[23:0]
0xA000006C	0x1B	PUF_CHIP_ID_SLEEP	W/R	1'h1	[6]
		PUF_CHIP_ID_ISOLATE	W/R	1'h1	[5]
		ENABLE_SOFT_RESET	W/R	1'h0	[4]
		WAKEUP_ON_PEND_REQ	W/R	4'h0	[3:0]
0xA00000FC	0x3F	CHIP_ID	W/R	24'h000000	[23:0]

12.3 Retentive PUF Configuration Registers

12.3.1 `0xA00000XX`

PUF_CHIP_ID Reg `0x0D` (`0xA0000034`), Bit Field: [23:0], Default: `24'h000000`, R

This read-only register is mapped to PUF output. See Section 12.4 for how to get the valid output from PUF.

PUF_CHIP_ID_SLEEP Reg `0x1B` (`0xA000006C`), Bit Field: [6], Default: `1'h1`, W/R

If 1, PUF becomes power-gated and its output becomes floating. Thus, it is recommended to set `PUF_CHIP_ID_ISOLATE` to 1, before setting `PUF_CHIP_ID_SLEEP` to 1.

If 0, PUF becomes powered up and starts generating its output.

PUF_CHIP_ID_ISOLATE Reg `0x1B` (`0xA000006C`), Bit Field: [5], Default: `1'h1`, W/R

If 1, it masks the actual PUF output, and `PUF_CHIP_ID` becomes 0 regardless of the actual PUF output.

If 0, the actual PUF output is directly exposed to (and connected to) `PUF_CHIP_ID`.

CHIP_ID Reg `0x3F` (`0xA00000FC`), Bit Field: [23:0], Default: `24'h000000`, W/R

This register provides a dedicated space to store the PUF output. It is user's responsibility to save the PUF output in this register.

12.4 Operation

Follow the below instruction to get the PUF output.

1. Make sure `PUF_CHIP_ID_SLEEP`=1 and `PUF_CHIP_ID_ISOLATE`=1.
2. Set `PUF_CHIP_ID_SLEEP`=0.
3. Wait until the PUF generates its output and get stable. 1ms should be enough.
4. Set `PUF_CHIP_ID_ISOLATE`=0.
5. Store the value at `PUF_CHIP_ID` in `CHIP_ID`.

Follw the below instruction to turn off the PUF.

1. Set `PUF_CHIP_ID_ISOLATE`=1.
2. Set `PUF_CHIP_ID_SLEEP`=1.

Note that, `PUF_CHIP_ID_ISOLATE` must become 1 before setting `PUF_CHIP_ID_SLEEP`=1, in order to prevent any excessive power consumption.

13 Wakeup Timer

13.1 Power and Clock

- Wakeup Timer is operating at VDD_1P2, which is an always-on 1.2V.
- Wakeup Timer uses CLK_SLP, which is generated by Sleep Clock Generator (Section 16).
- Wakeup Timer is reset by the global POR reset signal. It can be also reset by user.

13.2 Memory-Mapped Addresses

Table 28 shows memory mapped address for Wakeup Timer. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

Table 28: Memory Map for Wakeup Timer

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
`0xA0000044`	`0x11`	`WUP_WREQ_EN`	W/R	`1'h0`	[23]
		`WUP_CNT_SAT`	W/R	`22'h000000`	[21:0]
`0xA0000048`	`0x12`	`WUP_CNT_VAL`	R	`22'h000000`	[21:0]
`0xA0001300`	-	`WUP_RESET_REQ`	W	`1'h0`	[0]

13.3 Retentive Wakeup Timer Configuration Registers

13.3.1 `0xA00000XX`

WUP_WREQ_EN Reg `0x11` (`0xA0000044`), Bit Field: [23], Default: `1'h0`, W/R

If 1, Wakeup Timer generates a wakeup request when `WUP_CNT_VAL` becomes equal to `WUP_CNT_SAT`.

If 0, Wakeup Timer does not generate a wakeup request.

WUP_CNT_SAT Reg `0x11` (`0xA0000044`), Bit Field: [21:0], Default: `22'h000000`, W/R

Wakeup Timer saturation value. If `WUP_WREQ_EN`=1, Wakeup Timer generates a wakeup request when `WUP_CNT_VAL` becomes equal to `WUP_CNT_SAT`.

WUP_CNT_VAL Reg `0x12` (`0xA0000048`), Bit Field: [21:0], Default: `22'h000000`, R

Wakeup Timer's current count value.

13.4 Non-Retentive Wakeup Timer Configuration Registers

13.4.1 `0xA00013XX`

WUP_RESET_REQ (`0xA0001300`), Default: `1'h0`, W

Reset Request. Writing any value in this register will reset Wakeup Timer. The reset is effective immediately, and is released CLK_SLP's next positive edge.

13.5 Operation

Wakeup Timer is always running once the global POR reset becomes released.

13.6 Interrupt

WakeupT Timer generates a Cortex-M0 IRQ shown in Table 29, right after the system wakes up by the wakeup request generated by Wakeup Timer. The interrupt can be further masked by Cortex-M0 NVIC registers.

Table 29: Wakeup Timer-related Cortex-M0 IRQ

Interrupt Port	Description
IRQ[0]	Interrupt right after system wakes up

13.7 Wakeup Request

Wakeup Timer generates the wakeup request if the following conditions are met.

- `WUP_WREQ_EN`=1
- `WUP_CNT_VAL` becomes equal to `WUP_CNT_SAT`.

Once the above conditions are met, Wakeup Timer generates a wakeup request which triggers MBus Master Ctrl to pull down DOUT signal.

14 XO Timer [PREv18(A/E) Only]

14.1 Power and Clock

- XO Timer is operating at VDD_1P2, which is an always-on 1.2V.
- XO Timer uses CLK_XO, which is generated by XO Driver (Section 19).
- XO Timer is reset by the global POR reset signal. It can be also reset by user.

14.2 Memory-Mapped Addresses

Table 30 shows memory mapped address for XO Timer. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

Table 30: Memory Map for XO Timer

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
0xA000004C	0x13	XOT_ENABLE	W/R	1'h0	[23]
		XOT_MODE	W/R	1'h0	[22]
		XOT_WREQ_EN	W/R	1'h0	[21]
		XOT_IRQ_EN	W/R	1'h0	[20]
		XOT_FORCE_COUT	W/R	1'h0	[19]
		XOT_CNT_SAT_LOWER	W/R	16'h0000	[15:0]
0xA0000050	0x14	XOT_CNT_SAT_UPPER	W/R	16'h0000	[15:0]
0xA0000054	0x15	XOT_CNT_EN	R	1'h0	[23]
		XOT_COUT_EN	R	1'h0	[22]
		XOT_CNT_VAL_LOWER	R	16'h0000	[15:0]
0xA0000058	0x16	XOT_CNT_VAL_UPPER	R	16'h0000	[15:0]
0xA0001400	-	XOT_RESET_REQ	W	1'h0	[0]
0xA0001404	-	XOT_START_CNT	W	1'h0	[0]
0xA0001408	-	XOT_STOP_CNT	W	1'h0	[0]
0xA000140C	-	XOT_START_COUT	W	1'h0	[0]
0xA0001410	-	XOT_STOP_COUT	W	1'h0	[0]

14.3 Retentive XO Timer Configuration Registers

14.3.1 0xA00000XX

XOT_ENABLE Reg 0x13 (0xA000004C), Bit Field: [23], Default: 1'h0, W/R (PREv18(A/E))

XO Timer enable signal.

If 0, XO Timer is clock-gated and all flip-flops are being reset. If 1, XO Timer becomes active and running.

Going 0 $\rightarrow$ is internally synchronized, but going 1 $\rightarrow$ is asynchronous.

XOT_MODE Reg 0x13 (0xA000004C), Bit Field: [22], Default: 1'h0, W/R (PREv18(A/E))

Selects between Non-Masking mode and Masking mode. See Table 31 and Section 14.5.

XOT_ENABLE must be set to 0 before changing XOT_MODE.

Table 31: XOT_MODE

XOT_MODE	**Description**
1'h0	Non-Masking Mode
1'h1	Masking Mode

XOT_WREQ_EN Reg 0x13 (0xA000004C), Bit Field: [21], Default: 1'h0, W/R (PREv18(A/E))

If 1, XO Timer generates a wakeup request when {XOT_CNT_VAL_UPPER, XOT_CNT_VAL_LOWER} becomes equal to {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_UPPER}.

If 0, XO Timer does not generate a wakeup request.

XOT_IRQ_EN Reg 0x13 (0xA000004C), Bit Field: [20], Default: 1'h0, W/R (PREv18(A/E))

If 1, XO Timer generates a Cortex-M0 interrupt when {XOT_CNT_VAL_UPPER, XOT_CNT_VAL_LOWER} becomes equal to {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_UPPER}.

If 0, XO Timer does not generate a Cortex-M0 interrupt.

XOT_FORCE_COUT Reg 0x13 (0xA000004C), Bit Field: [19], Default: 1'h0, W/R (PREv18(A/E))

If 1, it forces outputting XO_CLK pad output. This is mostly for debugging purpose.

If 0, it follows the internal operation to decide when to output XO_CLK pad output.

XOT_ENABLE must be set to 0 before changing XOT_FORCE_COUT.

XOT_CNT_SAT_LOWER Reg 0x13 (0xA000004C), Bit Field: [15:0], Default: 16'h0000, W/R (PREv18(A/E))

XO Timer saturation value (Lower 16 bits). If XOT_WREQ_EN=1, XO Timer generates a wakeup request when {XOT_CNT_VAL_UPPER, XOT_CNT_VAL_LOWER} becomes equal to {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_UPPER}.

XOT_CNT_SAT_UPPER Reg 0x14 (0xA0000050), Bit Field: [15:0], Default: 16'h0000, W/R (PREv18(A/E))

XO Timer saturation value (Upper 16 bits). If XOT_WREQ_EN=1, XO Timer generates a wakeup request when {XOT_CNT_VAL_UPPER, XOT_CNT_VAL_LOWER} becomes equal to {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_UPPER}.

XOT_CNT_EN Reg 0x15 (0xA0000054), Bit Field: [23], Default: 1'h0, R

For debugging. It maps to en_cnt_lc signal.

XOT_COUT_EN Reg 0x15 (0xA0000054), Bit Field: [22], Default: 1'h0, R

For debugging. It maps to en_cout_lc signal.

XOT_CNT_VAL_LOWER Reg 0x15 (0xA0000054), Bit Field: [15:0], Default: 16'h0000, R

XO Timer's current count value (Lower 16 bits).

XOT_CNT_VAL_UPPER Reg 0x16 (0xA0000058), Bit Field: [15:0], Default: 16'h0000, R

XO Timer's current count value (Upper 16 bits).

14.4 Non-Retentive XO Timer Configuration Registers

14.4.1 `0xA00014XX`

XOT_RESET_REQ (`0xA0001400`), Default: `1'h0`, W

Reset Request. Writing any value in this register will reset XO Timer. The reset is effective immediately, and is released CLK_XO's next positive edge.

XOT_START_CNT (`0xA0001404`), Default: `1'h0` ,W

Upon writing into this signal, XO Timer starts the internal counter. See Section 14.5.

XOT_STOP_CNT (`0xA0001408`), Default: `1'h0` ,W

Upon writing into this signal, XO Timer stops the internal counter. See Section 14.5.

XOT_START_COUT (`0xA000140C`), Default: `1'h0` ,W

Upon writing into this signal, XO Timer starts outputting XO_CLK pad output. See Section 14.5.

XOT_STOP_COUT (`0xA0001410`), Default: `1'h0` ,W

Upon writing into this signal, XO Timer stops outputting XO_CLK pad output. See Section 14.5.

14.5 Operation

XO Timer can be used in several scenarios.

14.5.1 Normal Use as a Sleep Timer

Set registers as shown below, then write anything in `XOT_START_CNT`. After this, user needs to make PRC/PRE sends an MBus Sleep message. Figure 7 shows waveforms when {`XOT_CNT_SAT_UPPER`, `XOT_CNT_SAT_LOWER`}=`0x10`.

- `XOT_ENABLE`=1
- `XOT_MODE`=0
- `XOT_WREQ_EN`=1
- `XOT_IRQ_EN`=0
- {`XOT_CNT_SAT_UPPER`, `XOT_CNT_SAT_LOWER`} as desired.

Note that, in Figure 7, `XOT_START_CNT` and CLK_XO seem to be synchronous, but they are not. `XOT_START_CNT` is synchronous to the CPU/Layer Ctrl clock (CLK_LC). XO Timer has an internal synchronizer to handle any incoming signal from the CLK_LC domain.

Once the system wakes up by the XO Timer, the XO counter (`XOT_CNT_VAL_UPPER` and `XOT_CNT_VAL_LOWER`) automatically goes back to 0 and stays there. User does not need to reset it again.

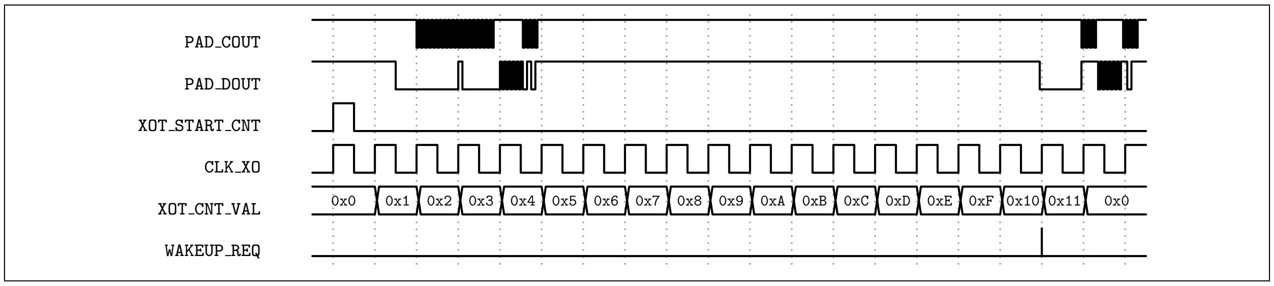

Figure 7: XO Timer Used as a Normal Sleep Timer

14.5.2 Normal Use as a Timer in System Active Mode

Set registers as shown below, then write anything in XOT_START_CNT. Once the XO counter (XOT_CNT_VAL_*) becomes equal to the saturation value (XOT_CNT_SAT_*), XO Timer generates the Cortex-M0 interrupt (if XOT_IRQ_EN=1), and the counter goes back to 0.

Figure 8 shows the behavior when {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_LOWER}=0xC.

- XOT_ENABLE=1
- XOT_MODE=0
- XOT_WREQ_EN=0
- XOT_IRQ_EN=1
- {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_LOWER} as desired.

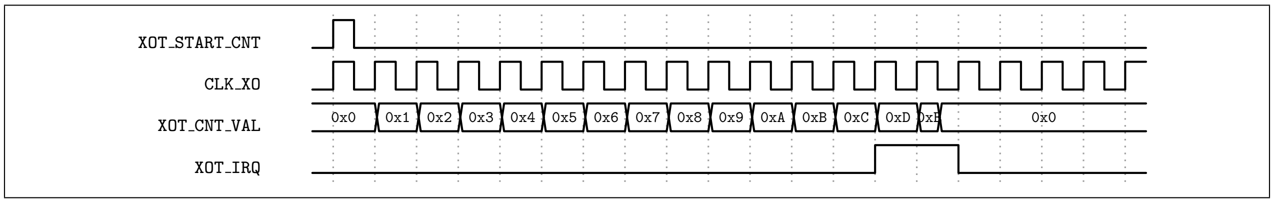

Figure 8: XO Timer Used for XO_CLK Pad Output Masking in System Sleep Mode

14.5.3 Free Running XO_CLK Pad Output in System Active Mode

Set registers as shown below, then write anything in XOT_START_COUT, which will output XO_CLK pad output. Write anything in XOT_STOP_COUT to stop the XO_CLK pad output.

Figure 9 shows the behavior.

- XOT_ENABLE=1
- XOT_MODE=0
- XOT_WREQ_EN=0
- XOT_IRQ_EN=0

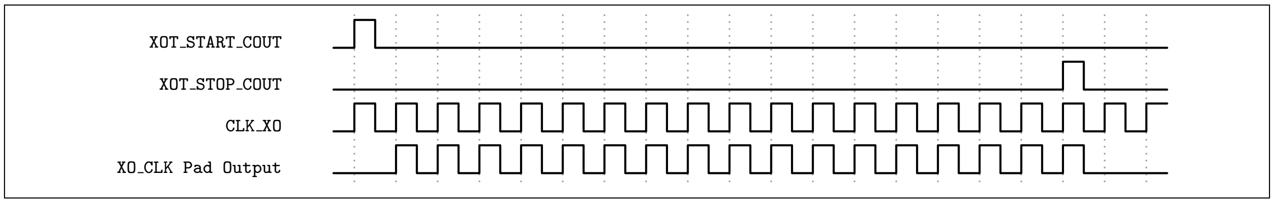

Figure 9: XO Timer Used for Free Running XO_CLK Pad Output in System Active Mode

14.5.4 Free Running XO_CLK Pad Output & Sleep Timer

Set registers as shown below, then write anything in XOT_START_COUT, which will output XO_CLK pad output. Right before sending out the MBus Sleep message, write anything in XOT_START_CNT.

Once the system wakes up by XO Timer, write anything in XOT_STOP_COUT to stop the XO_CLK pad output, if needed.

Figure 10 shows the behavior when {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_LOWER}=0xA.

- XOT_ENABLE=1
- XOT_MODE=0
- XOT_WREQ_EN=0
- XOT_IRQ_EN=0
- {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_LOWER} as desired.

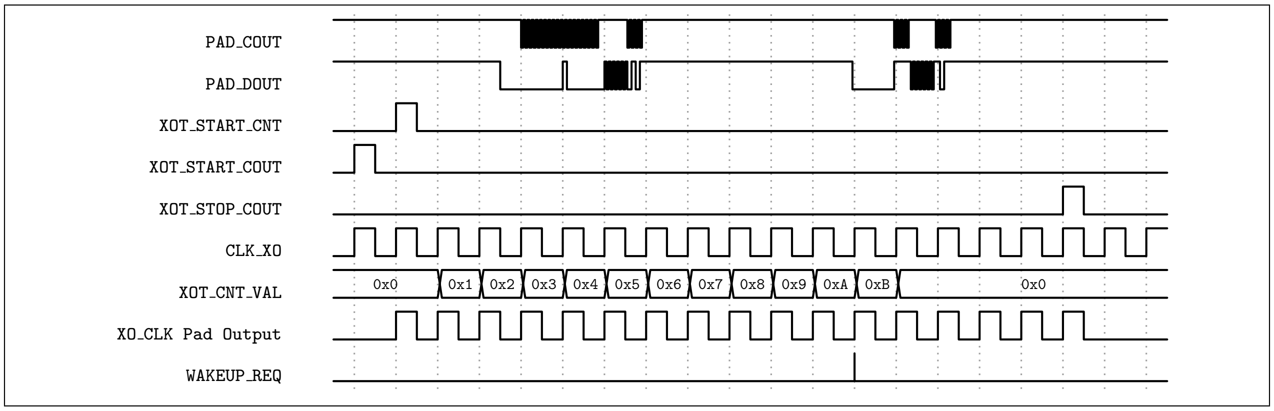

Figure 10: XO Timer Used for Free Running XO_CLK Pad Output & Sleep Timer

14.5.5 XO_CLK Pad Output Masking in System Active Mode

Set registers as shown below, then write anything in XOT_START_COUT, which will output XO_CLK pad output.

When needed, write anything in XOT_STOP_COUT, then the XO_CLK pad output stops and the internal XO Timer counter starts running.

Once the counter (XOT_CNT_VAL_*) becomes equal to the saturation value (XOT_CNT_SAT_*), XO Timer resumes outputting the XO_CLK pad output.

In other words, XO Timer skips a number of clock cycles in XO_CLK, and the number of cycles to be skipped is equal to (XOT_CNT_SAT + 1), where XOT_CNT_SAT is {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_LOWER}.

It keeps repeating this behavior whenever XOT_STOP_COUT is written. In order to exit from this mode, write 0 in XOT_MODE. Also, XOT_STOP_COUT must be written to stop the XO_CLK pad output.

Figure 11 shows the behavior when {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_LOWER}=0x3.

- XOT_ENABLE=1
- XOT_MODE=1
- XOT_WREQ_EN=0
- XOT_IRQ_EN as desired.
- {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_LOWER} as desired.

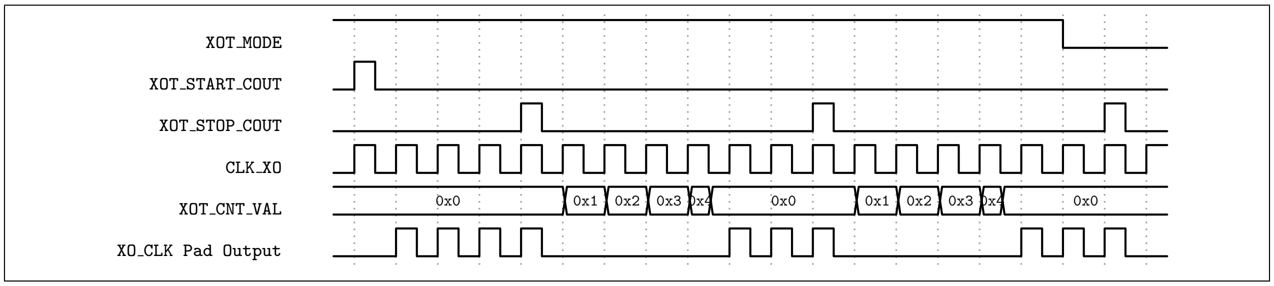

Figure 11: XO Timer Used for XO_CLK Pad Output Masking in System Active Mode

14.5.6 XO_CLK Pad Output Masking with System Sleep Mode

Set registers as shown below, then write anything in XOT_START_COUT, which will output XO_CLK pad output.

When needed, write anything in XOT_STOP_COUT, then the XO_CLK pad output stops and the internal XO Timer counter starts running. User needs to send the MBus Sleep message right after writing to XOT_STOP_COUT.

Once the counter (XOT_CNT_VAL_*) becomes equal to the saturation value (XOT_CNT_SAT_*), XO Timer resumes outputting the XO_CLK pad output. XO Timer also generates the wakeup request at the same time.

In other words, XO Timer skips a number of clock cycles in XO_CLK, and the number of cycles to be skipped is equal to (XOT_CNT_SAT + 1), where XOT_CNT_SAT is {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_LOWER}. Once it skips the designated number of clock cycles, it resumes the XO_CLK output and also generates the wakeup request so that the system can wake up.

In order to exit from this mode, write 0 in XOT_MODE when the system is in Active mode. Also, XOT_STOP_COUT must be written to stop the XO_CLK pad output.

Figure 12 shows the behavior when {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_LOWER}=0x9.

- XOT_ENABLE=1
- XOT_MODE=1
- XOT_WREQ_EN=1
- XOT_IRQ_EN as desired.
- {XOT_CNT_SAT_UPPER, XOT_CNT_SAT_LOWER} as desired.

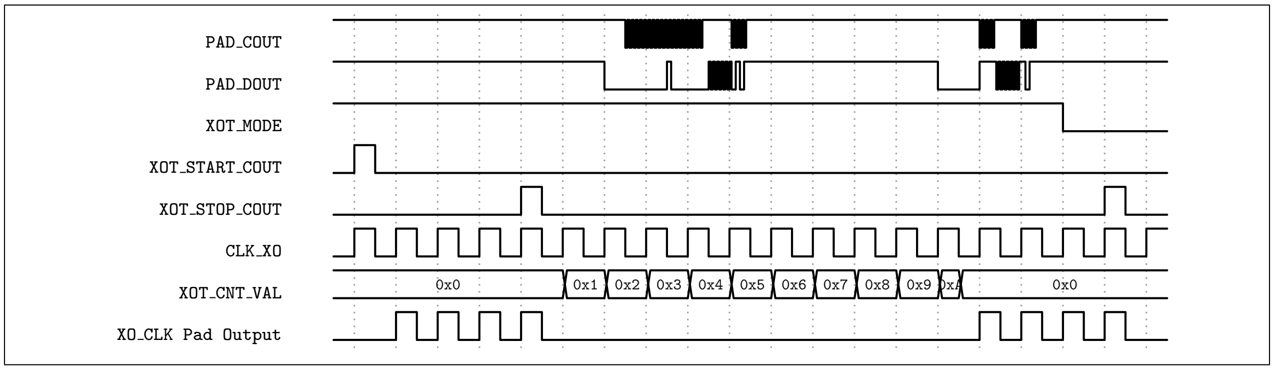

Figure 12: XO Timer Used for XO_CLK Pad Output Masking in System Sleep Mode

14.6 Interrupt

XO Timer generates a Cortex-M0 IRQ shown in Table 32, once the counter (XOT_CNT_VAL_*) becomes equal to the saturation value (XOT_CNT_SAT_*), and XOT_IRQ_EN is set to 1.

The interrupt can be further masked by Cortex-M0 NVIC registers.

Table 32: XO Timer-related Cortex-M0 IRQ

Interrupt Port	Description
IRQ[19]	Interrupt from XO Timer

14.7 Wakeup Request

XO Timer generates the wakeup request if the following conditions are met.

- XOT_WREQ_EN=1
- The counter (XOT_CNT_VAL_*) becomes equal to the saturation value (XOT_CNT_SAT_*)

Once the above conditions are met, XO Timer generates a wakeup request which triggers MBus Master Ctrl to pull down DOUT signal.

15 CPU/MBus Clock Generator

15.1 Power and Clock

- CPU/MBus Clock Generator is operating at VDD_1P2, which is an always-on 1.2V.
- CPU/MBus Clock Generator generates two clocks: CLK_CORE and CLK_MBC. CLK_CORE is used for CPU, Arbitration, Layer Ctrl, etc. CLK_MBC is used in the MBus and the source of the MBus Clock (COUT).
- CLK_CORE is reset by Layer Ctrl Clock Enable signal (lc_clken), and CLK_MBC is reset by the MBus Sleep signal (mbc_sleep_b).

15.2 Memory-Mapped Addresses

Table 33 shows memory mapped address for CPU/MBus Clock Generator. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

GOC_* signals are used for GOC operations and its clock generator. See Section 17 or Section 18.

Table 33: Memory Map for CPU/MBus Clock Generator

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
0xA000002C (Non-PRCv18G)	0x0B	CLK_GEN_HIGH_FREQ	W/R	1'h0	[21]
		CLK_GEN_DIV_CORE	W/R	3'h3	[20:18]
		CLK_GEN_DIV_MBC	W/R	3'h2	[17:15]
		CLK_GEN_RING	W/R	2'h1	[14:13]
		GOC_CLK_GEN_SEL_DIV	W/R	2'h1	[11:10]
		GOC_CLK_GEN_SEL_FREQ	W/R	3'h7	[9:7]
		GOC_ONECLK_MODE	W/R	1'h0	[6]
		GOC_SEL_DLY	W/R	2'h0	[5:4]
		GOC_SEL	W/R	4'h8	[3:0]
0xA000002C (PRCv18G)	0x0B	GOC_DBG_ENABLE	W/R	1'h0	[23]
		GOC_DBG_SEL	W/R	1'h0	[22]
		CLK_GEN_HIGH_FREQ	W/R	1'h0	[21]
		CLK_GEN_DIV_CORE	W/R	3'h3	[20:18]
		CLK_GEN_DIV_MBC	W/R	3'h2	[17:15]
		CLK_GEN_RING	W/R	2'h1	[14:13]
		GOC_R_OFFSET	W/R	2'h1	[11:10]
		GOC_MAIN_CLK_TUNE_DIV	W/R	2'h1	[9:8]
		GOC_BASE_CLK_TUNE_DIV	W/R	2'h0	[7:6]
		GOC_TRAIN_MAX_ERROR	W/R	2'h1	[5:4]
		GOC_MAVG_THRESHOLD	W/R	2'h1	[3:2]
		GOC_MF_THRESHOLD	W/R	2'h1	[1:0]

15.3 Retentive CPU/MBus Clock Generator Configuration Registers

Below registers control the clock frequencies. See Section 15.4 for details.

15.3.1 `0xA00000XX`

CLK_GEN_HIGH_FREQ Reg `0x0B` (`0xA000002C`), Bit Field: [21], Default: `1'h0`, W/R

If 1, uses the High-Frequency ring.

If 0, uses the Regular-Frequency ring.

CLK_GEN_DIV_CORE Reg `0x0B` (`0xA000002C`), Bit Field: [20:18], Default: `3'h3`, W/R

Division ratio for CLK_CORE.

CLK_GEN_DIV_MBC Reg `0x0B` (`0xA000002C`), Bit Field: [17:15], Default: `3'h2`, W/R

Division ratio for CLK_MBC.

CLK_GEN_RING Reg `0x0B` (`0xA000002C`), Bit Field: [14:13], Default: `2'h1`, W/R

Clock Ring tab selection.

15.4 Operation

CPU/MBus Clock Generator generates 2 different clocks (CLK_CORE and CLK_MBC) and has 2 modes (Regular and High Frequency).

- CLK_CORE drives the Cortex-M0, Layer Ctrl, and SRAM.
- CLK_MBC drives the MBus Controller.

CLK_CORE and CLK_MBC are derived from the same clock ring; they are divided version of the clock ring, and the division ratio is separately set by `CLK_GEN_DIV_CORE` and `CLK_GEN_DIV_MBC`.

CLK_CORE and CLK_MBC frequencies can be configured through the retentive registers shown in Section 15.3.

Table 34 shows the simulation (TT, 25C) results of the CLK_CORE and CLK_MBC frequencies.

For CLK_CORE, DIV in the table indicates `CLK_GEN_DIV_CORE`, and RING indicates `CLK_GEN_RING`.

For CLK_MBC, DIV in the table indicates `CLK_GEN_DIV_MBC`, and RING indicates `CLK_GEN_RING`.

Table 34: CPU/MBus Clock Generator Frequency and Power (Simulation @ TT, 25C)

DIV	RING	CLK_GEN_HIGH_FREQ=0			CLK_GEN_HIGH_FREQ=1		
		Freq (kHz)	Period (us)	Power (uW)	Freq (kHz)	Period (us)	Power (uW)
3'h0	2'h3	1182.7	0.85	2.25	10779.5	0.09	13.64
3'h0	2'h2	933.7	1.07	1.77	8641.6	0.12	10.89
3'h0	2'h1	770.2	1.30	1.47	7178.1	0.14	9.07
3'h0	2'h0	689.9	1.45	1.31	6405.7	0.16	8.06
3'h1	2'h3	591.4	1.69	2.21	5389.6	0.19	13.23
3'h1	2'h2	466.8	2.14	1.74	4320.3	0.23	10.57
3'h1	2'h1	385.1	2.60	1.44	3589.3	0.28	8.80
3'h1	2'h0	345.0	2.90	1.28	3203.4	0.31	7.82
3'h2	2'h3	295.7	3.38	2.16	2694.8	0.37	12.82
3'h2	2'h2	233.4	4.28	1.70	2160.1	0.46	10.23
3'h2	2'h1	192.5	5.19	1.41	1794.8	0.56	8.52
3'h2	2'h0	172.5	5.80	1.26	1601.5	0.62	7.57
3'h3	2'h3	147.8	6.76	2.13	1347.5	0.74	12.66
3'h3	2'h2	116.7	8.57	1.68	1080.1	0.93	9.98
3'h3	2'h1	96.3	10.39	1.38	897.4	1.11	8.31
3'h3	2'h0	86.2	11.60	1.25	800.8	1.25	7.39
3'h4	2'h3	73.9	13.53	2.11	673.7	1.48	12.30
3'h4	2'h2	58.3	17.14	1.66	540.1	1.85	9.82
3'h4	2'h1	48.1	20.78	1.37	448.7	2.23	8.18
3'h4	2'h0	43.1	23.19	1.22	400.4	2.50	7.26
3'h5	2'h3	37.0	27.06	2.09	336.9	2.97	12.17
3'h5	2'h2	29.2	34.27	1.65	270.0	3.70	9.71
3'h5	2'h1	24.1	41.55	1.36	224.3	4.46	8.09
3'h5	2'h0	21.6	46.38	1.21	200.2	5.00	7.19
3'h6	2'h3	18.5	54.11	2.08	168.4	5.94	12.09
3'h6	2'h2	14.6	68.55	1.64	135.0	7.41	9.65
3'h6	2'h1	12.0	83.10	1.35	112.2	8.91	8.04
3'h6	2'h0	10.8	92.77	1.21	100.1	9.99	7.14
3'h7	2'h3	9.2	108.23	2.08	84.2	11.87	12.07
3'h7	2'h2	7.3	137.10	1.63	67.5	14.81	9.61
3'h7	2'h1	6.0	166.19	1.35	56.1	17.83	8.00
3'h7	2'h0	5.4	185.54	1.21	50.0	19.98	7.11

16 Sleep Clock Generator

16.1 Power and Clock

- Sleep Clock Generator is operating at VDD_1P2, which is an always-on 1.2V.
- Sleep Clock Generator generates CLK_SLP, which clocks the Wakeup Timer (Section 13).
- Sleep Clock Generator is reset by the global POR reset.

16.2 Memory-Mapped Addresses

There is no MMIO for Sleep Clock Generator, as it is always running once the POR reset becomes released.

16.3 Operation

Sleep Clock Generator generates CLK_SLP, which clocks the Wakeup Tiemr (Section 13).

Sleep Clock Generator is always running once the POR reset becomes released.

17 GOC FE (Front End) [Non-PRCv18G Only]

17.1 Power and Clock

- GOC FE is operating at VDD_1P2, which is an always-on 1.2V.
- GOC FE is running at CLK_GOC, which is generated by GOC Clock Generator included in GOC FE.
- GOC FE is reset by the global POR reset.

17.2 Memory-Mapped Addresses

Table 35 shows memory mapped address for GOC FE. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

CLK_GEN_* signals are used for CPU/MBus Clock Generator. See Section 15.

Table 35: Memory Map for GOC FE

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
`0xA000002C` (Non-PRCv18G)	`0x0B`	`CLK_GEN_HIGH_FREQ`	W/R	`1'h0`	[21]
		`CLK_GEN_DIV_CORE`	W/R	`3'h3`	[20:18]
		`CLK_GEN_DIV_MBC`	W/R	`3'h2`	[17:15]
		`CLK_GEN_RING`	W/R	`2'h1`	[14:13]
		`GOC_CLK_GEN_SEL_DIV`	W/R	`2'h1`	[11:10]
		`GOC_CLK_GEN_SEL_FREQ`	W/R	`3'h7`	[9:7]
		`GOC_ONECLK_MODE`	W/R	`1'h0`	[6]
		`GOC_SEL_DLY`	W/R	`2'h0`	[5:4]
		`GOC_SEL`	W/R	`4'h8`	[3:0]

17.3 Retentive GOC FE Configuration Registers

17.3.1 `0xA00000XX`

GOC_CLK_GEN_SEL_DIV Reg `0x0B` (`0xA000002C`), Bit Field: [11:10], Default: `2'h1`, W/R (Non-PRCv18G)

Controls the division ratio of GOC Clock Generator. See Table 38.

GOC_CLK_GEN_SEL_FREQ Reg `0x0B` (`0xA000002C`), Bit Field: [9:7], Default: `3'h7`, W/R (Non-PRCv18G)

Controls the raw frequency of GOC Clock Generator. See Table 38.

GOC_ONECLK_MODE Reg `0x0B` (`0xA000002C`), Bit Field: [6], Default: `1'h0`, W/R (Non-PRCv18G)

If 0, GOC FE enables the clock division for low power signature detection mode. This is the recommended default setting.

If 1, GOC FE disables sampling clock division.

GOC_SEL_DLY Reg `0x0B` (`0xA000002C`), Bit Field: [5:4], Default: `2'h0`, W/R (Non-PRCv18G)

Selects the number of clock cycles (ticks) from GOC_CLK for GOC_DATA acquisition. See Table 36.

Table 36: GOC_SEL_DLY

GOC_SEL_DLY	Num of Clock Ticks
2'h0	7 Clock Ticks
2'h1	14 Clock Ticks
2'h2	21 Clock Ticks
2'h3	29 Clock Ticks

GOC_SEL Reg `0x0B` (`0xA000002C`), Bit Field: [3:0], Default: `4'h8`, W/R (Non-PRCv18G)

Selects the front-end pull-down resistors. Setting a bit to 1 activates the corresponding resistor. See Table 37.

Table 37: GOC_SEL

GOC_SEL	Resistor Size
[0]	(W/L) = 2um / 14um
[1]	(W/L) = 3um / 6um
[2]	(W/L) = 6um / 4um
[3]	(W/L) = 5um / 0.5um

17.4 Operation

Table 38 shows the simulation results of GOC Clock Generator. In the table, SEL_FREQ indicates GOC_CLK_GEN_SEL_FREQ, and SEL_DIV indicates GOC_CLK_GEN_SEL_DIV.

Table 38: GOC Clock Generator Frequency and Power (Simulation @ TT, 25C)

SEL_FREQ	SEL_DIV	TT @ 25C		FF @ 120C		SS @ -20C	
		Freq(Hz)	Power(pW)	Freq(Hz)	Power(pW)	Freq(Hz)	Power(pW)
3'h0	2'h0	197.6	393	27940	42630	2.7	6.1
	2'h1	99.3	386.2	13790	41460	1.3	6.1
	2'h2	49.7	384.2	7009	41310	0.6	6
	2'h3	24.8	381.1	3506	40800	0.3	6
3'h1	2'h0	223.9	401	32260	44610	3.2	6.3
	2'h1	112.4	392.8	16130	43180	1.7	6.2
	2'h2	56.2	392.2	8092	42660	0.9	6.2
	2'h3	28	388.8	4047	42210	0.4	6.2
3'h2	2'h0	287.3	424.8	41250	48520	3.8	6.5
	2'h1	144.6	416.2	20630	46740	2	6.5
	2'h2	72.3	415.5	10360	46200	0.9	6.1
	2'h3	36.1	410.5	5181	45500	0.5	6.2
3'h3	2'h0	343.6	441.2	59280	56160	3.5	6
	2'h1	172.5	426	29750	53660	1.4	5.6
	2'h2	86.3	426.6	14900	52850	0.9	5.9
	2'h3	43	419.8	7449	51940	0.5	5.9
3'h4	2'h0	5267	2442	359700	187800	70	32.2
	2'h1	2640	2248	180300	175200	34.9	29.4
	2'h2	1320	2202	90210	171600	17.6	29
	2'h3	660.1	2121	45090	165600	8.8	28.1
3'h5	2'h0	7670	3062	512713	236213	101.4	39.7
	2'h1	3842	2775	256792	211612	51.6	36.6
	2'h2	1929	2731	128612.9	208098	25.6	35.5
	2'h3	961.8	2622	64270	199200	12.8	33.7
3'h6	2'h0	11550	3928	718559	283233	156.9	51.6
	2'h1	5774	3468	360116.5	252204	78.4	45.8
	2'h2	2898	3393	180200	245000	39	43.9
	2'h3	1448	3221	90090	232100	19.5	41.7
3'h7	2'h0	38970	10840	2498000	790700	505.2	135.1
	2'h1	19460	9315	1258000	687100	252.6	115.8
	2'h2	9724	8998	624400	658600	126.2	111.2
	2'h3	4880	8459	314600	622700	63.1	104

18 AMB GOC (Ambient GOC) [PRCv18G Only]

Ambient GOC (AMB GOC) consists of two parts: Analog Front-End (AFE) and Digital Back-End (DBE).

18.1 Power and Clock

- AMB AMB GOC is operating at VDD_1P2, which is an always-on 1.2V.
- AMB AMB GOC is running at BASE_CLK and MAIN_CLK clcoks. These two clocks are generated within AFE and then fed into DBE.
- AMB AMB GOC is reset by the global POR reset.

18.2 Memory-Mapped Addresses

Table 39 shows memory mapped address for AMB AMB GOC. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

CLK_GEN_* signals are used for CPU/MBus Clock Generator. See Section 15.

GOCEP_* signals are used for GOC/EP Digital Backend. See Section 9.

18.3 Retentive AMB AMB GOC Configuration Registers

18.3.1 `0xA00000XX`

GOC_DBG_ENABLE Reg `0x0B` (`0xA000002C`), Bit Field: [23], Default: `1'h0`, W/R (PRCv18G)

FIXME

GOC_DBG_SEL Reg `0x0B` (`0xA000002C`), Bit Field: [22], Default: `1'h0`, W/R (PRCv18G)

FIXME

GOC_R_OFFSET Reg `0x0B` (`0xA000002C`), Bit Field: [11:10], Default: `2'h1`, W/R (PRCv18G)

FIXME

GOC_MAIN_CLK_TUNE_DIV Reg `0x0B` (`0xA000002C`), Bit Field: [9:8], Default: `2'h1`, W/R (PRCv18G)

It sets the clock frequency used to sample the GOC light in Main Mode.

GOC_BASE_CLK_TUNE_DIV Reg `0x0B` (`0xA000002C`), Bit Field: [7:6], Default: `2'h0`, W/R (PRCv18G)

It sets the clock frequency used to sample the GOC light in Base Mode.

Table 39: Memory Map for AMB AMB GOC

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
0xA000002C	0x0B	GOC_DBG_ENABLE	W/R	1'h0	[23]
		GOC_DBG_SEL	W/R	1'h0	[22]
		CLK_GEN_HIGH_FREQ	W/R	1'h0	[21]
		CLK_GEN_DIV_CORE	W/R	3'h3	[20:18]
		CLK_GEN_DIV_MBC	W/R	3'h2	[17:15]
		CLK_GEN_RING	W/R	2'h1	[14:13]
		GOC_R_OFFSET	W/R	2'h1	[11:10]
		GOC_MAIN_CLK_TUNE_DIV	W/R	2'h1	[9:8]
		GOC_BASE_CLK_TUNE_DIV	W/R	2'h0	[7:6]
		GOC_TRAIN_MAX_ERROR	W/R	2'h1	[5:4]
		GOC_MAVG_THRESHOLD	W/R	2'h1	[3:2]
		GOC_MF_THRESHOLD	W/R	2'h1	[1:0]
0xA0000030	0x0C	GOC_BASE_SEL_OSC_STAGE	W/R	1'h0	[23]
		GOC_SEL_DIV_AZ	W/R	2'h2	[22:21]
		GOC_AZ_ENB	W/R	1'h0	[20]
		GOC_SIG_GAIN_B	W/R	2'h3	[19:18]
		GOC_R_REF	W/R	3'h2	[17:15]
		GOC_I_VBIAS	W/R	3'h0	[14:12]
		GOC_I_AMBIENT	W/R	3'h1	[11:9]
		GOC_I_SIG_REF	W/R	3'h0	[8:6]
		GOC_I_CMP_1ST	W/R	3'h0	[5:3]
		GOC_I_CMP_2ND	W/R	3'h0	[2:0]
0xA000003C	0x0F	GOCEP_FREEZE_RUN_CPU	W/R	1'h0	[23]
		GOC_DBE_MAIN_FAIL_TYPE	W/R	2'h0	[14:13]
		GOC_DBE_MAIN_FAIL	W/R	1'h0	[12]
		GOCEP_CPU_WAS_RUNNING	W/R	1'h0	[8]
		GOCEP_FAIL_PREMATURE	W/R	1'h0	[7]
		GOCEP_FAIL_TIMEOUT	W/R	1'h0	[6]
		GOCEP_FAIL_MEM_CHECKSUM	W/R	1'h0	[5]
		GOCEP_FAIL_HEADER_CHECKSUM	W/R	1'h0	[4]
		GOCEP_FAIL_CHIP_ID	W/R	1'h0	[3]
		GOCEP_FAIL	W/R	1'h0	[2]
		GOCEP_PASS	W/R	2'h0	[1:0]

Table 40: GOC_MAIN_CLK_TUNE_DIV Setting

GOC_MAIN_CLK_TUNE_DIV[1:0]	Clock Frequency
2'h0	141.32 kHz
2'h1	159.43 kHz
2'h2	197.85 kHz
2'h3	260.75 kHz

Table 41: GOC_BASE_CLK_TUNE_DIV Setting

GOC_BASE_CLK_TUNE_DIV[1:0]	Clock Frequency
2'h0	253.0 Hz
2'h1	126.5 Hz
2'h2	63.2 Hz
2'h3	31.6 Hz

GOC_TRAIN_MAX_ERROR Reg `0x0B` (`0xA000002C`), Bit Field: [5:4], Default: `2'h1`, W/R (PRCv18G)

It sets the maximum amount of error in period estimation allowed during training session. In order to successfully estimate the period, the equation below must be met.

$$\frac{|Period_{current} - Period_{previous}|}{Period_{previous}} < \frac{1}{2^{(GOC\_TRANIN\_MAX\_ERROR+1)}} \quad (3)$$

GOC_MAVG_THRESHOLD Reg `0x0B` (`0xA000002C`), Bit Field: [3:2], Default: `2'h1`, W/R (PRCv18G)

It sets the threshold in the Moving Average Filter to distinguish 'Light' and 'No Light'. The Moving Average Filter counts the number of 'Light' samples among the last 8 samples. If the number of 'Light' samples exceeds the value specified by `GOC_MAVG_THRESHOLD`, it decides that it has received 'Light'. Otherwise, it is regarded as 'No Light'.

GOC_MF_THRESHOLD Reg `0x0B` (`0xA000002C`), Bit Field: [1:0], Default: `2'h1`, W/R (PRCv18G)

It sets the threshold in the Matched Filter to distinguish 'Bit 1' and 'Bit 0'. At every estimated period point, the value of the matched filter (MF) is compared with the Matched Filter Threshold (MF_{TH}) specified in Table 42. Note that the value of the Matched Filter (MF) varies from 0 to 16.

0 $\leq$ MF ¡ MF_{TH}: Regarded as 'Bit 0' 16 - MF_{TH} ¡ MF $\leq$ 16: Regarded as 'Bit 1' Otherwise, it is regarded as 'Sync Loss'

Table 42: `GOC_MF_THRESHOLD` Setting

`GOC_MF_THRESHOLD`[1:0]	Matched Filter Threshold (MF_{TH})
2'h0	2
2'h1	4
2'h2	6
2'h3	8

GOC_BASE_SEL_OSC_STAGE Reg `0x0C` (`0xA0000030`), Bit Field: [23], Default: `1'h0`, W/R (PRCv18G)

FIXME

GOC_SEL_DIV_AZ Reg `0x0C` (`0xA0000030`), Bit Field: [22:21], Default: `2'h2`, W/R (PRCv18G)

FIXME

GOC_AZ_ENB Reg `0x0C` (`0xA0000030`), Bit Field: [20], Default: `1'h0`, W/R (PRCv18G)

FIXME

GOC_SIG_GAIN_B Reg `0x0C` (`0xA0000030`), Bit Field: [19:18], Default: `2'h3`, W/R (PRCv18G)

FIXME

GOC_R_REF Reg `0x0C` (`0xA0000030`), Bit Field: [17:15], Default: `3'h2`, W/R (PRCv18G)

FIXME

GOC_I_VBIAS Reg `0x0C` (`0xA0000030`), Bit Field: [14:12], Default: `3'h0`, W/R (PRCv18G)
FIXME

GOC_I_AMBIENT Reg `0x0C` (`0xA0000030`), Bit Field: [11:9], Default: `3'h1`, W/R (PRCv18G)
FIXME

GOC_I_SIG_REF Reg `0x0C` (`0xA0000030`), Bit Field: [8:6], Default: `3'h0`, W/R (PRCv18G)
FIXME

GOC_I_CMP_1ST Reg `0x0C` (`0xA0000030`), Bit Field: [5:3], Default: `3'h0`, W/R (PRCv18G)
FIXME

GOC_I_CMP_2ND Reg `0x0C` (`0xA0000030`), Bit Field: [2:0], Default: `3'h0`, W/R (PRCv18G)
FIXME

GOC_DBE_MAIN_FAIL_TYPE Reg `0x0F` (`0xA000003C`), Bit Field: [14:13], Default: `2'h0`, W/R (PRCv18G)
FIXME

GOC_DBE_MAIN_FAIL Reg `0x0F` (`0xA000003C`), Bit Field: [12], Default: `1'h0`, W/R (PRCv18G)
FIXME

18.4 Operation

18.4.1 Block Diagram

Figure 13 shows a brief block diagram of Ambient GOC. Note that the diode in the left side is not included in PRC/PRE and has to be connected through a pad. The Main Comparator (MAIN_COMP) in AFE and related circuits are power-gated (turned-off) when not in use. DBE is always powered-on; it runs off BASE_CLK by default, and it transitions its mode to use MAIN_CLK when needed.

Right after Power-on-Reset, Ambient GOC goes into Base Mode, where AFE only enables Base Comparator (BASE_COMP) and Base Clock (BASE_CLK), and DBE is clocked by BASE_CLK. BASE_CLK is very slow (31.6Hz 253Hz), so Ambient GOC consumes little power, while constantly sampling incoming light signal using the slow BASE_CLK.

Once Ambient GOC detects the passcode while in Base Mode, it transitions to Main Mode, where AFE enables Main Comparator (MAIN_COMP) and Main Clock (MAIN_CLK), and DBE is clocked by MAIN_CLK. MAIN_CLK has a much faster frequency (141.32kHz 260.75kHz), so the GOC throughput becomes significantly enhanced.

Once it becomes Main Mode, user has to send the passcode again, and then send actual data bits (i.e., header, address, data, etc). Ambient GOC oversamples the incoming light signals. The oversampling ratio must be in a certain range, which will be discussed shortly.

Hence, the incoming light pattern frequency must be much slower than BASE_CLK (when in Base Mode) or MAIN_CLK (when in Main Mode). DBE has the Moving Average Filter to minimize the effect from noise. It stores the last 8 samples, and compares the number of 'Light (i.e., cmp = 1)' samples with the value specified by GOC_MAVG_THRESHOLD. If the number of 'Light (i.e., cmp = 1)' samples in the last 8 samples exceeds the value specified by GOC_MAVG_THRESHOLD, it sets the output mavg to 1, which is then fed into the Matched Filter. Otherwise, mavg is set to 0.

The Matched Filter has a 16-bit shift register which is fed by the Moving Average output (i.e., mavg). At every clock edge, the pattern stored in the shift register is compared with the pre-defined pattern. Basically, this is an XNOR operation, where it is regarded as 'matched' when the corresponding locations in the shift register and the pre-defined pattern have the same value. Then, the number of 'matched' locations is compared with the threshold value specified by GOC_MF_THRESHOLD. If it exceeds the value specified by GOC_MF_THRESHOLD, it sets its output (MF) to 1, which is fed into FSM for further processing.

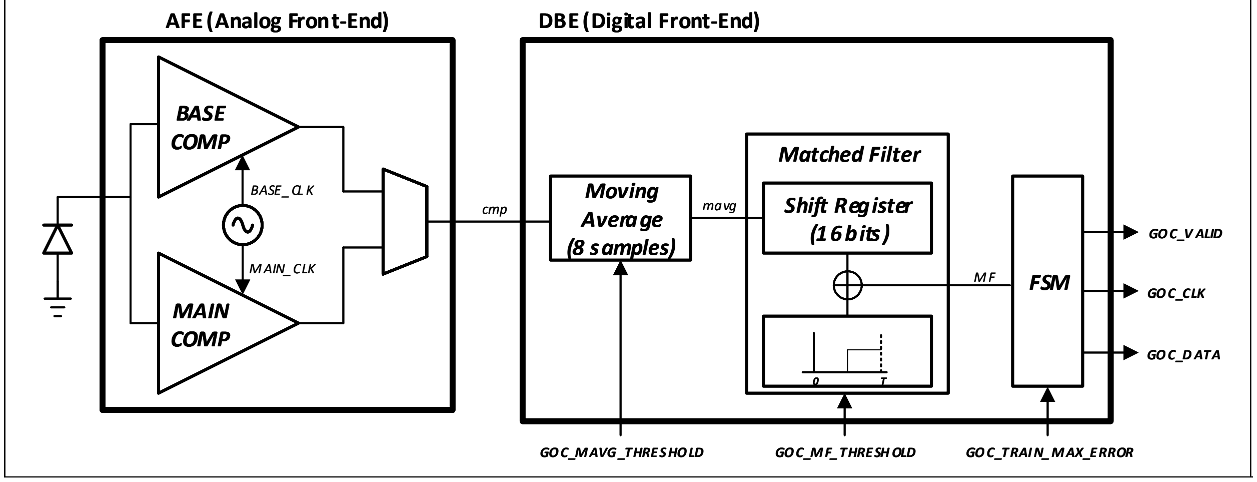

Figure 13: Ambient GOC Block Diagram

18.4.2 Manchester Code

For the light pattern, Ambient GOC requires Manchester code, as shown in Figure 14.

In order to send 'Bit 1', there should be 'no intense light' for the first half period followed by 'intense light' during the second half period.

In order to send 'Bit 0', there should be 'intense light' for the first half period, followed by 'no intense light' during the second half period.

For the definition of 'intense light', see Section 18.4.3. Also, there is a certain constraint on the period (T) due to the oversampling operation. See Section 18.4.4 for further details.

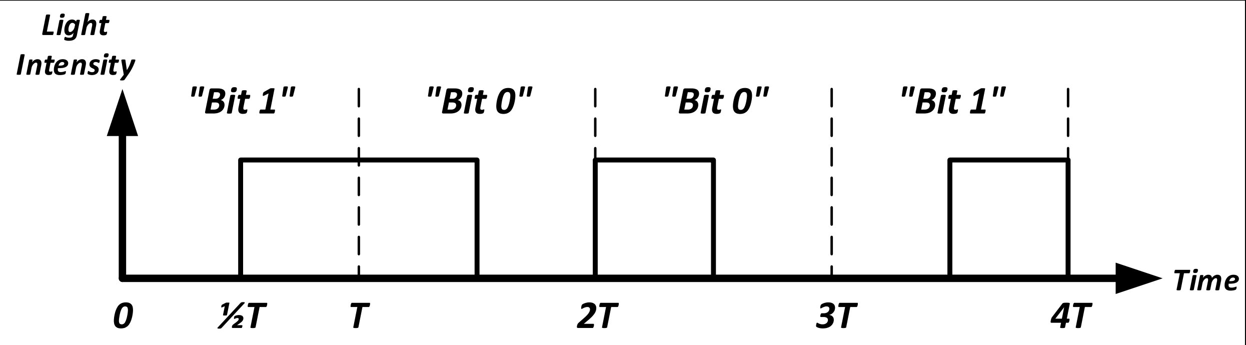

Figure 14: Ambient GOC Manchester Code

18.4.3 Ambient Light Tracking

The definition of 'intense light' varies with ambient light condition. Ambient GOC has internal module that tracks ambient light intensity. At every 256th BASE_CLK edge, it detects the ambient light intensity and adjusts the internal resistance. It can automatically adjust to a wide range of ambient light intensity, from 600 lux to 354.4 klux.

In order to be regarded as 'intense light', it is recommended to use at least 2x stronger light, in term of 'lux', than the ambient light intensity.

18.4.4 Oversampling

Ambient GOC oversamples the incoming light pattern. There is no fixed number regarding the oversampling ratio, since it can automatically adjust to the incoming light period. However, there is a upper/lower limit on the oversampling ratio that user can use.

The oversampling ratio is calculated based on the clock frequency of Ambient GOC. Ambient GOC has two modes, called Base Mode and Main Mode. It uses Base Clock (BASE_CLK) in Base Mode, and uses Main Clock (MAIN_CLK) in Main Mode. Each clock frequency can be changed using MBus Register File, by changing `GOC_BASE_CLK_TUNE_DIV` and `GOC_MAIN_CLK_TUNE_DIV`.

The lower limit of the oversampling ratio is governed by the number of bits in the shifter register in DBE. Thus, it can be 16 in the ideal situation; however, it is recommended to be 20 in reality due to noise/jitter concern.

The upper limit of the oversampling ratio is governed by GOC time-out feature implemented in DBE. In Base Mode, the upper limit is 256 in ideal situation, but it is recommended to be less than that. In Main Mode, the upper limit is 65536, but it is recommended to be less than that.

Table FIXME and Table FIXME show the upper/lower limits on the period ('T' in Figure 14) in Base Mode and Main Mode, respectively. However, it is recommended that user add more margin to take into account non-idealities in real world situations.

FIXME Table

FIXME Table

'Consistency' is much more important than the oversampling ratio, since Ambient GOC trains itself using the incoming light pattern to extract the period information. If the period varies significantly during a GOC transaction, it might result in synchronization errors. In order to avoid the synchronization error, the following condition must be met.

$$\frac{|T(n) - T_{est}(n)|}{T_{est}(n)} < \frac{1}{2^{(GOC\_TRANIN\_MAX\_ERROR+1)}} \tag{4}$$

where n is an integer denoting each occurrence of the symbols (i.e., 'Bit 1' and 'Bit 0' in Figure 14). `GOC_TRAIN_MAX_ERROR` is defined in MBus Register File, and T(n) is the current period detected by Ambient GOC. T_{est}(n) is the estimated period information internally stored in DBE. T_{est}(n) is updated at the end of every period using the equation below.

$$T_{est}(n+1) = \frac{3}{4}T_{est}(n) + \frac{1}{4}T(n) \tag{5}$$

Although the condition given in the equation above must be always met, it is strongly recommended that user put a much stricter constraint on the period variation. More variation in the period will increase the chance of detection errors in Ambient GOC.

18.4.5 How to Use

User should follow the sequence described the below.

Training in Base Mode The whole process begins with training. User needs to send a series of 'Bit 1's to train Ambient GOC so that it can estimate the period of the incoming light pattern. The number of the 'Bit 1's must be at least 10. However, there is no limit on the maximum number of 'Bit 1's. Once Ambient GOC successfully estimates the period using the first $>$ 10 'Bit 1' transmissions, DBE starts waiting for the passcode. DBE keeps waiting for the passcode as long as the next 'Bit 1' arrives before the time-out triggers.

Note that the light pattern period ('T' in Figure 14) must be within the ranges shown in Table FIXME with a sufficient margin.

Passcode in Base Mode Immediately following 'Training in Base Mode', user shall send a passcode using the same period that was used during 'Training in Base Mode.' The passcode is defined as `16'h5472`. The LSB must be sent first. Hence, the bit sequence must be sent like below (from left to right):

```
0 1 0 0 1 1 1 0 0 0 1 0 1 0 1 0
```

Training in Main Mode Once Ambient GOC detects the passcode in Base Mode, it immediately transitions its mode to Main Mode. During the transition, it turns on the Main Comparator (MAIN_COMP) and enables the Main Clock (MAIN_CLK) so that DBE starts running at MAIN_CLK. This process takes a time, typically corresponding to 0.5 1 BASE_CLK period. During this transition time, it is not mandatory to send any light pattern, but it is strongly recommended that user keeps sending a series of 'Bit 1's using the Main Mode speed; the light pattern period ('T' in Figure 14) must be within the ranges shown in Table FIXME with a sufficient margin.

This is because the transition time is immediately followed by 'Training in Main Mode.' The clock speed (MAIN_CLK) becomes much faster than that of BASE_CLK, so if there is no light pattern received when it starts 'Training in Main Mode', it is highly possible to have the time-out triggered.

As same as in 'Training in Base Mode', it is also required to send at least 10 of 'Bit 1's in series in 'Training in Main Mode'. However, it is recommend to send a much more number of 'Bit 1's to take into account the transition time and other non-idealities. A recommended number of 'Bit 1's is larger than 10 + (1.5 $\times$ BASE_CLK_period / (MAIN_CLK_period $\times$ Main_Mode_Oversampling_Ratio))

Passcode in Main Mode Immediately following 'Training in Main Mode', user shall send a passcode using the same period that was used during 'Training in Main Mode.' The passcode is defined as `16'h5472`. The LSB must be sent first. Hence, the bit sequence must be sent like below (from left to right):

```
0 1 0 0 1 1 1 0 0 0 1 0 1 0 1 0
```

Header/Address/Data in Main Mode Immediately following 'Passcode in Main Mode', user shall send the GOC header, address, and data, using the Main Mode speed; the light pattern period ('T' in Figure 14) must be within the ranges shown in Table FIXME with a sufficient margin.

The definition of header, address, and data is as same as in the Standard GOC/EP (Section 9). However, each bit must be properly translated to the Manchester code as shown in Figure 14.

Figure 15 shows the input light pattern required to activate Ambient GOC. Replace the 'Header/Address/Data in Main Mode' part with your actual bit sequence.

In the figure, T_{BASE} indicates the period used in Base Mode, and T_{MAIN} indicates the period used in Main Mode.

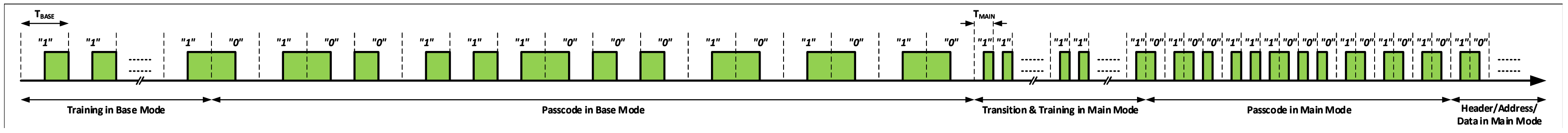

Figure 15: Light Pattern to Activate Ambient GOC

19 XO Driver [PREv18(A/E) Only]

This is a 32.768kHz Crystal Oscillator designed by Dongmin Yoon, and modified by Yu Zeng and Taekwang Jang.

19.1 Power and Clock

- XO Driver is operating at VDD_1P2 and VDD_0P6, which are an always-on 1.2V and 0.6V, respectively. However, XO Driver can be power-gated using `XO_SLEEP`.
- XO Driver generates CLK_XO, which is used in XO Timer (Section 14).
- XO Driver does not have a dedicated reset signal. However, the sleep signal (`XO_SLEEP`) also resets the XO Driver state.

19.2 Memory-Mapped Addresses

Table 43 shows memory mapped address for XO Driver. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

Table 43: Memory Map for XO Driver

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
0xA0000064	0x19	XO_SLEEP	W/R	1'h1	[22]
		XO_ISOLATE	W/R	1'h1	[21]
		XO_EN_DIV	W/R	1'h0	[20]
		XO_S	W/R	3'h0	[19:17]
		XO_SEL_CP_DIV	W/R	1'h0	[16]
		XO_EN_OUT	W/R	1'h0	[15]
		XO_PULSE_SEL	W/R	4'h4	[14:11]
		XO_DELAY_EN	W/R	3'h7	[10:8]
		XO_DRV_START_UP	W/R	1'h0	[7]
		XO_DRV_CORE	W/R	1'h0	[6]
		XO_RP_LOW	W/R	1'h0	[5]
		XO_RP_MEDIA	W/R	1'h0	[4]
		XO_RP_MVT	W/R	1'h0	[3]
		XO_RP_SVT	W/R	1'h0	[2]
		XO_SCN_CLK_SEL	W/R	1'h0	[1]
		XO_SCN_ENB	W/R	1'h1	[0]
0xA0000068	0x1A	XO_CAP_TUNE	W/R	12'h000	[11:0]

19.3 Retentive XO Driver Configuration Registers

19.3.1 `0xA00000XX`

XO_SLEEP Reg `0x19` (`0xA0000064`), Bit Field: [22], Default: `1'h1`, W/R (PREv18(A/E))

XO Driver sleep signal.

If 1, XO Driver becomes power-gated and its output becomes floating. Thus, it is recommended to set `XO_ISOLATE` to 1, before setting `XO_SLEEP` to 1.

If 0, XO Driver becomes powered up and its output becomes floating.

XO_ISOLATE Reg `0x19` (`0xA0000064`), Bit Field: [21], Default: `1'h1`, W/R (PREv18(A/E))

If 1, it masks the actual XO Driver output, and CLK_XO becomes 0 regardless of the actual XO Driver state.

If 0, the XO Driver output becomes directly exposed to (and connected to) CLK_XO.

XO_EN_DIV Reg `0x19` (`0xA0000064`), Bit Field: [20], Default: `1'h0`, W/R (PREv18(A/E))

FIXME

XO_S Reg `0x19` (`0xA0000064`), Bit Field: [19:17], Default: `3'h0`, W/R (PREv18(A/E))

FIXME

XO_SEL_CP_DIV Reg `0x19` (`0xA0000064`), Bit Field: [16], Default: `1'h0`, W/R (PREv18(A/E))

FIXME

XO_EN_OUT Reg `0x19` (`0xA0000064`), Bit Field: [15], Default: `1'h0`, W/R (PREv18(A/E))

FIXME

XO_PULSE_SEL Reg `0x19` (`0xA0000064`), Bit Field: [14:11], Default: `4'h4`, W/R (PREv18(A/E))

Pulse selection. This must be used in conjunction with `XO_DELAY_EN`. It is one-hot encoding. See Table 49 for valid values.

See Figure 16 for circuit implementation. Also see Section 19.6.1 and Section 19.4.

XO_DELAY_EN Reg `0x19` (`0xA0000064`), Bit Field: [10:8], Default: `3'h7`, W/R (PREv18(A/E))

Delay enable for pulse width lengthening. This must be used in conjunction with `XO_PULSE_SEL`.

See Table 49 for valid values.

See Figure 16 for circuit implementation. Also see Section 19.6.1 and Section 19.4.

XO_DRV_START_UP Reg `0x19` (`0xA0000064`), Bit Field: [7], Default: `1'h0`, W/R (PREv18(A/E))

If 1, it uses startup circuit to drive crystal. Mutually exclusive with `XO_DRV_CORE` signal.

Also see Section 19.4.

XO_DRV_CORE Reg `0x19` (`0xA0000064`), Bit Field: [6], Default: `1'h0`, W/R (PREv18(A/E))

If 1, it uses core circuit to drive crystal. Mutually exclusive with `XO_DRV_START_UP` signal.

Also see Section 19.4.

XO_RP_LOW Reg `0x19` (`0xA0000064`), Bit Field: [5], Default: `1'h0`, W/R (PREv18(A/E))

See Section 19.6.2.

XO_RP_MEDIA Reg `0x19` (`0xA0000064`), Bit Field: [4], Default: `1'h0`, W/R (PREv18(A/E))

See Section 19.6.2.

XO_RP_MVT Reg `0x19` (`0xA0000064`), Bit Field: [3], Default: `1'h0`, W/R (PREv18(A/E))

See Section 19.6.2.

XO_RP_SVT Reg `0x19` (`0xA0000064`), Bit Field: [2], Default: `1'h0`, W/R (PREv18(A/E))

See Section 19.6.2.

XO_SCN_CLK_SEL Reg `0x19` (`0xA0000064`), Bit Field: [1], Default: `1'h0`, W/R (PREv18(A/E))

Selects clock source for SCN.

Set to 0 when SCN output has not yet been fully established to use 0.6V crystal signal.

Set to 1 during normal operation. Crystal's 0.3V oscillation signal is level-converted to 0.6V for SCN clock.

Also see Section 19.4.

XO_SCN_ENB Reg `0x19` (`0xA0000064`), Bit Field: [0], Default: `1'h1`, W/R (PREv18(A/E))

If 0, it enables SCN. After crystal begins to oscillate with start-up circuit, SCN needs to be started to establish internal voltage for the core circuit to operate.

If 1, it disables SCN.

Also see Section 19.4.

XO_CAP_TUNE Reg `0x1A` (`0xA0000068`), Bit Field: [11:0], Default: `12'h000`, W/R (PREv18(A/E))

Adds additional capacitance on OSC_IN and OSC_DRV. See Section 19.6.3

19.4 Operation

1. Upon startup, it is assumed that the registers have the values shown in Table 44.

Table 44: XO Driver Startup Register Values

Register	Value
`XO_SLEEP`	`1'h1`
`XO_ISOLATE`	`1'h1`
`XO_PULSE_SEL`	Desired Value
`XO_DELAY_EN`	Desired Value
`XO_DRV_START_UP`	`1'h0`
`XO_DRV_CORE`	`1'h0`
`XO_SCN_CLK_SEL`	`1'h0`
`XO_SCN_ENB`	`1'h1`

2. Set `XO_SLEEP`=0. This will power up XO Driver. Wait for *minimum* of 1ms.
3. Set `XO_ISOLATE`=0. This will un-isolate XO Driver.

4. Set XO_DRV_START_UP to 1'h1. Wait for *minimum* of 1s. (Starts the crystal oscillation at 0.6V)
5. Set XO_SCN_CLK_SEL to 1'h1. Wait for *minimum* of 300us.
6. Set registers with values shown in Table 45. Wait for *minimum* of 1s.

Table 45: XO Driver Operation#6 Register Values

Register	Value
XO_SCN_CLK_SEL	1'h0
XO_SCN_ENB	1'h0

7. Set the following registers simultaneously as shown in Table 46.The 32.768kHz clock (CLK_XO) should be generated and stable at this point.

Table 46: XO Driver Operation#7 Register Values

Register	Value
XO_DRV_START_UP	1'h0
XO_DRV_CORE	1'h1
XO_SCN_CLK_SEL	1'h1

8. In order to turn off XO Driver, set the following registers simultaneously as shown in Table 47.

Table 47: XO Driver Operation#8 Register Values

Register	Value
XO_DRV_CORE	1'h0
XO_SCN_ENB	1'h1

9. XO Driver can be further powered-off to save sleep power consumption. Set registers as shown in Table 48.

Table 48: XO Driver Operation#9 Register Values

Register	Value
XO_SLEEP	1'h1
XO_ISOLATE	1'h1

19.5 XO_DELAY_EN and XO_PULSE_SEL Detailed Description

See Figure 16.

19.6 Tuning

19.6.1 Pulse Length

Depending on the desired pulse length, XO_PULSE_SEL and XO_DELAY_EN must be properly set before starting the operation. Not all of the bit combinations are available. See Table 49 for possible combinations and use one of them.

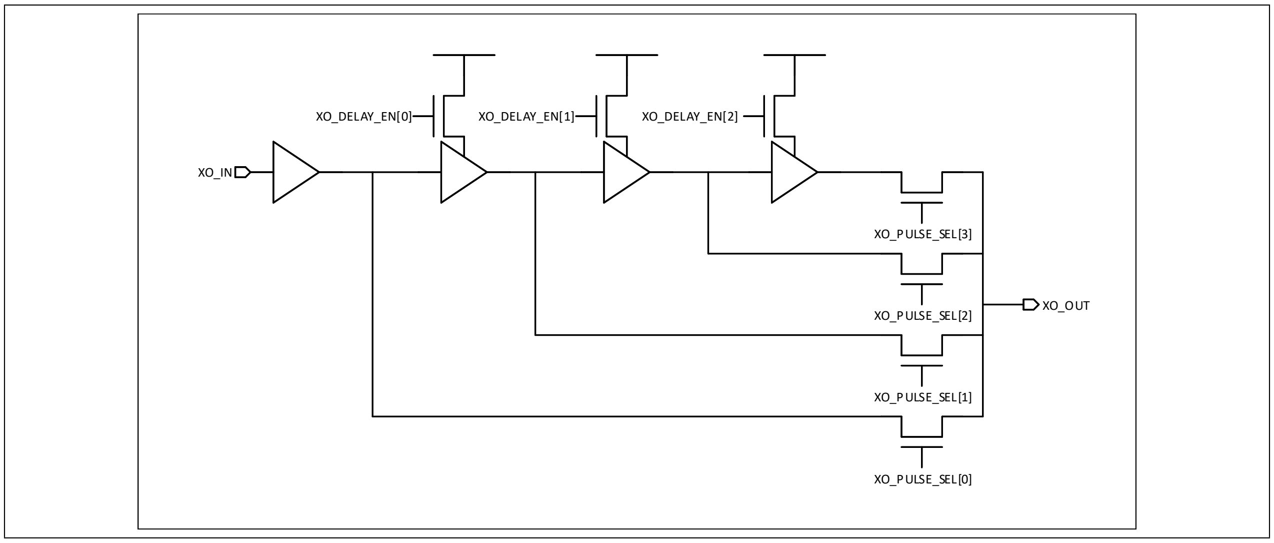

Figure 16: XO_DELAY_EN and XO_PULSE_SEL Detailed Diagram

Table 49: XO_PULSE_SEL and XO_DELAY_EN Setting

Pulse Length	XO_PULSE_SEL	XO_DELAY_EN
Shortest Pulse	4'h1	3'h0
Shorter Pulse	4'h2	3'h1
Longer Pulse	4'h4	3'h3
Longest Pulse	4'h8	3'h7

19.6.2 DC Biasing

XO Driver provides four (4) pseudo-resistors to fine-tune DC bias of the crystal. The following registers in MBus Register File enables/disables each pseudo-resistor.

- XO_RP_LOW: If 1, it turns on the lowest-Vth pseudo-resistor to DC bias the crystal. Recommended to be on when process variation is on slow side.
- XO_RP_MEDIA: If 1, it turns on the normal-Vth pseudo-resistor to DC bias the crystal. Recommended to be on when process variation is at TT corner.
- XO_RP_MVT: If 1, it turns on the higher-Vth pseudo-resistor to DC bias the crystal. Recommended to be on when process variation is on fast side.
- XO_RP_SVT: If 1, it turns on the highest-Vth pseudo-resistor to DC bias the crystal. Recommended to be on when process variation is on the extremely fast side.

19.6.3 Additional Capacitance

PRE provides two pads to connect a crystal: OSC_IN and OSC_DRV. Parasitic capacitances on these two paths should be closely matched to get the best accuracy/perforamance. However, wire loading on these two nodes, including wire-bonding and PCB traces, may be different depending on assembly/packaging. XO_CAP_TUNE provides a way to compensate this mismatch.

XO_CAP_TUNE[11:6] enables/disables additional capacitance on OSC_IN pad. The amount of capacitance added to OSC_IN is "1.8pF $\times$ Number of 1's in XO_CAP_TUNE[11:6]"

XO_CAP_TUNE[5:0] enables/disables additional capacitance on OSC_DRV pad. The amount of capacitance added to OSC_DRV is "1.8pF $\times$ Number of 1's in XO_CAP_TUNE[5:0]"

From testing of the previous PRE version (PREv12), a recommended value is `12’h07C0`, which adds 9pF to OSC_IN.

20 SPI [PREv18(A/E) Only]

This is an implementation of ARM PrimeCell® Synchronous Serial Port (PL022). Please see Technical Reference Manual for complete details.

20.1 Power and Clock

- SPI is operating at VDD_0P6_LC, which is a power-gated 0.6V, shared with Layer Ctrl.
- SPI uses the same clock as Layer Ctrl (clk_lc).
- SPI are reset by the Layer Ctrl Reset signal (lc_reset_b).
- SPI pads (SPIRX, SPISEL, SPICLK, SPITX) I/O voltage is at VDD_COTS.

Although SPI shares the power domain and reset signal with the Layer Ctrl, there is a special register that ”freezes” the SPI output before the system goes into sleep. See `SPI_FREEZE_OUTPUT`.

20.2 Memory-Mapped Addresses

Table 50 shows memory mapped address for SPI. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

GPIO_* signals are used for GPIO operations. See Section 21.

Table 50: Memory Map for SPI

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
0xA0000060	0x18	SPI_FREEZE_OUTPUT	W/R	1'h0	[23]
		SPI_PAD_ENABLE_INPUT	W/R	1'h0	[21]
		SPI_PAD_ENABLE_OUTPUT	W/R	1'h0	[20]
		GPIO_FREEZE_OUTPUT	W/R	1'h0	[16]
		GPIO_WIRQ_POSEDGE_ENABLE	W/R	4'h0	[15:12]
		GPIO_WIRQ_NEGEDGE_ENABLE	W/R	4'h0	[11:8]
		GPIO_PAD_ENABLE	W/R	8'h00	[7:0]
0xA0004000	-	SSPCR0 (Control Register 0)	W/R	16'h0000	-
0xA0004004	-	SSPCR1 (Control Register 1)	W/R	4'h0	-
0xA0004008	-	SSPDR (Data Register)	W/R	16'h0000	-
0xA000400C	-	SSPSR (Status Register)	W/R	5'h03	-
0xA0004010	-	SSPCPSR (Clock Prescale Register)	W/R	8'h00	-
0xA0004014	-	SSPIMSC (Interrupt Mask Set or Clear Register)	W/R	4'h0	-
0xA0004018	-	SSPRIS (Raw Interrupt Status Register)	W/R	4'h8	-
0xA000401C	-	SSPMIS (Masked Interrupt Status Register)	W/R	4'h0	-
0xA0004020	-	SSPICR (Interrupt Clear Register)	W/R	4'h0	-
0xA0004024	-	SSPDMACR (DMA Control Register)	W/R	2'h0	-

20.3 Retentive SPI Registers

20.3.1 0xA00000XX

SPI_FREEZE_OUTPUT Reg 0x18 (0xA0000060), Bit Field: [23], Default: 1'h0, W/R (PREv18(A/E))

User can set this bit to 1 to keep the current SPI output (SPISEL, SPICLK, SPITX), so that outputs remain frozen even in sleep mode, until the user resets `SPI_FREEZE_OUTPUT` to 0. This is useful when the system goes in and out of the sleep mode but the user wants to keep the SPI outputs same.

If the system goes into sleep with `SPI_FREEZE_OUTPUT`=0, then the SPI output (SPISEL, SPICLK, SPITX) may become floating or unknown value during the sleep mode.

Note that, when the system wakes up, the internal SPI signals become reset to their default values. If user writes 0 to `SPI_FREEZE_OUTPUT` after the wakeup, the SPI output may change to their default values as well. In order to prevent this, user may want to store the previous SPI output in another location (such as in the retentive SRAM), and overwrite the SPI output using these stored values before changing `SPI_FREEZE_OUTPUT` to 0 after the wakeup.

SPI_PAD_ENABLE_INPUT Reg `0x18` (`0xA0000060`), Bit Field: [21], Default: `1'h0`, W/R (PREv18(A/E))

If 1, the SPI input pad (SPIRX) is enabled.

If 0, the SPI input pad (SPIRX) is disabled.

SPI_PAD_ENABLE_OUTPUT Reg `0x18` (`0xA0000060`), Bit Field: [20], Default: `1'h0`, W/R (PREv18(A/E))

If 1, the SPI output pads (SPISEL, SPICLK, SPITX) are enabled.

If 0, the SPI output pads (SPISEL, SPICLK, SPITX) are disabled.

20.4 Non-Retentive SPI Configuration Registers

20.4.1 `0xA0004XXX`

These registers are non-retentive, and a part of ARM PrimeCell® Synchronous Serial Port (PL022). Please see Technical Reference Manual for complete details.

SSPCR0 (`0xA0004000`), Default: `16'h0000`, W/R

Control Register 0. For detailed information, see *ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual*.

SSPCR1 (`0xA0004004`), Default: `4'h0` , W/R

Control Register 1. For detailed information, see *ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual*.

SSPDR (`0xA0004008`), Default: `16'h0000`, W/R

Data Register. For detailed information, see *ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual*.

SSPSR (`0xA000400C`), Default: `5'h03` , W/R

Status Register. For detailed information, see *ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual*.

SSPCPSR (`0xA0004010`), Default: `8'h00` , W/R

Clock Prescale Register. For detailed information, see *ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual*.

SSPIMSC (`0xA0004014`), Default: `4'h0` , W/R

Interrupt Mask Set or Clear Register. For detailed information, see *ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual*.

SSPRIS (`0xA0004018`), Default: `4'h8` , W/R

Raw Interrupt Status Register. For detailed information, see *ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual*.

SSPMIS (`0xA000401C`), Default: `4'h0` , W/R

Masked Interrupt Status Register. For detailed information, see *ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual*.

SSPICR (`0xA0004020`), Default: `4'h0` , W/R

Interrupt Clear Register. For detailed information, see *ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual*.

SSPDMACR (`0xA0004024`), Default: `2'h0` , W/R

DMA Control Register. For detailed information, see *ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual*.

20.5 Operation

See *ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual* for complete details.

Before using SPI, user has to make sure that `SPI_FREEZE_OUTPUT`=0, `SPI_PAD_ENABLE_INPUT`=1, `SPI_PAD_ENABLE_OUTPUT`=1. Otherwise, SPI does not work properly.

Before the system goes into sleep, user may need to make sure that `SPI_FREEZE_OUTPUT`=1 in order to keep the current SPI outputs (SPISEL, SPICLK, SPITX). Otherwise, the SPI outputs may become floating or unknown value when the system goes into sleep.

Note that, when the system wakes up, the internal SPI signals become reset to their default values. If user writes 0 to `SPI_FREEZE_OUTPUT` after the wakeup, the SPI output may change to their default values as well. In order to prevent this, user may want to store the previous SPI output in another location (such as in the retentive SRAM), and overwrite the SPI output using these stored values before changing `SPI_FREEZE_OUTPUT` to 0 after the wakeup.

20.6 Interrupt

SPI can be configured to generate a Cortex-M0 IRQ shown in Table 51. The interrupt can be further masked by Cortex-M0 NVIC registers.

See *ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual* for how to configure and enable the IRQ.

Table 51: SPI-related Cortex-M0 IRQ

Interrupt Port	Description
IRQ[18]	[PREv18(A/E) Only] Interrupt from SPI

21 GPIO [PREv18(A/E) Only]

This is an implementation of ARM PrimeCell® General Purpose Input/Output (PL061). Please see Technical Reference Manual for complete details.

21.1 Power and Clock

- GPIO is operating at VDD_0P6_LC, which is a power-gated 0.6V, shared with Layer Ctrl.
- GPIO uses the same clock as Layer Ctrl (clk_lc).
- GPIO are reset by the Layer Ctrl Reset signal (lc_reset_b).
- GPIO pads (GPIO[7:0]) I/O voltage is at VDD_COTS.

Although GPIO shares the power domain and reset signal with the Layer Ctrl, there is a special register that "freezes" the GPIO output before the system goes into sleep. See `GPIO_FREEZE_OUTPUT`.

21.2 Memory-Mapped Addresses

Table 52 shows memory mapped address for GPIO. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

SPI_* signals are used for GPIO operations. See Section 20.

Table 52: Memory Map for GPIO

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
0xA0000060	0x18	SPI_FREEZE_OUTPUT	W/R	1'h0	[23]
		SPI_PAD_ENABLE_INPUT	W/R	1'h0	[21]
		SPI_PAD_ENABLE_OUTPUT	W/R	1'h0	[20]
		GPIO_FREEZE_OUTPUT	W/R	1'h0	[16]
		GPIO_WIRQ_POSEDGE_ENABLE	W/R	4'h0	[15:12]
		GPIO_WIRQ_NEGEDGE_ENABLE	W/R	4'h0	[11:8]
		GPIO_PAD_ENABLE	W/R	8'h00	[7:0]
0xA0005000	-	GPIODATA	W/R	8'h00	[7:0]
0xA0005400	-	GPIODIR	W/R	8'h00	[7:0]
0xA0005410	-	GPIOIE	W/R	8'h00	[7:0]

21.3 Retentive GPIO Registers

21.3.1 0xA00000XX

GPIO_FREEZE_OUTPUT Reg `0x18` (`0xA0000060`), Bit Field: [16], Default: `1'h0`, W/R (PREv18(A/E))

User can set this bit to 1 to keep the current GPIO output (GPIOSEL, GPIOCLK, GPIOTX), so that outputs remain frozen even in sleep mode, until the user resets `GPIO_FREEZE_OUTPUT` to 0. This is useful when the system goes in and out of the sleep mode but the user wants to keep the GPIO outputs same.

If the system goes into sleep with `GPIO_FREEZE_OUTPUT`=0, then the GPIO output (GPIOSEL, GPIOCLK, GPIOTX) may become floating or unknown value during the sleep mode.

Note that, when the system wakes up, the internal GPIO signals become reset to their default values. If user writes 0 to `GPIO_FREEZE_OUTPUT` after the wakeup, the GPIO output may change to their default values as well. In order to prevent this, user may want to store the previous GPIO output in another location (such as in the retentive SRAM), and overwrite the GPIO output using these stored values before changing `GPIO_FREEZE_OUTPUT` to 0 after the wakeup.

GPIO_WIRQ_POSEDGE_ENABLE Reg `0x18` (`0xA0000060`), Bit Field: [15:12], Default: `4'h0`, W/R (PREv18(A/E))

Each bit in `GPIO_WIRQ_POSEDGE_ENABLE`[3:0] corresponds to the each bit in GPIO[3:0]. If `GPIO_WIRQ_POSEDGE_ENABLE`[n] is set to 1, then it generates an wakeup request to wake up the whole system when GPIO[n] sees a positive edge.

GPIO_WIRQ_NEGEDGE_ENABLE Reg `0x18` (`0xA0000060`), Bit Field: [11:8], Default: `4'h0`, W/R (PREv18(A/E))

Each bit in `GPIO_WIRQ_NEGEDGE_ENABLE`[3:0] corresponds to the each bit in GPIO[3:0]. If `GPIO_WIRQ_NEGEDGE_ENABLE`[n] is set to 1, then it generates an wakeup request to wake up the whole system when GPIO[n] sees a negative edge.

GPIO_PAD_ENABLE Reg `0x18` (`0xA0000060`), Bit Field: [7:0], Default: `8'h00`, W/R (PREv18(A/E))

If 1, the GPIO pads (GPIO[7:0]) is enabled.

If 0, the GPIO pads (GPIO[7:0]) is disabled.

21.4 Non-Retentive GPIO Configuration Registers

These registers are non-retentive, and a part of ARM PrimeCell® General Purpose Input/Output (PL061). Please see Technical Reference Manual for complete details.

21.4.1 `0xA0005XXX`

GPIODATA (`0xA0005000`), Default: `8'h00`, W/R

Data to be set or read depending on `GPIODIR` register. For detailed information, see *ARM PrimeCell® General Purpose Input/Output (PL061) Technical Reference Manual*.

GPIODIR (`0xA0005400`), Default: `8'h00` , W/R

Controls direction of GPIO.

If 0, the corresponding bit is set to output.

If 1, the corresponding bit is set to input.

For detailed information, see *ARM PrimeCell® General Purpose Input/Output (PL061) Technical Reference Manual.*

GPIOIE (`0xA0005410`), Default: `8'h00`, W/R

GPIO Interrupt Enable.

If 0, the interrupt from the corresponding bit is disabled

If 1, the interrupt from the corresponding bit is enabled.

21.5 Operation

See *ARM PrimeCell® General Purpose Input/Output (PL061) Technical Reference Manual* for complete details.

Before using GPIO, user has to make sure that `GPIO_FREEZE_OUTPUT`=0, `GPIO_PAD_ENABLE`=1. Otherwise, GPIO does not work properly.

Before the system goes into sleep, user may need to make sure that `GPIO_FREEZE_OUTPUT`=1 in order to keep the current GPIO output, if any. Otherwise, the GPIO outputs may become floating or unknown value when the system goes into sleep.

Note that, when the system wakes up, the internal GPIO signals become reset to their default values. If user writes 0 to `GPIO_FREEZE_OUTPUT` after the wakeup, the GPIO output may change to their default values as well. In order to prevent this, user may want to store the previous GPIO output in another location (such as in the retentive SRAM), and overwrite the GPIO output using these stored values before changing `GPIO_FREEZE_OUTPUT` to 0 after the wakeup.

21.6 Interrupt

GPIO can be configured to generate a Cortex-M0 IRQ shown in Table 53, if enabled by `GPIOIE`, when one or more of the corresponding bits in GPIODATA is written. The interrupt can be further masked by Cortex-M0 NVIC registers.

See *ARM PrimeCell® General Purpose Input/Output (PL061) Technical Reference Manual* for how to configure and enable the IRQ.

Table 53: GPIO-related Cortex-M0 IRQ

Interrupt Port	Description
IRQ[17]	[PREv18(A/E) Only] Interrupt from GPIO

21.7 Wakeup Request

GPIO can be configured to generate a wakeup request so that the system wakes up when it sees a change in GPIO pads. The behavior may be affected by `WAKEUP_ON_PEND_REQ` setting, as well.

21.7.1 Wakeup using Positive/Negative Edges

Wakeup at Positive Edge Each bit in `GPIO_WIRQ_POSEDGE_ENABLE`[3:0] corresponds to the each bit in GPIO[3:0]. If `GPIO_WIRQ_POSEDGE_ENABLE`[n] is set to 1, then it generates an wakeup request to wake up the whole system when GPIO[n] sees a positive edge.

Wakeup at Negative Edge Each bit in `GPIO_WIRQ_NEGEDGE_ENABLE`[3:0] corresponds to the each bit in GPIO[3:0]. If `GPIO_WIRQ_NEGEDGE_ENABLE`[n] is set to 1, then it generates an wakeup request to wake up the whole system when GPIO[n] sees a negative edge.

22 COTS Power Switches [PREv18(A/E) Only]

22.1 Power and Clock

COTS (Custom-Off-the-Shelf) Power Switches are controlled through MBus Register File. Thus, it can be regarded as retentive. The switch transistors are controlled at VDD_COTS.

PREv18(A/E) have three (3) COTS Power Switches.

22.2 Memory-Mapped Addresses

Table 54 shows memory mapped address for COTS Power Switches. Signals with MBus Reg ID are retentive, meaning that they retain their values even in sleep mode. Signals without MBus Reg ID reset to their default values upon system wake-up.

Table 54: Memory Map for COTS Power Switches

MMIO ADDR	MBus Reg ID	Register Name	W/R	Default	Bit Field
`0xA000005C`	`0x17`	`CPS_ON`	W/R	`3'h0`	[2:0]

22.2.1 Retentive COTS Power Switches Registers

CPS_ON Reg `0x17` (`0xA000005C`), Bit Field: [2:0], Default: `3'h0`, W/R (PREv18(A/E))

Turns on and off the COTS Power Switches.

- CPS_ON[2]: If 1, VCOTS_IN2 and VCOTS_IN2 are connected. If 0, they are disconnected.
- CPS_ON[1]: If 1, VCOTS_IN1 and VCOTS_IN1 are connected. If 0, they are disconnected.
- CPS_ON[0]: If 1, VCOTS_IN0 and VCOTS_IN0 are connected. If 0, they are disconnected.