nuvoTon

32-BIT ARM926EJ-S BASED MCU

# NUC950ADN
# 32-bit ARM926EJ-S Based Microcontroller
# Product Data Sheet





*Publication Release Date: Jun. 18, 2010*
*Revision: A4*

nuvoTon

32-BIT ARM926EJ-S BASED MCU

# Table of Contents

# 32-BIT ARM926EJ-S BASED MCU

## 1. GENERAL DESCRIPTION

This chip is built around an outstanding CPU core: the 16/32 ARM926EJ-S RISC processor designed by Advanced RISC Machines, Ltd. The ARM926EJ-S core, offers 8K-byte I-cache and 8K-byte D-cache with MMU, is a low power, general-purpose integrated circuits. One 10/100 Mb MAC of Ethernet controller is built-in to reduce total system cost. A TFT type LCD controller and 2D graphics engine with various integrated on chip functions, this micro-controller is suitable for a high end, high performance and low cost related products as well as general purpose applications.

The following integrated on-chip functions are described in detail in this document.

| | Main Function |
|---|---|
| CPU | • ARM926EJ-S |
| Platform | • Programmable PLL System Clock Synthesizer |
| | • AMBA Peripherals |
| | • Timer, Watchdog Timer |
| | • Advanced Interrupt Controller |
| | • General DMA Controller |
| | • External Bus Interface Controller |
| Networking | • Ethernet MAC Controller |
| Display Interface | • LCD Controller |
| Graphics | • 2D Graphic Engine |
| Audio Interface | • 2-Channel I2S Controller |
| | • 2-Channel AC97 Controller |
| USB Interface | • USB 1.1/2.0 High/Full/Low Speed Host Controller |
| | • USB 2.0 High/Full Speed Device Controller |
| Storage Interface | • NAND Flash Controller with ECC1/ECC4 |
| | • SD/SDIO/MMC Controller |
| | • Memory Stick (MS) Controller |
| Peripheral & Misc. | • GPIO |
| | • 4-Channel PWM |
| | • UART/HS-UART |
| | • USI (SPI/uWire) |

| | Main Function |
|---|---|
| | • I2C (Master) Controller |
| | • Keypad Scan Controller |

## 2. FEATURES

### Architecture

- Efficient and powerful ARM926EJ-S core with fully 16/32-bit RISC architecture
- Little Endian mode supported
- 8K-byte I-cache and 8K-byte D-cache with MMU
- Cost-effective JTAG-based debug solution

### Power management

- Programmable clock enable for individual peripherals
- IDLE mode to halt ARM Core and keep peripheral working
- Power-Down mode to stop all clocks included external crystal oscillator.
- Exit IDLE/Power-Down by interrupts

### PLL

- Supports two on-chip PLLs
- The external clock can be multiplied by on-chip PLL to provide high frequency system clock
- The input frequency range is 4-30MHz; 15MHz is preferred.
- Programmable clock frequency
- Wakeup by interrupt, USB device.

### Advanced Interrupt Controller

- 31 interrupt sources, including 3 external interrupt sources
- Programmable normal or fast interrupt mode (IRQ, FIQ)
- Programmable as either edge-triggered or level-sensitive for 3 external interrupt sources
- Programmable as either low-active or high-active for 3 external interrupt sources
- Priority methodology is encoded to allow for interrupt daisy-chaining
- Automatically mask out the lower priority interrupt during interrupt nesting
- Automatically clear the interrupt flag when the interrupt source is programmed to be edge-triggered

### General DMA Controller

- 2-channel General DMA for memory-to-memory data transfers without CPU intervention

- Increments or decrements a source or destination address in 8-bit, 16-bit or 32-bit data transfers
- 8-data burst mode

### External Bus Interface

- 8/16/32-bit external bus support for ROM/SRAM, flash memory, SDRAM and external I/Os
- Programmable access cycle (0-7 wait cycle)
- Four-word depth write buffer

### Ethernet MAC Controller

- 100/10-Mbps operation
- DMA engine with burst mode
- MAC Tx/Rx buffers (256 bytes Tx, 256 bytes Rx)
- Full compliance with IEEE standard 802.3
- RMII interface only
- Station Management Signaling
- On-Chip CAM (up to 16 destination addresses)
- Full-duplex mode with PAUSE feature
- Long/short packet modes

### LCD Controller

- Support the 8/12/16/18-bit data interface to connect with 80/68 series MPU type LCM module
- Convert RGB-565, YUV-422 display data to RGB-444, RGB-565, RGB-666, YUV-422 color format for display output
- Support CCIR-656( with vsync / hsync / data enable sync signal ) 8/16-bit YUV data output format to connect with external TV encoder
- Support 8/16 bpp OSD data with image overlay function to facilitate the diverse graphic UI.
- Support linear 1X – 8X image scaling up function.
- Support Picture-In-Picture display function
- Support hardware cursor.

### 2-D Graphics Engine

- Color depth 8-bit/16-bit/32-bit in RGB domain or RGB332/RGB565/RGB888 are supported
- Contains 2D Bit Block Transfer (BitBLT) functions as defined in Microsoft GDI. It includes HostBLT, Pattern BLT, Color/Font Expanding BLT, Transparent BLT, Tile BLT, Block Move BLT, Copy File BLT, Color/Font Expansion, and Rectangle Fill, etc.

- Supports fore/background colors and all Microsoft 256 ternary raster-operation codes (ROP)
- Clipping window can be defined as inside or outside clipping
- Implements the alpha-blending function for source/destination picture overlaying
- Fast Bresenham line drawing algorithm is used to draw solid/textured lines
- Supports rectangular border or frame drawing
- Supports picture re-sizing by 1/255 ~ 254/255 down-scaling and 1 ~ 1.996 up-scaling (1+254/255).
- Supports object rotations in different degrees, that is L45/L90/R45/R90/M180/F180/X180, where
  - L45/L90 means rotate left 45/90 degrees,
  - R45/R90 means rotate right 45/90 degrees,
  - M180 means mirror (flop),
  - F180 means up-side-down (flip) and X180 for rotations by 180 degrees

### 2-Channel AC97 / I2S Controller

- Support I2S interface.
- Support AC97 interface.
- Built-in an 8x32 bits internal buffer.
- Support DMA function for data transfer between internal buffer and system memory.
- Support 16-bit I2S and MSB-justified format.

### USB Host Controller with transceiver

- Fully compliant with USB Revision 2.0 specification.
- Enhanced Host Controller Interface (EHCI) Revision 1.0 compatible.
- Open Host Controller Interface (OHCI) Revision 1.0 compatible.
- Supports high-speed (480Mbps), full-speed (12Mbps) and low-speed (1.5Mbps) USB devices.
- Supports Control, Bulk, Interrupt, Isochronous and Split transfers.
- Built-in DMA for real-time data transfer.
- Support two ports (one port transceiver is shared with USB Device Controller)

### USB Device Controller with transceiver

- Compliant with USB version 2.0 specification.
- Software control for device remote-wakeup.
- Supports 6 configurable IN/OUT endpoints in addition to Control Endpoint. Each of these endpoints can be configures as IN or OUT with Isochronous, Bulk or Interrupt transfer.
- Three different modes of operation of an in-endpoint (Auto validation mode, manual validation mode, Fly mode.

- Supports Endpoint Maximum Packet Size up to 1024 bytes.

### Flash Memory Interface (FMI)

- Directly connect to Secure Digital (SD, MMC and SDIO) flash memory card, Memory Stick (Memory stick PRO) and NAND type flash memory.
- Supports DMA function to accelerate the data transfer between the internal buffer, external SDRAM, and flash memory card.
- Two 512 bytes internal buffers are embedded inside

### I2C Master

- Compatible with $I^2C$ standard, support master mode only
- Multi Master Operation
- Clock stretching and wait state generation
- Provide multi-byte transmit operation, up to 4 bytes can be transmitted in a single transfer
- Software programmable acknowledge bit
- Arbitration lost interrupt, with automatic transfer cancellation
- Start/Stop/Repeated Start/Acknowledge generation
- Start/Stop/Repeated Start detection
- Bus busy detection
- Supports 7 bit addressing mode
- Software mode $I^2C$

### Universal Serial Interface (USI)

- Support MICROWIRE/SPI master mode
- Support full/half duplex synchronous serial data transfer
- Variable length of transfer word up to 32 bits
- Provide burst mode operation, transmit/receive can be executed up to four times in one transfer
- MSB or LSB first data transfer
- Receive and Transmit on both rising or falling edge of serial clock independently

### UART

- Three UART (serial I/O) blocks with interrupt-based operation
- Support for 5-bit, 6-bit, 7-bit or 8-bit serial data transmit and receive
- Programmable baud rates
- 1,1½ or 2 stop bits

- Odd or even parity
- Break generation and detection
- Parity, overrun and framing error detection
- X16 clock mode
- Support for IrDA and two debug ports

### Timers

- Five programmable 24-bit timers with 8-bit pre-scalar
- One programmable 20-bit Watchdog timer
- One-short mode, period mode or toggle mode operation

### 4-Channel PWM

- Four 16-bit timers
- Two 8-bit pre-scalars & Two 4-bit divider
- Programmable duty control of output waveform (PWM)
- Auto reload mode or one-shot pulse mode
- Dead-zone generator

### Keypad Scan Interface

- Scan up to 16x8 with an external 4 to 16 decoder; or 4x8 array without auxiliary component
- Programmable de-bounce time
- One or two keys scan with interrupt and three keys reset function.
- Support low power wakeup function

### Programmable I/Os

- Pins individually configurable to input, output or I/O mode for dedicated signals
- I/O ports are Programmable and Configurable for Multiple functions

### Operation Voltage Range

- VDD18 for IO Buffer: 1.8V+/-10%
- VDD33 for Core Logic: 3.3V+/-10%
- USBVDDC0/USBVDDC1/USBVDDT0/USBVDDT1 for USB: 3.3V+/-5%
- PLLVDD18 for PLL: 1.8V+/-10%

# 32-BIT ARM926EJ-S BASED MCU

### Operation Temperature Range

- -40℃~+85℃

### Operating Frequency

- Up to 200 MHz for ARM926EJ-S CPU

### Package Type

- 216-Pin LQFP, Pb free

## 3. PIN DIAGRAM

### NUC950ADN Pin Diagram

# 4. PIN ASSIGNMENT

**Table 4.1 NUC950ADN Pins Assignment**

| Pad Name | NUC950ADN |
|---|---|
| **Clock & Reset** | **( 5 pins )** |
| **EXTAL15M** | 102 |
| **XTAL15M** | 103 |
| **EXTAL48M0** | 209 |
| **XTAL48M0** | 208 |
| **nRESET** | 1 |
| **TAP Interface** | **( 5 pins )** |
| **TMS** | 10 |
| **TDI** | 11 |
| **TDO** | 12 |
| **TCK** | 13 |
| **nTRST** | 14 |
| **External Bus Interface** | **( 71 pins )** |
| **MA [21:0]** | 60-48,46-38 |
| **MD [31:0]** | 100-98,95-88,86,84-78,75-63 |
| **nWBE [3:0] / SDQM [3:0]** | 32-29 |
| **nSCS [1:0]** | 23-22 |
| **nSRAS** | 35 |
| **nSCAS** | 36 |
| **MCKE** | 33 |
| **nSWE** | 34 |
| **MCLK** | 25 |
| **nWAIT** | 16 |
| **nBTCS** | 21 |
| **nECS [2:0]** | 20-18 |
| **nOE** | 17 |

| Pad Name | NUC950ADN |
|---|---|
| **Ethernet Interface** | **( 10 pins )** |
| PHY_MDC / GPIOF[0] | 172 |
| PHY_MDIO / GPIOF[1] | 173 |
| PHY_TXD [1:0] / GPIOF[3:2] | 175-174 |
| PHY_TXEN / GPIOF[4] | 176 |
| PHY_REFCLK / GPIOF[5] | 177 |
| PHY_RXD [1:0] / GPIOF[7:6] | 179-178 |
| PHY_CRSDV / GPIOF[8] | 180 |
| PHY_RXERR / GPIOF[9] | 181 |
| **AC97/I2S/PWM** | **( 5 pins )** |
| AC97_nRESET / I2S_SYSCLK / - / GPIOG[12] | 184 |
| AC97_DATAI / I2S_DATAI / PWM [0] / GPIOG[13] | 185 |
| AC97_DATAO / I2S_DATAO / PWM [1] / GPIOG[14] | 186 |
| AC97_SYNC / I2S_WS / PWM [2] / GPIOG[15] | 187 |
| AC97_BITCLK / I2S_BITCLK / PWM [3] / GPIOG[16] | 188 |

| USB Interface | ( 10 pins ) |
|---|---|
| DP0 | 215 |
| DN0 | 214 |
| REXT0 | 212 |
| UPWR0 | 195 |
| OVI | 196 |
| HDS | 198 |
| DP1 | 206 |
| DN1 | 205 |
| REXT1 | 203 |
| UPWR1 | 200 |
| **Pad Name** | **NUC950ADN** |
| I2C/USI(SPI/MW) | ( 4 pins ) |
| SCL0 / SFRM / GPIOG[0] | 3 |
| SDA0 / SSPTXD / GPIOG[1] | 4 |
| SCL1 / SCLK / GPIOG[2] | 5 |
| SDA1 / SSPRXD / GPIOG[3] | 6 |
| **Pad Name** | **NUC950ADN** |
| UART | ( 6 pins ) |
| TXD0 / GPIOE[0] | 189 |
| RXD0 / GPIOE[1] | 190 |
| TXD1(B) / GPIOE[2] | 191 |
| RXD1(B) / GPIOE[3] | 192 |
| TXD2(IrDA) / GPIOE[6] | 193 |
| RXD2(IrDA) / GPIOE[7] | 194 |

| Pad Name | NUC950ADN |
|---|---|
| **SDIO(SD) / Memory Stick** | **( 8 pins )** |
| **SD_CMD / MS_BS / GPIOD[0]** | 156 |
| **SD_CLK / MS_CLK / GPIOD[1]** | 157 |
| **SD_DAT0 / MS_DAT0 / GPIOD[2]** | 158 |
| **SD_DAT1 / MS_DAT1 / GPIOD[3]** | 159 |
| **SD_DAT2 / MS_DAT2 / GPIOD[4]** | 160 |
| **SD_DAT3 / MS_DAT3 / GPIOD[5]** | 161 |
| **SD_CDn / MS_CDn / GPIOD[6]** | 163 |
| **SD_nPWR / MS_nPWR / GPIOD[8]** | 164 |
| **Pad Name** | **NUC950ADN** |
| **NAND Flash(SM) / KPI** | **( 15pins )** |
| **SM_CS0n / KPI_ROW[0] / GPIOC[0]** | 142 |
| **SM_ALE / KPI_ROW[1] GPIOC[1]** | 140 |
| **SM_CLE / KPI_ROW[2] GPIOC[2]** | 141 |

| | |
|---|---|
| SM_WEn / KPI_ROW[3] / GPIOC[3] | 143 |
| SM_REn / GPIOC[4] | 144 |
| SM_WPn / GPIOC[5] | 145 |
| SM_RBn / GPIOC[6] | 146 |
| SM_D[7:0] / KPI_COL[7:0] / GPIOC[14:7] | 154–147 |
| **Pad Name** | **NUC950ADN** |
| **LCD** | **( 23 pins )** |
| VD [17:0] | 130–122,120,118-117,114-109 |
| HSYNC | 131 |
| VSYNC | 139 |
| VDEN | 132 |
| VICLK | 133 |
| VOCLK | 137 |
| **Pad Name** | **NUC950ADN** |
| **Miscellaneous** | **( 6 pins )** |
| nIRQ [2:0] / GPIOH[2:0] | 169-167 |
| nWDOG / GPIOI[16] | 2 |
| GPIOE13 | 165 |
| TEST | 166 |
| **Power/Ground** | **(48 pins)** |
| VDD18 | 9,27,76,96,115,135,170,197 |
| VDD33 | 15,26,47,62,87,101,119,136,162,182 |
| VSS | 7,8,24,28,37,61,77,85,97,104,116,121,134,138,155,171,183,199 |
| USBVDDC0 (3.3V) | 211 |
| USBVSSC0 | 210 |
| USBVDDT0 (3.3V) | 216 |
| USBVSST0 | 213 |
| USBVDDC1 (3.3V) | 202 |
| USBVSSC1 | 201 |
| USBVDDT1 (3.3V) | 207 |

| | |
|---|---|
| **USBVSST1** | 204 |
| **PLLVDD18** | 108,105 |
| **PLLVSS** | 107,106 |

# 5. PIN DESCRIPTION

## 5.1 Pin Description for Interface

| Pin Name | IO Type | Description |
|---|---|---|
| **Clock & Reset (5)** | | |
| EXTAL15M | I | 15MHz External Clock / Crystal Input for PLL |
| XTAL15M | O | 15MHz Crystal Output |
| EXTAL48M0 | O | 48MHz Crystal Output for USB2.0 PHY |
| XTAL48M0 | I | 48MHz Crystal Input for USB2.0 PHY |
| nRESET | I | System Reset (Low active) |
| **TAP Interface (5)** | | |
| TCK | ID | JTAG Test Clock, internal pull-down |
| TMS | IU | JTAG Test Mode Select, internal pull-up |
| TDI | IU | JTAG Test Data in, internal pull-up |
| TDO | O | JTAG Test Data out |
| nTRST | IU | JTAG Reset, active-low, internal pull-up |
| **External Bus Interface (72)** | | |
| MA [21:0] | O | Address Bus of external memory and IO devices.<br>(MA[21:13] are set to input mode when nRESET low active) |
| MD [31:0] | IO (D) | Data Bus of external memory and IO device<br>(Pull-down are programmable) |
| nWBE [3:0] /<br>SDQM [3:0] | O | Write Byte Enable for specific device (nECS [2:0]).<br>Data Bus Mask signal for SDRAM (nSCS [1:0]), (Low active) |
| nSCS [1:0] | O | SDRAM chip select for two external banks, (Low active) |
| nSRAS | O | Row Address Strobe for SDRAM, (Low active) |
| nSCAS | O | Column Address Strobe for SDRAM, (Low active) |
| nSWE | O | SDRAM Write Enable, (Low active) |
| MCKE | O | SDRAM Clock Enable |
| MCLK | O | System Master Clock Out, SDRAM clock |
| nWAIT | IU | External Wait, (Low active), internal pull-up |
| nBTCS | O | ROM/Flash Chip Select, (Low active) |
| nECS [2:0] | O | External I/O Chip Select, (Low active) |
| nOE | O | ROM/Flash, External Memory Output Enable, (Low active) |
| **Ethernet RMII Interface (10)** | | |
| PHY_MDC | O(IS) | RMII Management Data Clock |

| Pin Name | IO Type | Description |
|---|---|---|
| PHY_MDIO | IO(D) | RMII Management Data I/O<br>(Pull-down is programmable) |
| PHY_TXD [1:0] | O(IU) | RMII Transmit Data bus<br>(Pull-up are programmable) |
| PHY_TXEN | O(ID) | RMII Transmit Enable<br>(Pull-down is programmable) |
| PHY_REFCLK | O(ID) | RMII Reference Clock.<br>(Pull-down is programmable) |
| PHY_RXD [1:0] | I(OU) | RMII Receive Data bus<br>(Pull-up are programmable) |
| PHY_CRSDV | I(OD) | RMII Carrier Sense / Receive Data Valid<br>(Pull-down is programmable) |
| PHY_RXERR | I(OD) | RMII Receive Data Error<br>(Pull-down is programmable) |
| AC97/I2S/PWM (5) | | |
| AC97_nRESET /<br>I2S_SYSCLK | O(ID) | AC97 Controller RESET Output.<br>I2S Controller System Clock Output.<br>(Pull-down is programmable) |
| AC97_DATAI /<br>I2S_DATAI /<br>PWM [0] | IO(D) | AC97 Controller Data Input.<br>I2S Controller Data Input.<br>PWM Channel 0 Output.<br>(Pull-down is programmable) |
| AC97_DATAO /<br>I2S_DATAO /<br>PWM [1] | O(ID) | AC97 Controller Data Output.<br>I2S Controller Data Output.<br>PWM Channel 1 Output.<br>(Pull-down is programmable) |
| AC97_SYNC /<br>I2S_WS /<br>PWM [2] | IO(D) | AC97 Controller Synchronous Pulse Output.<br>I2S Controller Word Select.<br>PWM Channel 2 Output.<br>(Pull-down is programmable) |
| AC97_BITCLK /<br>I2S_BITCLK /<br>PWM [3] | IOSD | AC97 Controller Bit Clock Input.<br>I2S Controller Bit Clock.<br>PWM Channel 3 Output.<br>(Pull-down with Schmitt trigger input ) |
| USB Interface (10) | | |
| DP0 | IO | Differential Positive USB Port0 IO signal |
| DN0 | IO | Differential Negative USB Port0 IO signal |
| REXT0 | A | External Resister Connect for Port0 |

| Pin Name | IO Type | Description |
|---|---|---|
| DP1 | IO | Differential Positive USB Port1 IO signal |
| DN1 | IO | Differential Negative USB Port1 IO signal |
| REXT1 | A | External Resister Connect for Port1 |
| UPWR1 | O | USB Port1 Power Control signal |
| OVI | I | USB Over Current Detection signal |
| HDS | I | USB PHY 0 Device/Host Mode Select Control signal |
| UPWR0 | O | USB Port0 Power Control signal<br>This pin is always driven to Low when USB Port0 is at Device mode (the HDS pin at high state) |
| I2C/USI(SPI/MW) Interface (4) | | |
| SCL0 /<br>SFRM | IOS | I2C Serial Clock Line 0.<br>USI Serial Frame.<br>(Input with Schmitt trigger) |
| SDA0 /<br>SSPTXD | IOS | I2C Serial Data Line 0.<br>USI Serial Transmit Data.<br>(Input with Schmitt trigger) |
| SCL1 /<br>SCLK | IOS | I2C Serial Clock Line 1.<br>USI Serial Clock.<br>(Input with Schmitt trigger) |
| SDA1 /<br>SSPRXD | IOS | I2C Serial Data Line 1.<br>USI Serial Receive Data.<br>(Input with Schmitt trigger) |
| UART0/UART1/UART2 Interface (6) | | |
| TXD0 | IO(D) | UART0 Transmit Data.<br>(Pull-down is programmable) |
| RXD0 | IO(D) | UART0 Receive Data.<br>(Pull-down is programmable) |
| TXD1 | IO(D) | UART1 Transmit Data<br>(Pull-down is programmable) |
| RXD1 | IO(D) | UART1 Receive Data<br>(Pull-down is programmable) |
| TXD2(IrDA) | IO(D) | UART2 Transmit Data supporting SIR IrDA.<br>(Pull-down is programmable) |
| RXD2(IrDA) | IO(D) | UART2 Receive Data supporting SIR IrDA.<br>(Pull-down is programmable) |
| SD/SDIO/Memory Stick Interface (8) | | |

| Pin Name | IO Type | Description |
|---|---|---|
| SD0_CMD / MS0_BS | IO(U) | SD/SDIO Mode – Command/Response (SPI Mode – Data In)<br>Memory Stick Mode – Bus State.<br>(Pull-up is programmable) |
| SD0_CLK / MS0_CLK | IO(U) | SD/SDIO Mode – Clock; (SPI Mode – Clock)<br>Memory Stick Mode – Clock<br>(Pull-up is programmable) |
| SD0_DAT0 / MS0_DAT0 | IO(U) | SD/SDIO Mode – Data Line Bit 0;<br>Memory Stick Mode – Data Line Bit 0;<br>(Pull-up is programmable) |
| SD0_DAT1 / MS0_DAT1 | IO(U) | SD/SDIO Mode – Data Line Bit 1;<br>Memory Stick Mode – Data Line Bit 1;<br>(Pull-up is programmable) |
| SD0_DAT2 / MS0_DAT2 | IO(U) | SD/SDIO Mode – Data Line Bit 2;<br>Memory Stick Mode – Data Line Bit 2;<br>(Pull-up is programmable) |
| SD0_DAT3 / MS0_DAT3 | IO(U) | SD/SDIO Mode – Data Line Bit 3;<br>Memory Stick Mode – Data Line Bit 3;<br>(Pull-up is programmable) |
| SD0_CDn / MS0_CDn | IO(U) | SD/SDIO Mode – Card Detect.<br>Memory Stick Mode – Card Detect.<br>(Pull-up is programmable) |
| SD_nPWR | IO(U) | SD/SDIO Power FET Control Signal Output.<br>(Pull-up is programmable) |
| **NAND Flash Interface (15)** | | |
| SM_CS0n | O(IU) | NAND Flash Chip Select #0<br>(Pull-up is programmable) |
| SM_ALE | O(IU) | NAND Flash Address Latch Enable<br>(Pull-up is programmable) |
| SM_CLE | O(IU) | NAND Flash Command Latch Enable<br>(Pull-up is programmable) |
| SM_WEn | O(IU) | NAND Flash Write Enable (Low active)<br>(Pull-up is programmable) |
| SM_REn | O(IU) | NAND Flash Read Enable (Low active)<br>(Pull-up is programmable) |
| SM_WPn | O(IU) | NAND Flash Write Protect (Low active)<br>(Pull-up is programmable) |
| SM_RBn | I(OU) | NAND Flash Busy (Low active)<br>(Pull-up is programmable) |

| Pin Name | IO Type | Description |
|---|---|---|
| SM_D[7:0] | IO(U) | NAND Flash Data Bus<br>(Pull-up is programmable) |
| **Keypad Interface (KPI) (12)** | | |
| KPI_COL[7:0] | I | Keypad Column Scan Input Bus<br>This bus is shared with NAND Flash Interface, which is programmable setting. |
| KPI_ROW[3:0] | O | Keypad Row Scan Output Bus<br>This bus is shared with NAND Flash Interface, which is programmable setting. |
| **LCD Interface (23)** | | |
| VD [17:0] | O(IU) | LCD Pixel Data Output.<br>(Pull-up is programmable) |
| HSYNC | O | Horizontal Sync or Line Sync. |
| VSYNC | O | Vertical Sync or Frame Sync. |
| VDEN | O | Data Enable or Display Control Signal. |
| VOCLK | O | Pixel Clock Output. |
| VICLK | IU | Pixel Clock Input. |
| **Miscellaneous(6)** | | |
| nIRQ[2:0] | I(OU) | External Interrupt Request<br>(Pull-up is programmable) |
| nWDOG | O | Watchdog Timer Timeout Flag (Low active) |
| GPIOE13 | IO(U) | Bit 13 of the GPIOE port |
| TEST | I | Test Mode<br>This pin has to pull low in normal operation. |
| **Power/Ground** | | |
| VDD18 | P | Core Logic power (1.8V) |
| VDD33 | P | IO Buffer power (3.3V) |
| VSS | G | IO Buffer and Core ground (0V) |
| USBVDDC0 | P | USB Port0 PHY power (3.3V) |
| USBVSSC0 | G | USB Port0 PHY ground (0V) |
| USBVDDT0 | P | USB Port0 PHY Transceiver power (3.3V) |
| USBVSST0 | G | USB Port0 PHY Transceiver ground (0V) |
| USBVDDC1 | P | USB Port1 PHY power (3.3V) |
| USBVSSC1 | G | USB Port1 PHY ground (0V) |
| USBVDDT1 | P | USB Port1 PHY Transceiver power (3.3V) |
| USBVSST1 | G | USB Port1 PHY Transceiver ground (0V) |
| PLLVDD18 | P | PLL power (1.8V) |
| PLLVSS18 | G | PLL ground (0V) |

## 5.2 GPIO Share Pin Description

In this chip, there are GPIOC~GPIOH groups for general IO control. All of GPIO pins are shared with the other interface and define as the following

| GPIO Group | Shared pin function |
|---|---|
| GPIOC (15 pins) | NAND Flash Interface / KPI |
| GPIOC[0] | SM_CS0n / KPI_ROW[0] |
| GPIOC[1] | SM_ALE / KPI_ROW[1] |
| GPIOC[2] | SM_CLE / KPI_ROW[2] |
| GPIOC[3] | SM_WEn / KPI_ROW[3] |
| GPIOC[4] | SM_REn |
| GPIOC[5] | SM_WPn |
| GPIOC[6] | SM_RBn |
| GPIOC[7] | SM_D[0] / KPI_COL[0] |
| GPIOC[8] | SM_D[1] / KPI_COL[1] |
| GPIOC[9] | SM_D[2] / KPI_COL[2] |
| GPIOC[10] | SM_D[3] / KPI_COL[3] |
| GPIOC[11] | SM_D[4] / KPI_COL[4] |
| GPIOC[12] | SM_D[5] / KPI_COL[5] |
| GPIOC[13] | SM_D[6] / KPI_COL[6] |
| GPIOC[14] | SM_D[7] / KPI_COL[7] |

| | |
|---|---|
| GPIOD (8 pins) | SD(SDIO) / Memory Stick Interface |

| | |
|---|---|
| GPIOD[0] | SD_CMD / MS_BS |
| GPIOD[1] | SD_CLK / MS_CLK |
| GPIOD[2] | SD_DAT0 / MS_DAT0 |
| GPIOD[3] | SD_DAT1 / MS_DAT1 |
| GPIOD[4] | SD_DAT2 / MS_DAT2 |
| GPIOD[5] | SD_DAT3 / MS_DAT3 |
| GPIOD[6] | SD_CDn / MS_CDn |
| GPIOD[8] | SD_nPWR / MS_nPWR |

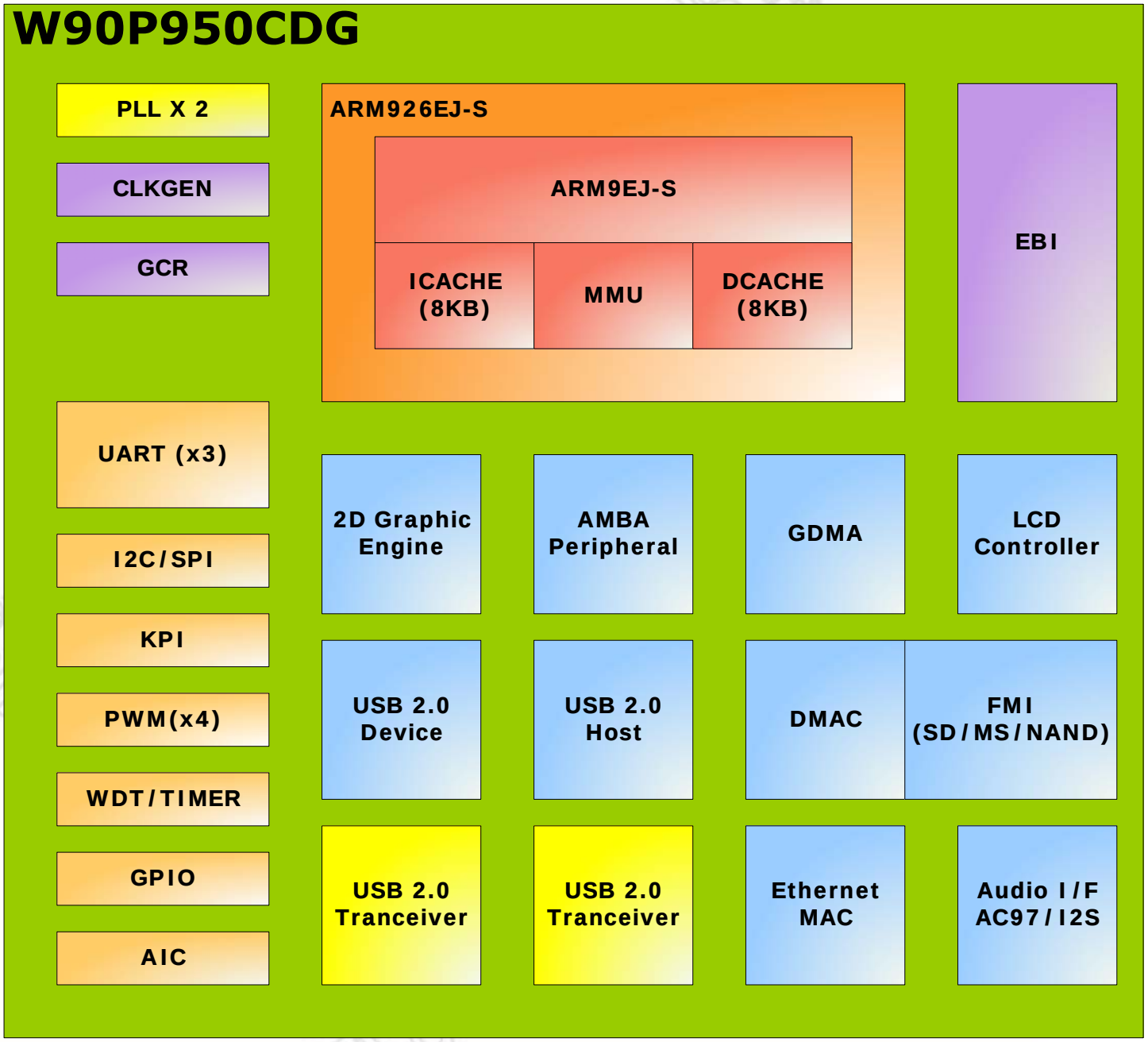
| GPIOE (7 pins) | UART Interface |
|---|---|
| GPIOE[0] | TXD0 |
| GPIOE[1] | RXD0 |
| GPIOE[2] | TXD1 |
| GPIOE[3] | RXD1 |
| GPIOE[6] | TXD2(IrDA) |
| GPIOE[7] | RXD2(IrDA) |
| GPIOE[13] | GPIOE13 |

| GPIOF (10 pins) | RMII Interface |
|---|---|
| GPIOF [0] | PHY_MDC |
| GPIOF [1] | PHY_MDIO |
| GPIOF [3:2] | PHY_TXD [1:0] |
| GPIOF [4] | PHY_TXEN |
| GPIOF [5] | PHY_REFCLK |
| GPIOF [7:6] | PHY_RXD [1:0] |
| GPIOF [8] | PHY_CRSDV |
| GPIOF [9] | PHY_RXERR |

| GPIOG (9 pins) | I2C/USI<br>XDMA,<br>AC97/I2S/PWM Interface |
|---|---|
| GPIOG[0] | SCL0 /<br>SFRM |
| GPIOG[1] | SDA0 /<br>SSPTXD |
| GPIOG[2] | SCL1 /<br>SCLK |
| GPIOG[3] | SDA1 /<br>SSPRXD |
| GPIOG[12] | AC97_nRESET<br>I2S_SYSCLK |
| GPIOG[13] | AC97_DATAI /<br>I2S_DATAI /<br>PWM [0] |
| GPIOG[14] | AC97_DATAO /<br>I2S_DATAO /<br>PWM [1] |
| GPIOG[15] | AC97_SYNC /<br>I2S_WS /<br>PWM [2] |
| GPIOG[16] | AC97_BITCLK /<br>I2S_BITCLK /<br>PWM [3] |

| GPIOH (3 pins) | nIRQ Interface |
|---|---|
| GPIOH[2:0] | nIRQ[2:0] |

## 6. FUNCTIONAL BLOCK



# 7. FUNCTIONAL DESCRIPTION

## 7.1 ARM926EJ-S CPU CORE

The ARM926EJ-S CPU core is a member of the ARM9 family of general-purpose microprocessors. The ARM926EJ-S CPU core supports the 32-bit ARM and 16-bit Thumb instruction sets, enabling the user to tradeoff between high performance and high code density. The ARM926EJ-S processor has a Harvard cached architecture with MMU and provides a complete high-performance processor subsystem.

## 7.2 System Manager

### 7.2.1 Overview

The System Manager has the following functions.

- System memory map
- The width of external memory address
- Data bus connection with external memory
- Product identifier register
- Bus arbitration
- PLL module
- Clock select register
- Power-On setting

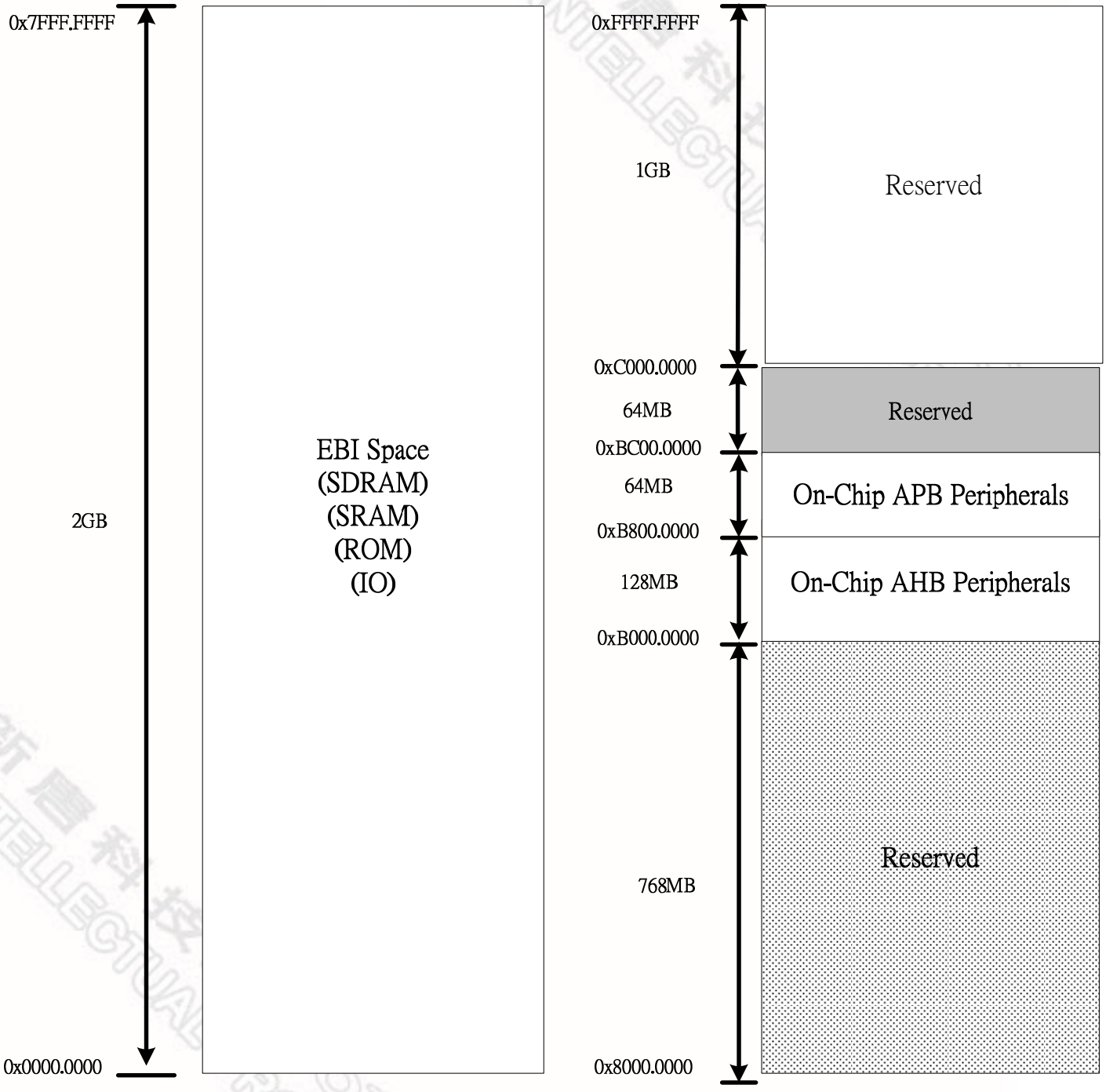
### 7.2.2 System Memory Map

This chip provides 2G bytes memory space (0x0000_0000~0x7FFF_FFFF) for the SDRAM, RAM, ROM and IO Devices, 192M bytes space (0xB000_0000~0xBBFF_FFFF) for On-Chip Peripherals and the other memory spaces are reserved.

The size and location of each SDRAM memory bank is determined by the register settings for "current bank base address pointer" and "current bank size" (SDCONF0 and SDCONF1). Please note that when setting the bank control registers, the address boundaries of consecutive banks must not be overlapped.

Except On-Chip Peripherals, the start address of each memory bank is not fixed. You can use bank control registers to assign a specific bank start address by setting the bank's base pointer (13 bits). The address resolution is 256K bytes. The bank's start address is defined as "base pointer << 18", and the bank's size is "current bank size". (EXT0CON ~ EXT2CON)

The CPU booting start address is fixed at address 0x0000_0000 after reset or power-on. In the event of an access request to an address outside any programmed bank size, an abort signal is generated. The maximum accessible memory size of each external IO bank is 4MB@8bit (8MB@16bit and 16MB@32bit), and 128M bytes on SDRAM banks.


0x7FFF.FFFF
2GB
0x0000.0000
EBI Space
(SDRAM)
(SRAM)
(ROM)
(IO)
0xFFFF.FFFF
1GB
Reserved
0xC000.0000
64MB
Reserved
0xBC00.0000
64MB
On-Chip APB Peripherals
0xB800.0000
128MB
On-Chip AHB Peripherals
0xB000.0000
768MB
Reserved
0x8000.0000

| Address Space | Token | Modules |
|---|---|---|
| 0x0000_0000 – 0x7FFF_FFFF | ------ | EBI(SDRAM, ROM, RAM, IO) Memory Space |
| 0x8000_0000 – 0xAFFF_FFFF | ------ | Reserved<br>Shadow of EBI Memory Space(0x0000_0000~0x2FFF_FFFF) |
| 0xB000_0000 – 0xB000_01FF | GCR_BA | System Global Control Registers |
| 0xB000_0200 – 0xB000_02FF | CLK_BA | Clock Control Registers |
| 0xB000_1000 – 0xB000_1FFF | EBI_BA | EBI Control Registers |
| 0xB000_3000 – 0xB000_3FFF | EMC_BA | Ethernet MAC Control Registers |
| 0xB000_4000 – 0xB000_4FFF | GDMA_BA | GDMA Control Registers |
| 0xB000_5000 – 0xB000_5FFF | USBH_BA | EHCI USB Host Control Registers |
| 0xB000_6000 – 0xB000_6FFF | USBD_BA | USB Device Control Registers |
| 0xB000_7000 – 0xB000_7FFF | USBO_BA | OHCI USB Host Control Registers |
| 0xB000_8000 – 0xB000_8FFF | LCM_BA | Display, LCM Interface Control Registers |
| 0xB000_9000 – 0xB000_9FFF | ACTL_BA | Audio Interface Control Registers |
| 0xB000_A000 – 0xB000_AFFF | ------ | Reserved |
| 0xB000_B000 – 0xB000_BFFF | GE_BA | 2D Graphic Engine Control Register |

| Address Space | Token | Modules |
|---|---|---|
| 0xB000_C000 – 0xB000_CFFF | DMAC_BA | DMA Controller Registers |
| 0xB000_D000 – 0xB000_DFFF | FMI_BA | Flash Memory Interface Control Registers |

| Address Space | Token | Modules |
|---|---|---|
| 0xB800_0000 – 0xB800_00FF | UART0_BA | UART 0 Control Registers |
| 0xB800_0100 – 0xB800_01FF | UART1_BA | UART 1 Control Registers |
| 0xB800_0200 – 0xB800_02FF | UART2_BA | UART 2 Control Registers (Tx,Rx for IrDA) |
| 0xB800_0300 – 0xB800_03FF | ------ | Reserved |
| 0xB800_0400 – 0xB800_04FF | ------ | Reserved |
| 0xB800_1000 – 0xB800_1FFF | TMR_BA | Timer Control Registers |
| 0xB800_2000 – 0xB800_2FFF | AIC_BA | Interrupt Controller Registers |
| 0xB800_3000 – 0xB800_3FFF | GPIO_BA | GPIO Control Registers |
| 0xB800_4000 – 0xB800_4FFF | ------ | Reserved |
| 0xB800_5000 – 0xB800_5FFF | ------ | Reserved |
| 0xB800_6000 – 0xB800_60FF | I2C0_BA | I2C 0 Control Register |
| 0xB800_6100 – 0xB800_61FF | I2C1_BA | I2C 1 Control Register |
| 0xB800_6200 – 0xB800_62FF | USI_BA | Universal Serial Interface Register (USI) |
| 0xB800_7000 – 0xB800_7FFF | PWM_BA | Pulse Width Modulation(PWM) Control Registers |
| 0xB800_8000 – 0xB800_8FFF | KPI_BA | Keypad Interface Control Registers |

| Address Space | Token | Modules |
|---|---|---|
| 0xB800_9000 – 0xB800_9FFF | ------ | Reserved |
| 0xB800_A000 – 0xB800_AFFF | ------ | Reserved |

### 7.2.3 Address Bus Generation

The address bus generation is depended on the required data bus width **(DBWD)** and address bus alignment control bit **(ADRS)** of each IO bank.

**Address Bus Generation Guidelines (When ADRS bit = 0)**

| Data Bus Width | External Address Pins MA [21:0] | Maximum Accessible Memory Size |
|---|---|---|
| 8-bit | MA21 – MA0 (Internal) | 4M bytes |
| 16-bit | MA22 – MA1 (Internal) | 8M bytes (4M half-words) |
| 32-bit | MA23 – MA2 (Internal) | 16M bytes (4M words) |

**Address Bus Generation Guidelines (When ADRS bit = 1)**

| Data Bus Width | External Address Pins MA [21:0] | Maximum Accessible Memory Size |
|---|---|---|
| 8-bit | MA21 – MA0 (Internal) | 4M bytes |
| 16-bit | MA21 – MA0 (Internal) | 4M bytes, MA[0] ignored (2M half-words) |
| 32-bit | MA21 – MA0 (Internal) | 4M bytes, MA[1:0] ignored (1M words) |

### AHB Bus Arbitration

The system bus is AHB-compliant and supports modules with standard AHB master or slave interfaces. The AHB arbiter has two priority-decision modes, i.e., the fixed priority mode and the rotate priority mode. In the rotate priority mode, there are three types for AHB-Master bus. The selection of modes and types is determined on the **PRTMOD0** and **PRTMOD1**bits in the Arbitration Control Register. **PRTMOD0** is used to control the fixed priority of AHB1 (CPU AHB-Lite) Bus and **PRTMOD1** is used to control the fixed priority of AHB2 Master Bus.

#### 7.2.4.1 Fixed Priority Mode

Fixed priority mode is selected if **PRTMODx** = 0. The order of priorities on the AHB mastership among the on-chip master modules is fixed. If two or more master modules request to AHB at the same time, the mastership is always granted to the module with the highest priority.

**AHB Bus Priority Order in Fixed Priority Mode**

| Priority Sequence | PRTMOD0 = 0 AHB1 Bus | PRTMOD1 = 0 AHB2 Bus |
|---|---|---|
| 1 (Lowest) | ARM CPU Instruction | AHB Bridge |
| 2 | ARM CPU Data | 2-D Graphic |
| 3 | GDMA0 | --- |
| 4 | GDMA1 | SDIO(FMI) |
| 5 | --- | USB Device |
| 6 | --- | USB Host |
| 7 | --- | EMC Controller |
| 8 | --- | LCD Controller |
| 9 (Highest) | --- | Audio Controller (AC97 & I2S) |

The ARM core normally has the lowest priority under the fixed priority mode; however, this chip provides a mechanism to raise the priority to the highest. If the IPEN bit (bit-1 of Arbitration Control Register) is set to 1, the **IPACT** bit (bit-2 of Arbitration Control Register) will be automatically set to 1 while an unmasked external interrupt occurs. Under this circumstance, the ARM core gains the highest AHB priority.

The programmer can recover the original priority order by directly writing "0" to clear the **IPACT** bit. For example, this can be done that at the end of an interrupt service routine. Note that **IPACT** only can be automatically set to 1 by an external interrupt when **IPEN** = 1. It will not take effect if a programmer to directly write 1 to **IPACT** to raise ARM core's AHB priority.

#### 7.2.4.2 Rotate Priority Mode

Rotate priority mode is selected if PRTMODx = 1. The AHB arbiter uses a round robin arbitration scheme by which every master module can gain the bus ownership in turn.

For AHB2 DMA Master Bus, the Audio and LCD Display, have the higher priority in the rotate type.

### Power-On Setting

After power on reset, Power-On setting registers are latched from EBI Address pins (MA [21:13]) to configure this chip.

| Power-On Setting | Pin |
|---|---|
| Booting Device Select | MA [21:20] |
| Internal System Clock Select | MA17 |
| GPIO Pin Configuration Select | MA [15:14] |
| USB PHY0 Mode Select | HDS |

**MA [21:20] : Booting Device Select**

| MA[21:20] | | Booting Device |
|---|---|---|
| Pull-down | Pull-down | SPI Flash ROM |
| Pull-down | Pull-up | NAND-type Flash ROM |
| Pull-up | Pull-down | USB ISP |
| Pull-up | Pull-up | NOR-type Flash ROM |

**MA19 : Pull-up is necessary**

**MA18 : Can either Pull-up or Pull-down**

**MA17 : Internal System Clock Select**

If pin MA17 is pull-down, the external clock from EXTAL15M pin is served as internal system clock.

If pin MA17 is pull-up, the PLL output clock is used as internal system clock.

**MA16 : Pull-down is necessary**

**MA [15:14] : GPIO Pin Configuration Select**

| MA[15:14] | State | GPIO Pin Function |
|---|---|---|
| MA14 | Pull-down | GPIOC/D/E Group Select |
| | Pull-up | NAND Flash/UART Group Select |
| MA15 | Pull-down | GPIOF Group Select |
| | Pull-up | RMII Group Select |

**MA13 : Pull-up is necessary**

**HDS: USB PHY0 Mode Select**

| HDS | USB PHY0 Mode |
|---|---|
| Pull-down | USB20 Host |
| Pull-up | USB20 Device |

**System Booting**

NUC950ADN supports four kinds of system booting devices, which including

(1) SPI Flash ROM

(2) NAND-type Flash ROM

(3) USB ISP

(4) NOR-type Flash ROM

**Booting Device Select**

| MA[21:20] | | Booting Device |
|---|---|---|
| Pull-down | Pull-down | SPI Flash ROM |
| Pull-down | Pull-up | NAND-type Flash ROM |
| Pull-up | Pull-down | USB ISP |
| Pull-up | Pull-up | NOR-type Flash ROM |

### 7.2.7 System Global Control Registers Map

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GCR_BA = 0xB000_0000** | | | | |
| **PDID** | 0xB000_0000 | R | Product Identifier Register | 0xxx90_09x0 |
| **PWRON** | 0xB000_0004 | R/W | Power-On Setting Register | N/A |
| **ARBCON** | 0xB000_0008 | R/W | Arbitration Control Register | 0x0000_0000 |
| **MFSEL** | 0xB000_000C | R/W | Multiple Function Pin Select Register | 0x0800_0000 |
| **EBIDPE** | 0xB000_0010 | R/W | EBI Data Pin Pull-up/down Enable Register | 0xFFFF_FFFF |
| **LCDDPE** | 0xB000_0014 | R/W | LCD Data Pin Pull-up/down Enable Register | 0x0003_FFFF |
| **GPIOCPE** | 0xB000_0018 | R/W | GPIOC Pin Pull-up/down Enable Register | 0x0000_7FFF |
| **GPIODPE** | 0xB000_001C | R/W | GPIOD Pin Pull-up/down Enable Register | 0x0000_07FF |
| **GPIOEPE** | 0xB000_0020 | R/W | GPIOE Pin Pull-up/down Enable Register | 0x0000_3FFF |
| **GPIOFPE** | 0xB000_0024 | R/W | GPIOF Pin Pull-up/down Enable Register | 0x0000_03FF |
| **GPIOGPE** | 0xB000_0028 | R/W | GPIOG Pin Pull-up/down Enable Register | 0x0001_FFFF |
| **GPIOHPE** | 0xB000_002C | R/W | GPIOH Pin Pull-up/down Enable Register | 0x0000_00FF |
| **GTMP1** | 0xB000_0034 | R/W | General Temporary Register 1 | N/A |
| **GTMP2** | 0xB000_0038 | R/W | General Temporary Register 2 | N/A |

| | | | | |
|---|---|---|---|---|
| **GTMP3** | 0xB000_003C | R/W | General Temporary Register 3 | N/A |

## Product Identifier Register（PDID）

This register is for only read and enables software to recognize certain characteristics of the chip ID and the version number.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| PDID | 0xB000_0000 | R | Product Identifier Register | 0xxx90_09x0 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| VERSION | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CHPID | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CHPID | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CHPID | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:24] | **VERSION** | **Version of chip** |
| [23:0] | **CHIPID** | **Chip identifier** |

## Power-On Setting Register（PWRON）

This register latches the chip power-on setting from EBI Address Bus during chip reset.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| PWRON | 0xB000_0004 | R/W | Power-On Setting Register | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | USBDEN | USBHD | RESERVED |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Booting Device Select | | RESERVED | | | GPIOSEL | | PLL |

| Bits | Descriptions | |
|---|---|---|
| [0] | **PLL** | **Internal System Clock Select (Read/Write)**<br>Power-On value latched from MA17<br>0= the external clock from EXTAL15M pin is served as internal system clock.<br>1= the PLL output clock is used as internal system clock. |
| [2:1] | **GPIOSEL** | **GPIO Pin Configuration Select(Read Only)**<br><br>**NF: NAND Type Flash Interface** |
| [5:3] | **RESERVED** | **Read Only**<br>These three bits are read only |

| | Latched pin | H/L | GPIO Pin Function |
|---|---|---|---|
| [1] | MA14 | 0 | GPIOC/D/E |
| | | 1 | NF/UART |
| [2] | MA15 | 0 | GPIOF |
| | | 1 | RMII |

<table>
<tr><td>[7:6]</td><td>Booting Device Select</td><td>Booting Device Select (Read Only)<br>these two bits are power-on reset from MA[21:20]
<table>
<tr><th colspan="2">Booting Device Select [7:6]</th><th>Booting Device</th></tr>
<tr><td>0</td><td>0</td><td>SPI Flash ROM</td></tr>
<tr><td>0</td><td>1</td><td>NAND-type Flash ROM</td></tr>
<tr><td>1</td><td>0</td><td>USB ISP</td></tr>
<tr><td>1</td><td>1</td><td>NOR-type Flash ROM</td></tr>
</table></td></tr>
<tr><td>[9]</td><td>USBHD</td><td>USB PHY0 Mode Select (Read/Write)<br>this bit is power-on reset from HDS
<table>
<tr><th>USBHD</th><th>USB PHY0 Mode</th><th>HDS Pin</th></tr>
<tr><td>0</td><td>USB20 Device</td><td>External Pull-Up</td></tr>
<tr><td>1</td><td>USB20 Host</td><td>External Pull-Down</td></tr>
</table></td></tr>
<tr><td>[10]</td><td>USBDEN</td><td>USB PHY0 Enable Control for USB Device Mode (Read/Write)<br>This bit is only active when the USBHD bit be zero (Device Mode)
<table>
<tr><th>USBDEN</th><th>USB PHY0 Enable</th></tr>
<tr><td>0</td><td>Set Device PHY at SE0 (Not active to external host)</td></tr>
<tr><td>1</td><td>Set Device PHY controlled by the UTMI interface of the USB Device Controller</td></tr>
</table></td></tr>
</table>

## Arbitration Control Register (ARBCON)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ARBCON | 0xB000_0008 | R/W | Arbitration Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | DGMASK | IPACT | IPEN | PRTMOD1 | PRTMOD0 |

| Bits | Descriptions | |
|---|---|---|
| [4] | **DGMASK** | **Default Grant Master Mask Control**<br>0 = AHB-Bridge always be the default grant master (default)<br>1 = No default grant master on AHB-2 Bus |
| [3] | **IPACT** | **Interrupt Priority Active**<br>When **IPEN**="1", this bit is set when the ARM core has an unmasked interrupt request.<br>This bit is available only when the **PRTMOD1**=0 and **PRTMOD0**=0. |
| [2] | **IPEN** | **Interrupt Priority Enable Bit**<br>0 = the ARM core has the lowest priority.<br>1 = enable to raise the ARM core priority to second<br>This bit is available only when the **PRTMOD**=0 and **PRTMOD0**=0. |
| [1] | **PRTMOD1** | **Priority Mode Select for AHB2 (AHB Master Bus)**<br>0 = Fixed Priority Mode (default)<br>1 = Rotate Priority Mode |
| [0] | **PRTMOD0** | **Priority Mode Select for AHB1 (CPU AHB-Lite Bus)**<br>0 = Fixed Priority Mode (default)<br>1 = Rotate Priority Mode |

## Multiple Function Pin Select Register (MFSEL)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MFSEL | 0xB000_000C | R/W | Multiple Function Pin Select Register | 0x0800_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | USBPHY0 | | WDEN | | RESERVED | GPSELH |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPSELG | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPSELG | | GPSELE | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPSELD | | | | GPSELC | | GPSELF | G-Option |

| Bits | Descriptions | |
|---|---|---|
| [29:28] | **USBPHY0** | **USB PHY0 Select Control Register**<br>**00 : Normal USB operation mode (Default)** |
| [27:26] | **WDEN** | **GPIOI Pin Function Select Control Register**<br>PIN / GPSELI[27] / GPIO Pin Function<br>GPIOI[16] / 0 / GPIOI[16]<br>GPIOI[16] / 1 / nWDOG<br>**See GPIO Shared Pin Description for more detail**<br>**GPSELI[27] default value is 1 for nWDOG ( Watch-Dog Timer Output )** |
| [24] | **GPSELH** | **GPIOH Pin Function Select Control Register**<br>PIN / GPSELH[24] / GPIO Pin Function<br>GPIOH[2:0] / 0 / GPIOH[2:0]<br>GPIOH[2:0] / 1 / nIRQ[2:0]<br>**GPSELH [24] default value is 0 for GPIOH group.** |

<table>
<tr><th>Bits</th><th>Descriptions</th><th></th></tr>
<tr><td>[23:14]</td><td>GPSELG</td><td>
GPIOG Pin Function Select Control Register
<table>
<tr><th>PIN</th><th>GPSELG[23:22]</th><th>GPIO Pin Function</th></tr>
<tr><td rowspan="4">GPIOG[16:12]</td><td>00</td><td>GPIOG[16:12]</td></tr>
<tr><td>01</td><td>PWM Interface</td></tr>
<tr><td>10</td><td>AC97 Interface</td></tr>
<tr><td>11</td><td>I2S Interface</td></tr>
<tr><th>PIN</th><th>GPSELG[17:16]</th><th>GPIO Pin Function</th></tr>
<tr><td rowspan="4">GPIOG[3:2]</td><td>00</td><td>GPIOG[3:2]</td></tr>
<tr><td>01</td><td>I2C Line1</td></tr>
<tr><td>10</td><td>USI Interface</td></tr>
<tr><td>11</td><td>Reserved</td></tr>
<tr><th>PIN</th><th>GPSELG[15:14]</th><th>GPIO Pin Function</th></tr>
<tr><td rowspan="4">GPIOG[1:0]</td><td>00</td><td>GPIOG[1:0]</td></tr>
<tr><td>01</td><td>I2C Line0</td></tr>
<tr><td>10</td><td>USI Interface</td></tr>
<tr><td>11</td><td>Reserved</td></tr>
</table>
See GPIO Shared Pin Description for more detail<br>
GPSELG [21-18] are Reserved.<br>
(G-Option bit should be set to 0)
</td></tr>
<tr><td>[13:8]</td><td>GPSELE</td><td>
GPIOE Pin Function Select Control Register<br>
See GPIO Shared Pin Description for more detail<br>
GPSELE [13:8] default value is 0 for GPIOE group.<br>
GPSELE[13] should be set to 0<br>
GPSELE [12:11] should be set to 0.
<table>
<tr><th></th><th></th><th>GPIO Pin Function</th></tr>
<tr><th>PIN</th><th>GPSELE[10]</th><th></th></tr>
<tr><td rowspan="2">GPIOE[7:4]</td><td>0</td><td>GPIOE[7:6]</td></tr>
<tr><td>1</td><td>UART2(IrDA)</td></tr>
<tr><th>PIN</th><th>GPSELE[9]</th><th></th></tr>
<tr><td rowspan="2">GPIOE[3:2]</td><td>0</td><td>GPIOE[3:2]</td></tr>
<tr><td>1</td><td>UART1</td></tr>
<tr><th>PIN</th><th>GPSELE[8]</th><th></th></tr>
<tr><td rowspan="2">GPIOE[1:0]</td><td>0</td><td>GPIOE[1:0]</td></tr>
<tr><td>1</td><td>UART0</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Bits</th><th>Descriptions</th><th></th></tr>
<tr><td>[7:4]</td><td>GPSELD</td><td>GPIOD Pin Function Select Control Register
<table>
<tr><th>PIN</th><th>GPSELD[7:4]</th><th>GPIO Pin Function</th></tr>
<tr><td rowspan="4">GPIOD[10:5]</td><td>0000</td><td>GPIOD[10:0]</td></tr>
<tr><td>1010</td><td>SD 0 Interface</td></tr>
<tr><td>1111</td><td>Memory Stick 0</td></tr>
<tr><td>the others</td><td>Reserved</td></tr>
</table>
See GPIO Shared Pin Description for more detail<br>
GPSELD[7:4] default value is depend on power-on setting</td></tr>
<tr><td>[3:2]</td><td>GPSELC</td><td>GPIOC Pin Function Select Control Register
<table>
<tr><th>PIN</th><th>GPSELC[3:2]</th><th>GPIO Pin Function</th></tr>
<tr><td rowspan="4">GPIOC[14:0]</td><td>00</td><td>GPIOC[14:0]</td></tr>
<tr><td>01</td><td>NAND Flash</td></tr>
<tr><td>10</td><td>KPI Interface</td></tr>
<tr><td>11</td><td>Reserved</td></tr>
</table>
See GPIO Shared Pin Description for more detail<br>
GPSELC[3:2] default value is depend on power-on setting</td></tr>
<tr><td>[1]</td><td>GPSELF</td><td>GPIOF Pin Function Select Control Register
<table>
<tr><th>PIN</th><th>GPSELF[1]</th><th>GPIO Pin Function</th></tr>
<tr><td rowspan="2">GPIOF[9:0]</td><td>0</td><td>GPIOF[9:0]</td></tr>
<tr><td>1</td><td>RMII Interface</td></tr>
</table>
See GPIO Shared Pin Description for more detail<br>
GPSELF[1] default value is depend on power-on setting</td></tr>
<tr><td>[0]</td><td>G-Option</td><td>This bit should be set to 0;</td></tr>
</table>

### EBI Data Pin Pull-up/down Enable Register (EBIDPE)

### LCD Data Pin Pull-up/down Enable Register (LCDDPE)

### GPIOC~GPIOH Pin Pull-up/down Enable Register (GPIOCPE~GPIOHPE)

These registers are used to control the IO pins to be internal pull-up or down, which can avoid the input pins floating if there is no external resistors.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EBIDPE** | 0xB000_0010 | R/W | EBI Data Pin Pull-down Enable Register | 0xFFFF_FFFF |
| **LCDDPE** | 0xB000_0014 | R/W | LCD Data Pin Pull-up Enable Register | 0x0003_FFFF |
| **GPIOCPE** | 0xB000_0018 | R/W | GPIOC Pin Pull-up Enable Register | 0x0000_7FFF |
| **GPIODPE** | 0xB000_001C | R/W | GPIOD Pin Pull-up Enable Register | 0x0000_07FF |
| **GPIOEPE** | 0xB000_0020 | R/W | GPIOE Pin Pull-up/down Enable Register | 0x0000_3FFF |
| **GPIOFPE** | 0xB000_0024 | R/W | GPIOF Pin Pull-up/down Enable Register | 0x0000_03FF |
| **GPIOGPE** | 0xB000_0028 | R/W | GPIOG Pin Pull-up/down Enable Register | 0x0001_FFFF |
| **GPIOHPE** | 0xB000_002C | R/W | GPIOH Pin Pull-up Enable Register | 0x0000_00FF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PPE | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| PPE | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| PPE | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| PPE | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **PPE** | **Pin Pull-down Enable Register**<br>1 = Disable the Pull-high/down for each relative pin (default)<br>0 = Enable the Pull-high/down for each relative pin |

| Register | Descriptions |
|---|---|
| **EBIDPE** | **EBI Data Pin Pull-down Enable Register**<br>PPE[31:0] Controls the Pull-down of the EBI Data Bus[31:0] |
| **LCDDPE** | **LCD Data Pin Pull-up Enable Register**<br>PPE[31:18] is reserved in this register<br>PPE[17:0] Controls the Pull-up of the VD[17:0] for LCD Interface |
| **GPIOCPE** | **GPIOC Pin Pull-up Enable Register**<br>PPE[31:15] is reserved in this register<br>PPE[1:0] Controls the Pull-up of the GPIOC[14:0] |
| **GPIODPE** | **GPIOD Pin Pull-up Enable Register**<br>PPE[31:9] is reserved in this register<br>PPE[8:0] Controls the Pull-up of the GPIOD[8:0]<br>No action : GPIOD[7] |
| **GPIOEPE** | **GPIOE Pin Pull-up/down Enable Register**<br>PPE[31:14] is reserved in this register<br>PPE[13:0] Controls the Pull-up/down of the GPIOE[13:0]<br>Pull-down : GPIOE[6:0]<br>Pull-up : GPIOE[13]<br>No action : GPIOE[12:7] and GPIOE[5:4] |
| **GPIOFPE** | **GPIOF Pin Pull-up/down Enable Register**<br>PPE[31:10] is reserved in this register<br>PPE[9:0] Controls the Pull-up/down of the GPIOF[9:0]<br>Pull-down : GPIOF[9:8], GPIOF[5:4], GPIOF[1]<br>Pull-up : GPIOF[7:6], GPIOF[3:2]<br>No action : GPIOF[0] |

| Register | Descriptions |
|---|---|
| **GPIOGPE** | **GPIOG Pin Pull-up/down Enable Register**<br>PPE[31:17] is reserved in this register<br>PPE[16:0] Controls the Pull-up of the GPIOG[16:0]<br>Pull-up : GPIOG[16:12]<br>No action : GPIOG[5:4], GPIOG[11:6] and GPIOG[3:0] |
| **GPIOHPE** | **GPIOH Pin Pull-up Enable Register**<br>PPE[31:3] is reserved in this register<br>PPE[2:0] Controls the Pull-up of the GPIOH[2:0] |

1 = Disable the Pull-high/down for each relative pin

0 = Enable the Pull-high/down for each relative pin

### General Temporary Register 1 ~ 3 (GTMP1 ~GTMP3)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GTMP1** | 0xB000_0034 | R/W | General Temporary Register 1 | Undefined |
| **GTMP2** | 0xB000_0038 | R/W | General Temporary Register 2 | Undefined |
| **GTMP3** | 0xB000_003C | R/W | General Temporary Register 3 | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DATA | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| DATA | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| DATA | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| DATA | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **DATA** | **General Temporary Data** |

## 7.3 Clock Controller

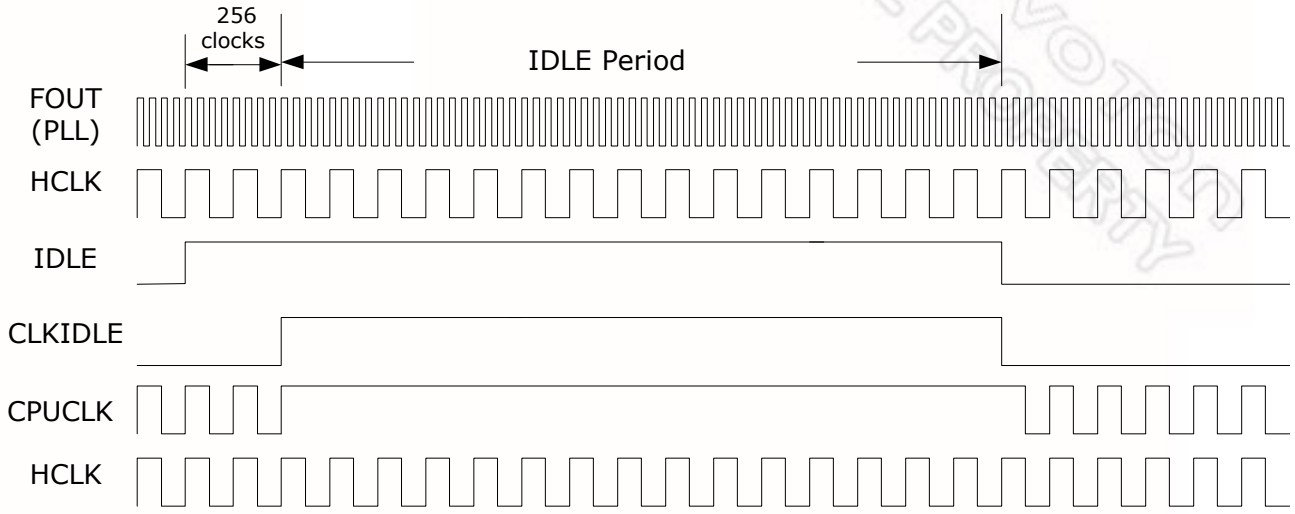
The clock controller generates all clocks for LCD, Audio, CPU, AMBA and all the engine modules. In this chip includes two PLL modules. The clock source for each module is come from the PLL, or from the external crystal input directly. For each clock there is bit on the CLKEN register to control the clock ON or OFF individually, and the divider setting is on the CLKDIV register. The register can also be used to control the clock enable or disable for power control.

### 7.3.1 Power management

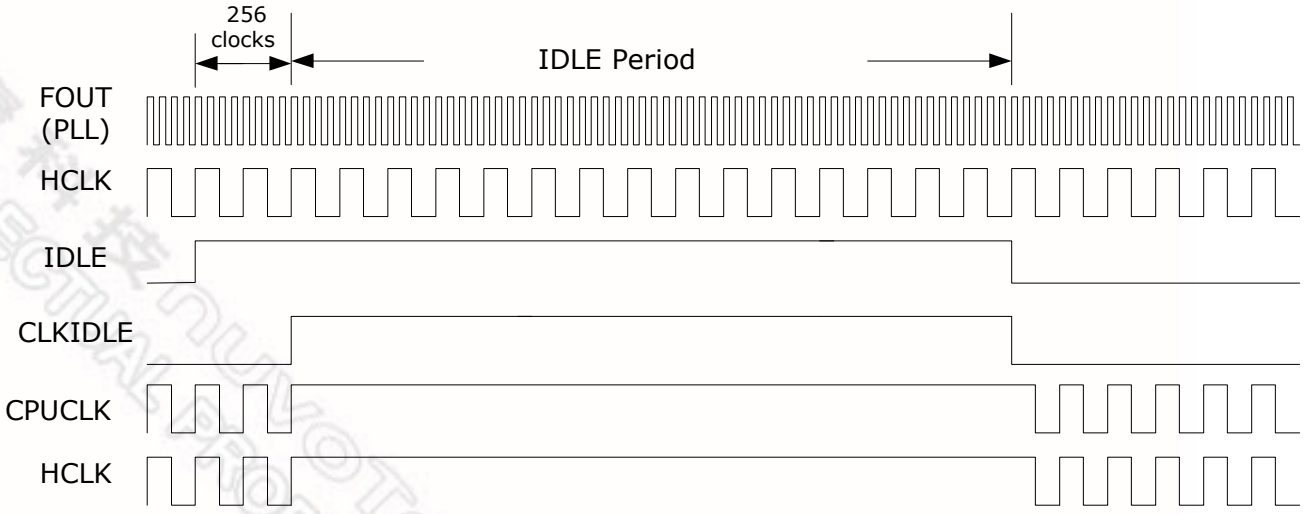
This chip provides three power management scenarios to reduce power consumption. The peripheral clocks can be enabled / disabled individually by controlling the corresponding bit in CLKSEL control register. Software can turn-off the unused modules' clock for power saving. It also provides **IDLE** and **Power-down** modes to reduce the power consumption.

### IDLE MODE

If the **IDLE** bit in Power Management Control Register (PMCON) is set, the ARM CORE clock source will be halted after 256 cycles, and then the ARM core will stop. The AHB or APB clocks are still active except the clock to cache controller and ARM core. This ARM core will exit from this mode when a **nIRQ** or **nFIQ** signals from any peripheral, such as Keypad and Timer overflow interrupts. The memory controller can also be forced to enter idle state if both the **MIDLE** and **IDLE** bits are set.



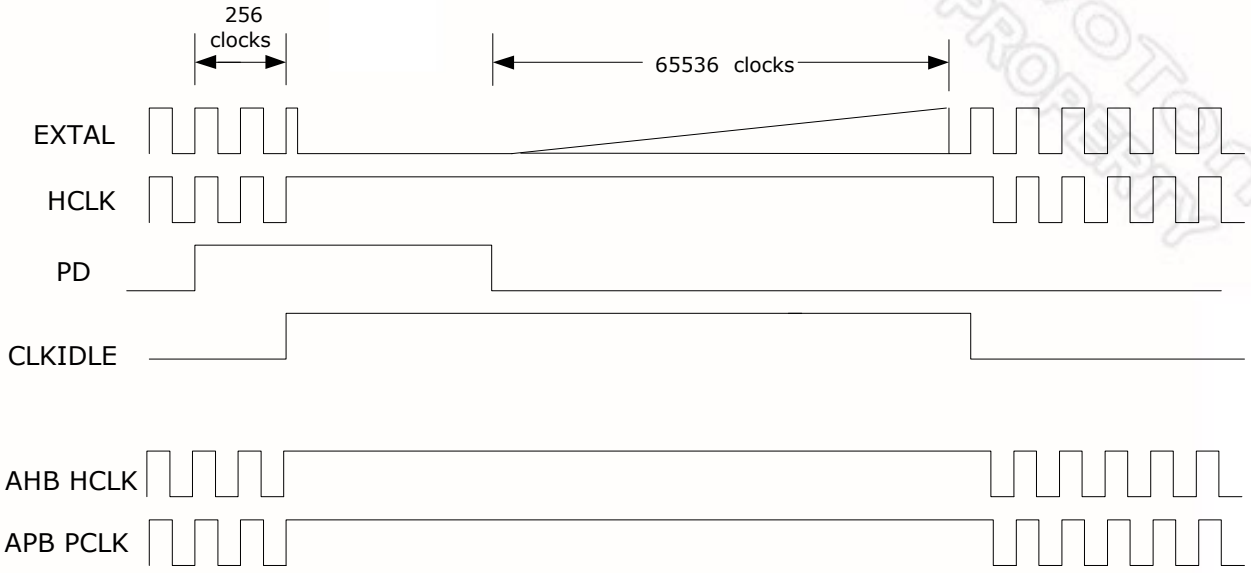

Case1. IDLE=1, PD=0, MIDLE=0




Case2. IDLE=1, PD=0, MIDLE=1

**Power-Down Mode**

The mode provides the minimum power consumption. When the system is not working or waiting an external event, software can write PD bit to turn off all the clocks includes system crystal oscillator and PLL to let ARM core to enter sleep mode after 256 clock cycles. In this state, all peripherals are also in sleep mode since the clock source is stopped. This system will exit from this mode when external interrupts (**nIRQ** signals) are detected; this chip provides external interrupts, USB device and Keypad to wakeup the clock.




Case3. IDLE=0, PD=1, MIDLE=0

### 7.3.2 Clock Control Registers Map

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_BA = 0xB000_0200** | | | | |
| **CLKEN** | 0xB000_0200 | R/W | Clock Enable Register | 0x0408_0834 |
| **CLKSEL** | 0xB000_0204 | R/W | Clock Select Register | 0x0000_0FFX |
| **CLKDIV** | 0xB000_0208 | R/W | Clock Divider Control Register | 0x0400_0000 |
| **PLLCON0** | 0xB000_020C | R/W | PLL Control Register 0 | 0x0000_2B63 |
| **PLLCON1** | 0xB000_0210 | R/W | PLL Control Register 1 | 0x0001_4F64 |
| **PMCON** | 0xB000_0214 | R/W | Power Management Control Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **IRQWAKECON** | 0xB000_0218 | R/W | IRQ Wakeup Control Register | 0x0000_0000 |
| **IRQWAKEFLAG** | 0xB000_021C | R/W | IRQ Wakeup Flag Register | 0x0000_0000 |
| **IPSRST** | 0xB000_0220 | R/W | IP Software Reset Register | 0x0000_0000 |
| **CLKEN1** | 0xB000_0224 | R/W | Clock Enable Register | 0x0000_0000 |
| **CLKDIV1** | 0xB000_0228 | R/W | Clock Divider Control 1 Register | 0x0000_0000 |

## Clock Enable Register (CLKEN)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLKEN | 0xB000_0200 | R/W | Clock Enable Register | 0x0408_0834 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| I2C1 | I2C0 | USI | RESERVED | GDMA | WDT | KPI | RESERVED |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TIMER4 | TIMER3 | TIMER2 | TIMER1 | TIMER0 | PWM | RESERVED | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | UART2 | UART1 | UART0 | G2D | USBH | USBD |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EMC | RESERVED | DMAC | FMI | RESERVED | | Audio | LCD |

| Bits | Descriptions | |
|---|---|---|
| [31] | **I2C1** | **I2C Interface 1 Clock Enable Bit**<br>0 = Disable I2C-1 clock<br>1 = Enable I2C-1 clock |
| [30] | **I2C0** | **I2C Interface 0 Clock Enable Bit**<br>0 = Disable I2C-0 clock<br>1 = Enable I2C-0 clock |
| [29] | **USI** | **USI Clock Enable Bit**<br>0 = Disable USI clock<br>1 = Enable USI clock |
| [28] | **Reserved** | **Reserved, write 0 is recommended.** |
| [27] | **GDMA** | **GDMA Clock Enable Bit**<br>0 = Disable GDMA clock<br>1 = Enable GDMA clock |
| [26] | **WDT** | **WDT Clock Enable Bit**<br>0 = Disable WDT counting clock<br>1 = Enable WDT counting clock |

| Bits | Descriptions | |
|---|---|---|
| [25] | **KPI** | **Keypad Cock Enable Bit**<br>0 = Disable keypad clock<br>1 = Enable keypad clock |
| [24] | **Reserved** | **Reserved, write 0 is recommended.** |
| [23] | **TIMER4** | **Timer4 Clock Enable Bit**<br>0 = Disable Timer clock<br>1 = Enable Timer clock |
| [22] | **TIMER3** | **Timer3 Clock Enable Bit**<br>0 = Disable Timer clock<br>1 = Enable Timer clock |
| [21] | **TIMER2** | **Timer2 Clock Enable Bit**<br>0 = Disable Timer clock<br>1 = Enable Timer clock |
| [20] | **TIMER1** | **Timer1 Clock Enable Bit**<br>0 = Disable Timer clock<br>1 = Enable Timer clock |
| [19] | **TIMER0** | **Timer0 Clock Enable Bit**<br>0 = Disable Timer clock<br>1 = Enable Timer clock |
| [18] | **PWM** | **PWM Clock Enable Bit**<br>0 = Disable PWM clock<br>1 = Enable PWM clock |
| [17:14] | **Reserved** | **Reserved, write 4'b0000 is recommended.** |
| [13] | **UART2** | **UART2 Clock Enable Bit**<br>0 = Disable UART2 clock<br>1 = Enable UART2 clock |
| [12] | **UART1** | **UART1 Clock Enable Bit**<br>0 = Disable UART1 clock<br>1 = Enable UART1 clock |

| Bits | Descriptions | |
|---|---|---|
| [11] | **UART0** | **UART0 Clock Enable Bit**<br>0 = Disable UART0 clock<br>1 = Enable UART0 clock |
| [10] | **G2D** | **2D Graphic Controller Clock Enable Bit**<br>0 = Disable 2D Graphic Controller clock<br>1 = Enable 2D Graphic Controller clock |
| [9] | **USBH** | **USB Clock Enable Bit**<br>0 = Disable USB Host Controller clock<br>1 = Enable USB Host Controller clock |
| [8] | **USBD** | **USB Device Clock Enable Bit**<br>0 = Disable USB Device Controller clock<br>1 = Enable USB Device Controller clock |
| [7] | **EMC** | **EMC Clock Enable Bit**<br>0 = Disable EMC Controller clock<br>1 = Enable EMC Controller clock |
| [6] | **Reserved** | **Reserved, write 0 is recommended.** |
| [5] | **DMAC** | **DMAC Clock Enable Bit**<br>0 = Disable DMAC Controller clock<br>1 = Enable DMAC Controller clock |
| [4] | **FMI** | **FMI Clock Enable Bit**<br>0 = Disable FMI Controller clock<br>1 = Enable FMI Controller clock |
| [3:2] | **Reserved** | **Reserved, write 2'b00 is recommended.** |
| [1] | **Audio** | **Audio Controller Clock Enable Bit**<br>0 = Disable Audio Controller clock<br>1 = Enable Audio Controller clock |
| [0] | **LCD** | **LCD Clock Enable Bit**<br>0 = Disable LCD Controller clock<br>1 = Enable LCD Controller clock |

## Clock Select Register (CLKSEL)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLKSEL | 0xB000_0204 | R/W | Clock Select Register | 0x0000_0FFX |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | MSDSEL |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MSDSEL | | | | RESERVED | | UART1SEL | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VCKSEL | | ACKSEL | | RESERVED | | CPUCKSEL | |

| Bits | Descriptions | |
|---|---|---|
| [16:12] | **MSDSEL** | **MS/SD Engine Clock Source Select Bit**<br>[16:15]<br>MSDSEL[16:15] / Clock Source: 0 0 PLL0 Clock; 0 1 PLL1 Clock; 1 0 EXTAL15M pin; 1 1 EXTAL15M pin (Default)<br>[14:12]<br>Selected PLL0 source divided from 1 to 8. |
| [9:8] | **UART1SEL** | **UART1 Clock Source Select Bit**<br>UART1SEL / Clock Source: 0 0 PLL0 Clock; 0 1 PLL1 Clock; 1 0 EXTAL15M pin; 1 1 EXTAL15M pin (Default) |

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[7:6]</td><td>VCKSEL</td><td>LCD Clock Source Select Bit<br><br>
<table>
<tr><th colspan="2">VCKSEL</th><th>Clock Source</th></tr>
<tr><td>0</td><td>0</td><td>PLL0 Clock</td></tr>
<tr><td>0</td><td>1</td><td>PLL1 Clock</td></tr>
<tr><td>1</td><td>0</td><td>VICLK pin</td></tr>
<tr><td>1</td><td>1</td><td>EXTAL15M pin (Default)</td></tr>
</table></td></tr>
<tr><td>[5:4]</td><td>ACKSEL</td><td>Audio Clock Source Select Bit<br><br>
<table>
<tr><th colspan="2">ACKSEL</th><th>Clock Source</th></tr>
<tr><td>0</td><td>0</td><td>PLL0 Clock</td></tr>
<tr><td>0</td><td>1</td><td>PLL1 Clock</td></tr>
<tr><td>1</td><td>0</td><td>I2S_BITCLK pin</td></tr>
<tr><td>1</td><td>1</td><td>EXTAL15M pin (Default)</td></tr>
</table></td></tr>
<tr><td>[1:0]</td><td>CPUCKSEL</td><td>CPU/AMBA Clock Source Select Bit<br>Default value is depended on power-on setting (Pin MA17)<br>
<table>
<tr><th colspan="2">CPUCKSEL</th><th>Clock Source</th></tr>
<tr><td>0</td><td>0</td><td>PLL0 Clock</td></tr>
<tr><td>0</td><td>1</td><td>PLL1 Clock</td></tr>
<tr><td>1</td><td>0</td><td>PLL0 /2 Clock</td></tr>
<tr><td>1</td><td>1</td><td>EXTAL15M pin</td></tr>
</table></td></tr>
</table>

## Clock Divider Control Register (CLKDIV)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLKDIV | 0xB000_0208 | R/W | Clock Divider Control Register | 0x0400_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | G2DDIV | RESERVED | | APBCKDIV | | AHBCKDIV | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | UART1DIV | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| VCKDIV | | | | ACKDIV | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | CPUCKDIV | | | |

| Bits | Descriptions | |
|---|---|---|
| [30] | **G2DDIV** | **G2D Clock Divider Control Register**<br>0: divider 2<br>1: divider 1 |
| [29:28] | **RESERVED** | |
| [27:26] | **APBCKDIV** | **AMBA APB Clock Divider Control Register** |

| APBCKDIV | | Clock Frequency |
|---|---|---|
| 0 | 0 | Reserved |
| 0 | 1 | AHBCLK/2 |
| 1 | 0 | AHBCLK/4 |
| 1 | 1 | AHBCLK/8 |

| Bits | Descriptions | |
|---|---|---|
| [25:24] | **AHBCKDIV** | **AMBA AHB Clock (AHBCLK) Divider Control Register** |

| AHBCKDIV | | Clock Frequency |
|---|---|---|
| 0 | 0 | CPUCLK/1 |
| 0 | 1 | CPUCLK/2 |
| 1 | 0 | CPUCLK/4 |
| 1 | 1 | CPUCLK/8 |

| Bits | Descriptions | |
|---|---|---|
| [19:16] | **UART1DIV** | **UART1 Clock Source Divider Control Register**<br><br>**UART1CK = UART1 clock / (UART1DIV +1)**<br>Where (1) UART1DIV is 0~15<br>(2) UART1 clock is the clock source output by UART1SEL control reg. |
| [15:12] | **VCKDIV** | **LCD Clock Source Divider Control Register**<br><br>**LCD_CLK = VCK clock / (VCKDIV +1)**<br>Where (1) VCKDIV is 0~15<br>(2) VCK clock is the clock source output by VCKSEL control register |
| [11:8] | **ACKDIV** | **Audio Clock Source Divider Control Register**<br><br>**Audio_CLK = ACK clock / (ACKDIV +1)**<br>Where (1) ACKDIV is 0~15<br>(2) ACK clock is the clock source output by ACKSEL control register |
| [3:0] | **CPUCKDIV** | **CPU Clock Source Divider Control Register**<br><br>**CPUCLK = CCK clock / (CPUCKDIV +1)**<br>Where (1) CPUCKDIV is 0~15<br>(2) CCK clock is the clock source output by CPUCKSEL control register |

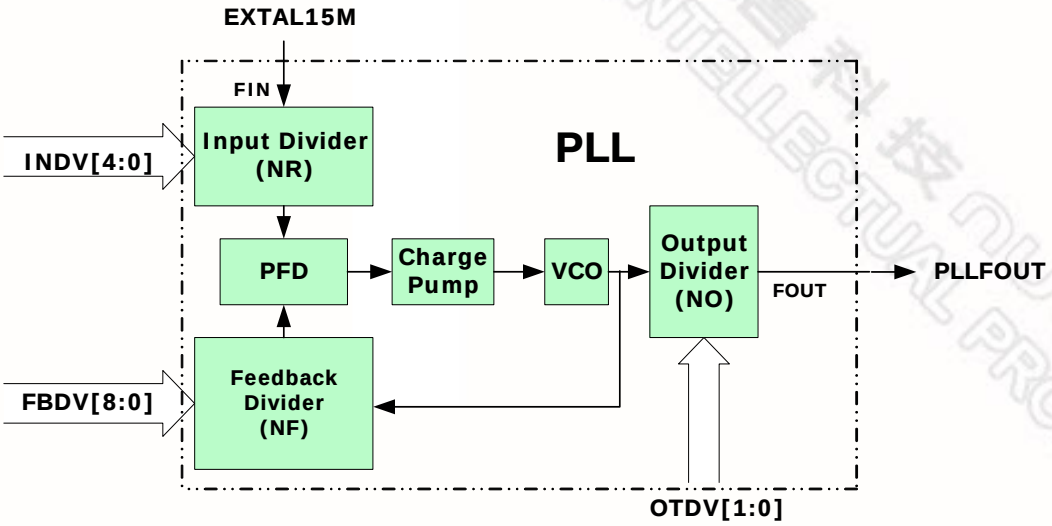
### PLL Control Register 0 (PLLCON0)

### PLL Control Register 1 (PLLCON1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| PLLCON0 | 0xB000_020C | R/W | PLL Control Register 0 | 0x0000_2B63 |
| PLLCON1 | 0xB000_0210 | R/W | PLL Control Register 1 | 0x0001_4F64 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | PWDEN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FBDV | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FBDV | OTDV | | INDV | | | | |

| Bits | Descriptions | |
|---|---|---|
| [16] | **PWDEN** | **Power Down Mode Enable**<br>0 = PLL is in normal mode (default)<br>1 = PLL is in power down mode |
| [15:7] | **FBDV** | **PLL VCO Output Clock Feedback Divider**<br>Feedback Divider divides the output clock from VCO of PLL. |
| [6:5] | **OTDV** | **PLL Output Clock Divider**<br>OTDV: 0 0 → Divided by 1<br>OTDV: 0 1 → Divided by 2<br>OTDV: 1 0 → Divided by 2<br>OTDV: 1 1 → Divided by 4 |
| [4:0] | **INDV** | **PLL Input Clock Divider**<br>Input Divider divides the input reference clock into the PLL. |




The formula of output clock of PLL is:

$$FOUT = FIN * \frac{NF}{NR} * \frac{1}{NO}$$

FOUT : Output clock of **Output Divider**

FIN : External clock into the **Input Divider**

NR : Input divider value (NR = INDV + 2)

NF : Feedback divider value (NF = FBDV + 2)

NO : Output divider value (NO = OTDV)

**Example Case:**

**The input clock frequency of EXTAL15M pin is 15MHz**

| PLL Output Frequency | 200MHz | 166MHz | 133MHz | 100MHz |
|---|---|---|---|---|
| PLLCON Reg. | 0x0000_4F24 | 0x0000_4124 | 0x0000_22A2 | 0x0000_4F64 |

| | | | | |
|---|---|---|---|---|
| **PLL Output Frequency** | **66MHz** | **169.34MHz (44.1K*3840)** | **122.88MHz (48K*2560)** | |
| **PLLCON Reg.** | **0x0000_2B63** | **0x0000_4E25** | **0x0000_92E7** | |

## Power Management Control Register (PMCON)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| PMCON | 0xB000_0214 | R/W | Power Management Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESET | MIDLE | PD | IDLE |

| Bits | Descriptions | |
|---|---|---|
| [3] | **RESET** | **Software Reset**<br><br>This is a software reset control bit. Set logic 1 to generate an internal reset pulse. This bit is auto-clear to logic 0 at the end of the reset pulse. |
| [2] | **MIDLE** | **Memory Controller IDLE enable**<br><br>Setting this bit HIGH to enable memory controller enter IDLE mode, the clock source of memory controller will be halted while ARM CORE enter IDLE mode.<br>1 = Memory controller will enter IDLE mode when IDLE bit is set.<br>0 = Memory controller still active when IDLE bit is set. |
| [1] | **PD** | **Power Down Enable**<br><br>Setting this bit HIGH, this chip enters power saving mode. The clock source 15M crystal oscillator and PLL both will stop to generate clock. User can use nIRQ [2:0], USB device, Keypad and external nRESET to wakeup chip.<br>1 = Power down mode enable<br>0 = Normal mode |

| Bits | Descriptions | |
|---|---|---|
| [0] | **IDLE** | **CPU IDLE mode Enable**<br><br>Setting this bit HIGH, ARM CPU Core enters power saving mode. The peripherals still working if the clock enable bit in **CLKSEL** is set. Any nIRQ or nFIQ to ARM core will let ARM core to exit IDLE state.<br>1 = CPU IDLE mode enable<br>0 = Normal mode |

## IRQ Wakeup Control Register (IRQWAKECON)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| IRQWAKECON | 0xB000_0218 | R/W | IRQ Wakeup Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | IRQWAKEUPPOL0 | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | IRQWAKEUPEN0 | | |

| Bits | Descriptions | |
|---|---|---|
| [10:8] | **IRQWAKEUPPOL0** | **Wakeup Polarity for nIRQ[2:0]**<br>1 = nIRQx is high level wakeup<br>0 = nIRQx is low level wakeup |
| [2:0] | **IRQWAKEUPEN0** | **Wakeup Enable for nIRQ[2:0]**<br>1 = nIRQx wakeup enable<br>0 = nIRQx wakeup disable |

The reserved bit has to keep on logical 0.

## IRQ Wakeup Flag Register (IRQWAKEFLAG)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| IRQWAKEFLAG | 0xB000_021C | R/W | IRQ Wakeup Flag Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | IRQWAKEFLAG | | |

| Bits | Descriptions | |
|---|---|---|
| [2:0] | **IRQWAKEFLAG** | **Wakeup Flag for nIRQ[2:0]**<br><br>After power down wakeup, software should check these flags to identify which IRQ is used to wakeup the system. **And clear the flags in IRQ interrupt service routine.**<br>1 = CPU is wakeup by nIRQx<br>0 = not wakeup |

## IP Software Reset Register (IPSRST)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| IPSRST | 0xB000_0220 | W | IP Software Reset Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | I2C | USI | RESERVED | | | KPI | RESERVED |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | TIMER | PWM | RESERVED | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | UART | G2D | USBH | USBD |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EMC | RESERVED | DMAC | FMI | GDMA | RESERVED | Audio | LCD |

| Bits | Descriptions | |
|---|---|---|
| [30] | **I2C** | **I2C Interface Software Reset Control Bit**<br>0 = write 0 is no action for both I2C0 and I2C1<br>1 = write 1 , a reset pulse is generated to reset both I2C0 and I2C1, and<br>This bit will be auto clear to zero. |
| [29] | **USI** | **USI Software Reset Control Bit**<br>0 = write 0 is no action for USI<br>1 = write 1 , a reset pulse is generated to reset USI, and<br>This bit will be auto clear to zero. |
| [25] | **KPI** | **Keypad Software Reset Control Bit**<br>0 = write 0 is no action for Keypad Controller<br>1 = write 1 , a reset pulse is generated to reset Keypad Controller, and<br>This bit will be auto clear to zero. |
| [19] | **TIMER** | **Timer Software Reset Control Bit**<br>0 = write 0 is no action for all of TIMERs and WDT<br>1 = write 1 , a reset pulse is generated to reset all of TIMERs and WDT, and<br>This bit will be auto clear to zero. |

| Bits | Descriptions | |
|---|---|---|
| [18] | **PWM** | **PWM Software Reset Control Bit**<br>0 = write 0 is no action for PWM Controller<br>1 = write 1 , a reset pulse is generated to reset PWM Controller, and<br>This bit will be auto clear to zero. |
| [11] | **UART** | **UART Software Reset Control Bit**<br>0 = write 0 is no action for all of UARTs<br>1 = write 1 , a reset pulse is generated to reset all of UARTs, and<br>This bit will be auto clear to zero. |
| [10] | **G2D** | **2D Graphic Controller Software Reset Control Bit**<br>0 = write 0 is no action for 2D graphic Controller<br>1 = write 1 , a reset pulse is generated to reset 2D Graphic Controller, and<br>This bit will be auto clear to zero. |
| [9] | **USBH** | **USB Software Reset Control Bit**<br>0 = write 0 is no action for USB Host Controller<br>1 = write 1 , a reset pulse is generated to reset USB Host Controller, and<br>This bit will be auto clear to zero. |
| [8] | **USBD** | **USB Device Software Reset Control Bit**<br>0 = write 0 is no action for USB Device Controller<br>1 = write 1 , a reset pulse is generated to reset USB Device Controller, and<br>This bit will be auto clear to zero. |
| [7] | **EMC** | **EMC Software Reset Control Bit**<br>0 = write 0 is no action for EMC Controller<br>1 = write 1 , a reset pulse is generated to reset EMC Controller, and<br>This bit will be auto clear to zero. |
| [5] | **DMAC** | **DMAC Software Reset Control Bit**<br>0 = write 0 is no action for DMA Controller<br>1 = write 1 , a reset pulse is generated to reset DMA Controller, and<br>This bit will be auto clear to zero. |

| Bits | Descriptions | |
|---|---|---|
| [4] | **FMI** | **FMI Software Reset Control Bit**<br>0 = write 0 is no action for FMI Controller<br>1 = write 1 , a reset pulse is generated to reset FMI Controller, and<br>This bit will be auto clear to zero. |
| [3] | **GDMA** | **GDMA Software Reset Control Bit**<br>0 = write 0 is no action for GDMA Controller<br>1 = write 1 , a reset pulse is generated to reset GDMA Controller, and<br>This bit will be auto clear to zero. |
| [1] | **Audio** | **Audio Controller Software Reset Control Bit**<br>0 = write 0 is no action for Audio Controller<br>1 = write 1 , a reset pulse is generated to reset Audio Controller, and<br>This bit will be auto clear to zero. |
| [0] | **LCD** | **LCD Controller Software Reset Control Bit**<br>0 = write 0 is no action for LCD Controller<br>1 = write 1 , a reset pulse is generated to reset LCD Controller, and<br>This bit will be auto clear to zero. |

## Clock Enable 1 Register (CLKEN1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLKEN1 | 0xB000_0224 | R/W | Clock Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | RMII | SD | MS |

| Bits | Descriptions | |
|---|---|---|
| [2] | **RMII** | **RMII Clock Enable Bit**<br>0 = Disable RMII clock<br>1 = Enable RMII clock |
| [1] | **SD** | **SD Clock Enable Bit**<br>0 = Disable SD clock<br>1 = Enable SD clock |
| [0] | **MS** | **MS Clock Enable Bit**<br>0 = Disable MS clock<br>1 = Enable MS clock |

### Clock Divider Control 1 Register (CLKDIV1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLKDIV1 | 0xB000_0228 | R/W | Clock Divider Control 1 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SD_DIV | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MS_DIV | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:8] | **SD_DIV** | **SD divider**<br><br>**SD_CLK = Source Clock / (SD_DIV +1)**<br>Where Source Clock selection is controlled by MSDSEL of register CLKSEL. |
| [7:0] | **MS_DIV** | **MS divider**<br><br>**MS_CLK = Source Clock / (MS_DIV +1)**<br>Where Source Clock selection is controlled by MSDSEL of register CLKSEL. |

## 7.4 External Bus Interface

### 7.4.1 Overview

This chip supports External Bus Interface (**EBI**), which controls the access to the external memory

(ROM/FLASH, SDRAM) and External I/O devices. The **EBI** has chip select signals to select one ROM/FLASH bank, two SDRAM banks, and three External I/O banks with 22-bit address bus. It supports 8-bit, 16-bit, and 32-bit external data bus width for each bank.

The EBI has the following functions：

- SDRAM controller
- EBI control register
- ROM/FLASH interface
- External I/O interface

### 7.4.2 Functional Description

**SDRAM Controller**

The SDRAM controller module contains configuration registers, timing control registers, common control register and other logic to provide 8, 16 or 32 bits SDRAM interface with a single 8, 16 or 32 bits SDRAM device or two 8-bit devices wired to give a 16-bit data path or two 16-bit devices wired to give a 32-bit data path.

The SDRAM controller has the following features：

- Supports up to 2 external SDRAM banks
- Maximum size of each SDRAM bank is 128M bytes
- 8, 16 or 32-bit data interface
- Programmable CAS Latency： 1、2 and 3
- Fixed Burst Length： 1
- Sequential burst type
- Write Burst Length mode is Burst
- Auto Refresh Mode and Self Refresh Mode
- Adjustable Refresh Rate
- Power up sequence

#### 7.4.2.2 SDRAM Components Supported

Table: SDRAM Components supported

| Size | Type | Banks | Row Addressing | Column Addressing |
|---|---|---|---|---|
| 16M bits | 2Mx8 | 2 | RA0~RA10 | CA0~CA8 |
| | 1Mx16 | 2 | RA0~RA10 | CA0~CA7 |
| 64M bits | 8Mx8 | 4 | RA0~RA11 | CA0~CA8 |
| | 4Mx16 | 4 | RA0~RA11 | CA0~CA7 |
| | 2Mx32 | 4 | RA0~RA10 | CA0~CA7 |
| 128M bits | 16Mx8 | 4 | RA0~RA11 | CA0~CA9 |
| | 8Mx16 | 4 | RA0~RA11 | CA0~CA8 |
| | 4Mx32 | 4 | RA0~RA11 | CA0~CA7 |
| 256M bits | 32Mx8 | 4 | RA0~RA12 | CA0~CA9 |
| | 16Mx16 | 4 | RA0~RA12 | CA0~CA8 |
| 512M bits | 64Mx8 | 4 | RA0~RA12 | CA0~CA9,CA11 |
| | 32Mx16 | 4 | RA0~RA12 | CA0~CA9 |

### AHB Bus Address Mapping to SDRAM Bus

Note: * indicates the signal is not used; ** indicates the signal is fixed at logic 0 and is not used;
The HADDR prefixes have been omitted on the following tables.
MA14 ~ MA0 are the Address pins of the EBI interface;
MA14 and MA13 are also the bank selected signals of SDRAM.

**SDRAM Data Bus Width: 32-bit**

| Total | Type | R x C | R/C | MA14 (BS1) | MA13 (BS0) | MA12 | MA11 | MA10 | MA9 | MA8 | MA7 | MA6 | MA5 | MA4 | MA3 | MA2 | MA1 | MA0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16M | 2Mx8 | 11x9 | R | ** | 11 | ** | 11* | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 |
| | | | C | ** | 11 | ** | 11* | AP | 25* | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 16M | 1Mx16 | 11x8 | R | ** | 10 | ** | 10* | 11 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 |
| | | | C | ** | 10 | ** | 10* | AP | 25* | 10* | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 64M | 8Mx8 | 12x9 | R | 11 | 12 | 11* | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 24 |
| | | | C | 11 | 12 | 11* | 23* | AP | 25* | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 64M | 4Mx16 | 12x8 | R | 11 | 10 | 11* | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 |
| | | | C | 11 | 10 | 11* | 23* | AP | 25* | 24* | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 64M | 2Mx32 | 11x8 | R | 11 | 10 | 11* | 23* | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 |
| | | | C | 11 | 10 | 11* | 23* | AP | 25* | 24* | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 128M* | 16Mx8 | 12x10 | R | 11 | 12 | 11* | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 24 |
| | | | C | 11 | 12 | 11* | 23* | AP | 25 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 128M | 8Mx16 | 12x9 | R | 11 | 12 | 11* | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 24 |
| | | | C | 11 | 12 | 11* | 23* | AP | 25* | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 128M | 4Mx32 | 12x8 | R | 11 | 10 | 11* | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 |
| | | | C | 11 | 10 | 11* | 23* | AP | 25* | 10* | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 256M* | 32Mx8 | 13x10 | R | 11 | 12 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 25 |
| | | | C | 11 | 12 | 24* | 23* | AP | 26 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 256M* | 16Mx16 | 13x9 | R | 11 | 12 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 25 |
| | | | C | 11 | 12 | 24* | 23* | AP | 26* | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 512M* | 64Mx8 | 13x11 | R | 11 | 12 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 25 |
| | | | C | 11 | 12 | 24* | 27 | AP | 26 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 512M* | 32Mx16 | 13x10 | R | 11 | 12 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 25 |
| | | | C | 11 | 12 | 24* | 23* | AP | 26 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |

**SDRAM Data Bus Width: 16-bit**

| Total | Type | R x C | R/C | MA14 (BS1) | MA13 (BS0) | MA12 | MA11 | MA10 | MA9 | MA8 | MA7 | MA6 | MA5 | MA4 | MA3 | MA2 | MA1 | MA0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16M | 2Mx8 | 11x9 | R | ** | 10 | ** | 10* | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 |
| | | | C | ** | 10 | ** | 10* | AP | 24* | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 16M | 1Mx16 | 11x8 | R | ** | 9 | ** | 9* | 10 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 |
| | | | C | ** | 9 | ** | 9* | AP | 24* | 9* | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 64M | 8Mx8 | 12x9 | R | 10 | 11 | 10* | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 23 |
| | | | C | 10 | 11 | 10* | 22* | AP | 24* | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 64M | 4Mx16 | 12x8 | R | 10 | 9 | 10* | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 |
| | | | C | 10 | 9 | 10* | 22* | AP | 24* | 23* | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 64M | 2Mx32 | 11x8 | R | 10 | 9 | 10* | 22* | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 |
| | | | C | 10 | 9 | 10* | 22* | AP | 24* | 23* | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 128M | 16Mx8 | 12x10 | R | 10 | 11 | 10* | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 23 |
| | | | C | 10 | 11 | 10* | 22* | AP | 24 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 128M | 8Mx16 | 12x9 | R | 10 | 11 | 10* | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 23 |
| | | | C | 10 | 11 | 10* | 22* | AP | 24* | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 128M | 4Mx32 | 12x8 | R | 10 | 9 | 10* | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 |
| | | | C | 10 | 9 | 10* | 22* | AP | 24* | 9* | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 256M* | 32Mx8 | 13x10 | R | 10 | 11 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 24 |
| | | | C | 10 | 11 | 23* | 22* | AP | 25 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 256M | 16Mx16 | 13x9 | R | 10 | 11 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 24 |
| | | | C | 10 | 11 | 23* | 22* | AP | 25* | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 512M | 64Mx8 | 13x11 | R | 10 | 11 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 24 |
| | | | C | 10 | 11 | 23* | 26 | AP | 25 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 512M | 32Mx16 | 13x10 | R | 10 | 11 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 24 |
| | | | C | 10 | 11 | 23* | 22* | AP | 25 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

**SDRAM Data Bus Width: 8-bit**

| Total | Type | R x C | R/C | MA14 (BS1) | MA13 (BS0) | MA12 | MA11 | MA10 | MA9 | MA8 | MA7 | MA6 | MA5 | MA4 | MA3 | MA2 | MA1 | MA0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16M | 2Mx8 | 11x9 | R | ** | 9 | ** | 9* | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
| | | | C | ** | **9** | ** | 9* | AP | 23* | **8** | **7** | **6** | **5** | **4** | **3** | 2 | 1 | 0 |
| 16M | 1Mx16 | 11x8 | R | ** | 8 | ** | 8* | **9** | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
| | | | C | ** | 8 | ** | 8* | AP | 23* | 8* | **7** | **6** | **5** | **4** | **3** | 2 | 1 | 0 |
| 64M | 8Mx8 | 12x9 | R | 9 | 10 | 9* | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | **22** |
| | | | C | 9 | 10 | 9* | 21* | AP | 23* | **8** | **7** | **6** | **5** | **4** | **3** | 2 | 1 | 1 |
| 64M | 4Mx16 | 12x8 | R | 9 | 8 | 9* | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
| | | | C | 9 | 8 | 9* | 21* | AP | 23* | 22* | **7** | **6** | **5** | **4** | **3** | 2 | 1 | 0 |
| 64M | 2Mx32 | 11x8 | R | 9 | 8 | 9* | 21* | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
| | | | C | 9 | 8 | 9* | 21* | AP | 23* | 22* | **7** | **6** | **5** | **4** | **3** | 2 | 1 | 0 |
| 128M | 16Mx8 | 12x10 | R | 9 | 10 | 9* | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | **22** |
| | | | C | 9 | 10 | 9* | 21* | AP | **23** | **8** | **7** | **6** | **5** | **4** | **3** | 2 | 1 | 0 |
| 128M | 8Mx16 | 12x9 | R | 9 | 10 | 9* | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | **22** |
| | | | C | 9 | 10 | 9* | 21* | AP | 23* | **8** | **7** | **6** | **5** | **4** | **3** | 2 | 1 | 0 |
| 128M | 4Mx32 | 12x8 | R | 9 | 8 | 9* | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
| | | | C | 9 | 8 | 9* | 21* | AP | 23* | 8* | **7** | **6** | **5** | **4** | **3** | 2 | 1 | 0 |
| 256M | 32Mx8 | 13x10 | R | 9 | 10 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | **23** |
| | | | C | 9 | 10 | 22* | 21* | AP | 24 | **8** | **7** | **6** | **5** | **4** | **3** | 2 | 1 | 0 |
| 256M | 16Mx16 | 13x9 | R | 9 | 10 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | **23** |
| | | | C | 9 | 10 | 22* | 21* | AP | 24* | 8 | **7** | **6** | **5** | **4** | **3** | 2 | 1 | 0 |
| 512M | 64Mx8 | 13x11 | R | 9 | 10 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | **23** |
| | | | C | 9 | 10 | 22* | 25 | AP | 24 | 8 | **7** | **6** | **5** | **4** | **3** | 2 | 1 | 0 |
| 512M | 32Mx16 | 13x10 | R | 9 | 10 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | **23** |
| | | | C | 9 | 10 | 22* | 21* | AP | 24 | 8 | **7** | **6** | **5** | **4** | **3** | 2 | 1 | 0 |

**SDRAM Power-Up Sequence**

The default value of the mode register is not defined, therefore the mode register must be written after power up to operate the SDRAM. This chip supports the function of Power-Up Sequence, that is, after system power on, the SDRAM Controller automatically executes the commands needed for Power-Up sequence and set the mode register of each bank to default value. The default value is :

- Burst Length = 1
- Burst Type = Sequential (fixed)
- CAS Latency = 2
- Write Burst Length = Burst (fixed)

The value of mode register can be changed after power up sequence by setting the value of corresponding

bank's configuration register "**LENGTH**" and "**LATENCY**" bits and set the **MRSET** bit enable to execute the Mode Register Set command.

### 7.4.3 EBI Register Mapping

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **(EBI_BA=0xB000_1000)** | | | | |
| **EBICON** | 0xB000_1000 | R/W | EBI control register | 0x0001_0001 |
| **ROMCON** | 0xB000_1004 | R/W | ROM/FLASH control register | 0x0000_0FFX |
| **SDCONF0** | 0xB000_1008 | R/W | SDRAM bank 0 configuration register | 0x0000_0800 |
| **SDCONF1** | 0xB000_100C | R/W | SDRAM bank 1 configuration register | 0x0000_0800 |
| **SDTIME0** | 0xB000_1010 | R/W | SDRAM bank 0 timing control register | 0x0000_0000 |
| **SDTIME1** | 0xB000_1014 | R/W | SDRAM bank 1 timing control register | 0x0000_0000 |
| **EXT0CON** | 0xB000_1018 | R/W | External I/O 0 control register | 0x0000_0000 |
| **EXT1CON** | 0xB000_101C | R/W | External I/O 1 control register | 0x0000_0000 |
| **EXT2CON** | 0xB000_1020 | R/W | External I/O 2 control register | 0x0000_0000 |
| **CKSKEW** | 0xB000_102C | R/W | Clock skew control register (for testing) | 0xXXXX_0048 |

### 7.4.4 EBI Register Details

## EBI Control Register (EBICON)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| EBICON | 0xB000_1000 | R/W | EBI Control Register | 0x0001_0001 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | EXBE2 | EXBE1 | EXBE0 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | REFEN | REFMOD | CLKEN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| REFRAT | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| REFRAT | | | | | WAITVT | | LITTLE |

| Bits | Descriptions | |
|---|---|---|
| [26] | **EXBE2** | **EXBE2: External IO Bank 2 Byte Enable**<br>0: nWBE[3:0] pin is byte write strobe signal<br>1: nWBE[3:0] pin is byte enable signals, nSWE will be used as write strobe signal to SRAM |
| [25] | **EXBE1** | **EXBE1: External IO Bank 1 Byte Enable**<br>0: nWBE[3:0] pin is byte write strobe signal<br>1: nWBE[3:0] pin is byte enable signals, nSWE will be used as write strobe signal to SRAM |
| [24] | **EXBE0** | **EXBE0: External IO Bank 0 Byte Enable**<br>0: nWBE[3:0] pin is byte write strobe signal<br>1: nWBE[3:0] pin is byte enable signals, nSWE will be used as write strobe signal to SRAM |
| [23:19] | **Reserved** | Write 0 for normal operation |
| [18] | **REFEN** | **Enable SDRAM refresh cycle for SDRAM bank0 & bank1**<br>This bit set will start the auto-refresh cycle to SDRAM. The refresh rate is according to REFRAT bits. |

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[17]</td><td>REFMOD</td><td>The refresh mode of SDRAM for SDRAM bank<br>Defines the refresh mode type of external SDRAM bank<br>0 = Auto refresh mode<br>1 = Self refresh mode</td></tr>
<tr><td>[16]</td><td>CLKEN</td><td>Clock enable for SDRAM<br>Enables the SDRAM clock enable (CKE) control signal<br>0 = Disable (power down mode)<br>1 = Enable （Default）</td></tr>
<tr><td>[15:3]</td><td>REFRAT</td><td>Refresh count value for SDRAM<br>The refresh period is calculated as $period = \frac{value}{fMCLK}$<br>The SDRAM Controller automatically provides an auto refresh cycle for every refresh period programmed into the REFRAT bits when the REFEN bit of each bank is set.</td></tr>
<tr><td>[2:1]</td><td>WAITVT</td><td>Valid time of nWAIT signal<br>This bit recognizes the nEWAIT signal at the next “nth” MCLK rising edge after the nOE or nWBE active cycle. WAITVT bits determine the n.
<table>
<tr><th colspan="2">WAITVT [2:1]</th><th>nth MCLK</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>2</td></tr>
<tr><td>1</td><td>0</td><td>3</td></tr>
<tr><td>1</td><td>1</td><td>4</td></tr>
</table></td></tr>
<tr><td>[0]</td><td>LITTLE</td><td>Little Endian mode<br>This bit always set to a logic 1 (Read Only)</td></tr>
</table>

## ROM/Flash Control Register (ROMCON)

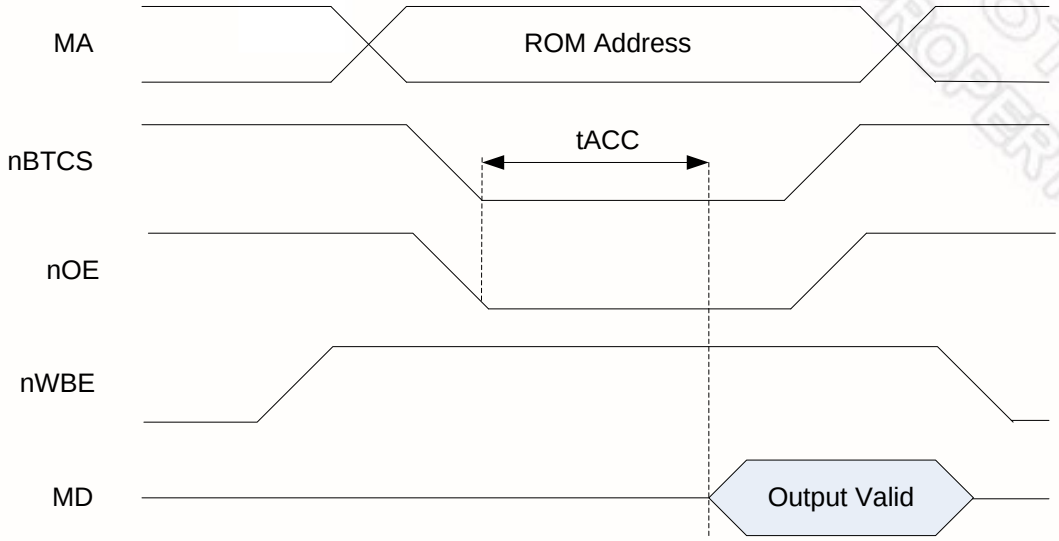
| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ROMCON | 0xB000_1004 | R/W | ROM/FLASH Control Register | 0x0000_0FFX |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| BASADDR | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BASADDR | | | | | SIZE | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SIZE | Reserved | | | tPA | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| tACC | | | | BTSIZE | | PGMODE | |

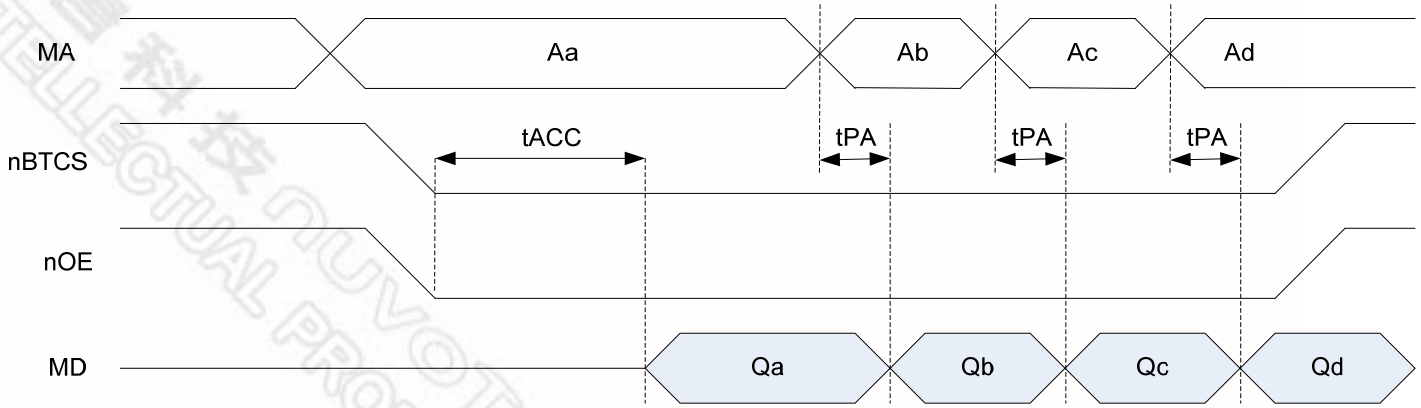
| Bits | Descriptions | |
|---|---|---|
| [31:19] | **BASADDR** | **Base Address Pointer of ROM/Flash Bank**<br>The start address is calculated as ROM/Flash bank base pointer << 18. The base address pointer together with the "**SIZE**" bits constitutes the whole address range of each bank. |
| [18:15] | **SIZE** | **Size of ROM/FLASH Memory**<br>.<br>**SIZE [18:15]** / **Byte**<br>0 0 0 0 — 256K<br>0 0 1 0 — 512K<br>0 1 0 0 — 1M<br>0 1 1 0 — 2M<br>1 0 0 0 — 4M<br>1 0 1 0 — 8M<br>1 1 0 0 — 16M<br>Others — Reserved |

<table>
<tr><th>Bits</th><th>Descriptions</th></tr>
<tr><td>[11:8]</td><td>tPA</td><td>Page Mode Access Cycle Time
<table>
<tr><th colspan="4">tPA[11:8]</th><th>MCLK</th><th colspan="4">tPA[11:8]</th><th>MCLK</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>10</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>2</td><td>1</td><td>0</td><td>0</td><td>1</td><td>12</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>3</td><td>1</td><td>0</td><td>1</td><td>0</td><td>14</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>4</td><td>1</td><td>0</td><td>1</td><td>1</td><td>16</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>5</td><td>1</td><td>1</td><td>0</td><td>0</td><td>18</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>6</td><td>1</td><td>1</td><td>0</td><td>1</td><td>20</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>7</td><td>1</td><td>1</td><td>1</td><td>0</td><td>22</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>8</td><td>1</td><td>1</td><td>1</td><td>1</td><td>24</td></tr>
</table></td></tr>
<tr><td>[7:4]</td><td>tACC</td><td>Access Cycle Time
<table>
<tr><th colspan="4">tACC[7:4]</th><th>MCLK</th><th colspan="4">tACC[7:4]</th><th>MCLK</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>3</td><td>1</td><td>0</td><td>0</td><td>0</td><td>10</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>3</td><td>1</td><td>0</td><td>0</td><td>1</td><td>12</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>3</td><td>1</td><td>0</td><td>1</td><td>0</td><td>14</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>4</td><td>1</td><td>0</td><td>1</td><td>1</td><td>16</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>5</td><td>1</td><td>1</td><td>0</td><td>0</td><td>18</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>6</td><td>1</td><td>1</td><td>0</td><td>1</td><td>20</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>7</td><td>1</td><td>1</td><td>1</td><td>0</td><td>22</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>8</td><td>1</td><td>1</td><td>1</td><td>1</td><td>24</td></tr>
</table></td></tr>
<tr><td>[3:2]</td><td>BTSIZE</td><td>Boot ROM/FLASH Data Bus Width<br>This ROM/Flash bank is designed for a boot ROM. BASADDR bits determine its start address. The external data bus width is determined by power-on setting when booting from external ROM.
<table>
<tr><th colspan="2">BTSIZE [3:2]</th><th>Bus Width</th></tr>
<tr><td>0</td><td>0</td><td>8-bit</td></tr>
<tr><td>0</td><td>1</td><td>16-bit</td></tr>
<tr><td>1</td><td>0</td><td>32-bit</td></tr>
<tr><td>1</td><td>1</td><td>RESERVED</td></tr>
</table></td></tr>
</table>

| Bits | Descriptions | |
|---|---|---|
| [1:0] | **PGMODE** | **Page Mode Configuration**<br>**PGMODE [1:0]** / **Mode**<br>0 0 — Normal ROM<br>0 1 — 4 word page<br>1 0 — 8 word page<br>1 1 — 16 word page |




**ROM/FLASH Read Operation Timing**




**ROM/FLASH Page Read Operation Timing**

## SDRAM Configuration Register (SDCONF0/1)

The configuration registers enable software to set a number of operating parameters for the SDRAM controller. There are two configuration registers SDCONF0 or SDCONF1 for SDRAM bank 0 or bank 1 respectively. Each bank can have a different configuration.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDCONF0 | 0xB000_1008 | R/W | SDRAM Bank 0 Configuration Register | 0x0000_0800 |
| SDCONF1 | 0xB000_100C | R/W | SDRAM Bank 1 Configuration Register | 0x0000_0800 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| BASADDR | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BASADDR | | | | | RESERVED | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MRSET | RESERVED | AUTOPR | LATENCY | | RESERVED | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| COMPBK | DBWD | | COLUMN | | SIZE | | |

| Bits | Descriptions | |
|---|---|---|
| [31:19] | **BASADDR** | **Base Address Pointer of SDRAM Bank 0/1**<br>The start address is calculated as SDRAM bank 0/1 base pointer << 18. The SDRAM base address pointer together with the "**SIZE**" bits constitutes the whole address range of each SDRAM bank. |
| [15] | **MRSET** | **SDRAM Mode Register Set Command for SDRAM Bank 0/1**<br>This bit set will issue a mode register set command to SDRAM. |
| [13] | **AUTOPR** | **Auto Pre-charge Mode of SDRAM for SDRAM Bank 0/1**<br>Enable the auto pre-charge function of external SDRAM bank 0/1<br>0 = Auto pre-charge<br>1 = No auto pre-charge |

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[12:11]</td><td>LATENCY</td><td>The CAS Latency of SDRAM Bank 0 / 1<br>Defines the CAS latency of external SDRAM bank 0/1
<table>
<tr><th colspan="2">LATENCY [12:11]</th><th>MCLK</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>2</td></tr>
<tr><td>1</td><td>0</td><td>3</td></tr>
<tr><td>1</td><td>1</td><td>REVERSED</td></tr>
</table></td></tr>
<tr><td>[7]</td><td>COMPBK</td><td>Number of Component Bank in SDRAM Bank 0 / 1<br>Indicates the number of component bank (2 or 4 banks) in external SDRAM bank 0/1.<br>0 = 2 banks<br>1 = 4 banks</td></tr>
<tr><td>[6:5]</td><td>DBWD</td><td>Data Bus Width for SDRAM Bank 0 / 1<br>Indicates the external data bus width connect with SDRAM bank 0/1<br>If DBWD = 00, the assigned SDRAM access signal is not generated i.e. disable.
<table>
<tr><th colspan="2">DBWD [6:5]</th><th>Bits</th></tr>
<tr><td>0</td><td>0</td><td>Bank disable</td></tr>
<tr><td>0</td><td>1</td><td>8-bit (byte)</td></tr>
<tr><td>1</td><td>0</td><td>16-bit (half-word)</td></tr>
<tr><td>1</td><td>1</td><td>32-bit (word)</td></tr>
</table></td></tr>
<tr><td>[4:3]</td><td>COLUMN</td><td>Number of Column Address bits in SDRAM Bank 0 / 1<br>Indicates the number of column address bits in external SDRAM bank 0/1.
<table>
<tr><th colspan="2">COLUMN [4:3]</th><th>Bits</th></tr>
<tr><td>0</td><td>0</td><td>8</td></tr>
<tr><td>0</td><td>1</td><td>9</td></tr>
<tr><td>1</td><td>0</td><td>10</td></tr>
<tr><td>1</td><td>1</td><td>11</td></tr>
</table></td></tr>
</table>

| Bits | Descriptions | |
|---|---|---|
| [2:0] | **SIZE** | **Size of SDRAM Bank 0 / 1**<br>Indicates the memory size of external SDRAM bank 0/1 |

| SIZE [2:0] | | | Size of SDRAM （Byte） |
|---|---|---|---|
| 0 | 0 | 0 | Bank disable |
| 0 | 0 | 1 | 2M |
| 0 | 1 | 0 | 4M |
| 0 | 1 | 1 | 8M |
| 1 | 0 | 0 | 16M |
| 1 | 0 | 1 | 32M |
| 1 | 1 | 0 | 64M |
| 1 | 1 | 1 | 128M |

## SDRAM Timing Control Register (SDTIME0/1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDTIME0 | 0xB000_1010 | R/W | SDRAM bank 0 timing control register | 0x0000_0000 |
| SDTIME1 | 0xB000_1014 | R/W | SDRAM bank 1 timing control register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | tRCD | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| tRDL | | tRP | | | tRAS | | |

| Bits | Descriptions | |
|---|---|---|
| [10:8] | tRCD | SDRAM Bank 0/1, /RAS to /CAS Delay |

| tRCD [10:8] | | | MCLK |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 3 |
| 0 | 1 | 1 | 4 |
| 1 | 0 | 0 | 5 |
| 1 | 0 | 1 | 6 |
| 1 | 1 | 0 | 7 |
| 1 | 1 | 1 | 8 |

| Bits | Descriptions | |
|---|---|---|
| [7:6] | tRDL | SDRAM Bank 0/1, Last Data in to Pre-charge Command |

| tRDL [7:6] | | MCLK |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 2 |
| 1 | 0 | 3 |
| 1 | 1 | 4 |

<table>
<tr><th>Bits</th><th>Descriptions</th><th></th></tr>
<tr><td>[5:3]</td><td>tRP</td><td>SDRAM Bank 0 / 1, Row Pre-charge Time
<table>
<tr><th colspan="3">tRP [5:3]</th><th>MCLK</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>3</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>4</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>5</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>6</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>7</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>8</td></tr>
</table></td></tr>
<tr><td>[2:0]</td><td>tRAS</td><td>SDRAM Bank 0 / 1, Row Active Time
<table>
<tr><th colspan="3">tRAS [2:0]</th><th>MCLK</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>3</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>4</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>5</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>6</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>7</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>8</td></tr>
</table></td></tr>
</table>

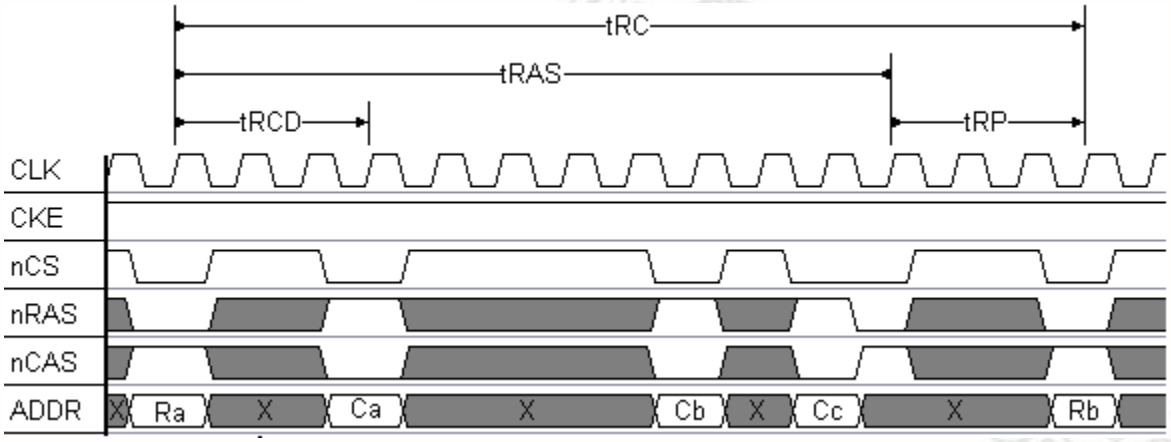



**Fig. Access timing 1 of SDRAM**

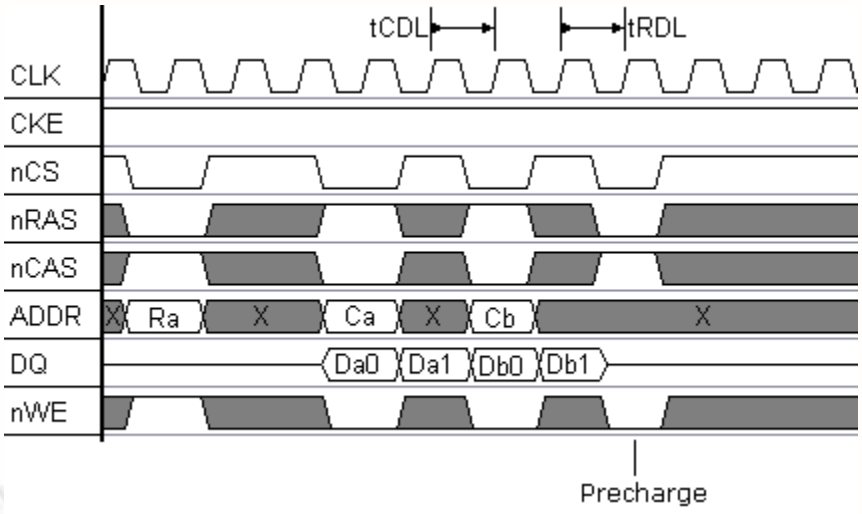



**Fig. Access timing 2 of SDRAM**

## External I/O Control Registers (EXT0CON – EXT2CON)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| EXT0CON | 0xB000_1018 | R/W | External I/O 0 control register | 0x0000_0000 |
| EXT1CON | 0XB000_101C | R/W | External I/O 1 control register | 0x0000_0000 |
| EXT3CON | 0xB000_1020 | R/W | External I/O 2 control register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| BASADDR | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BASADDR | | | | | SIZE | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ADRS | tACC | | | | tCOH | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| tACS | | | tCOS | | | DBWD | |

| Bits | Descriptions | |
|---|---|---|
| [31:19] | **BASADDR** | **Base Address Pointer of External I/O Bank 0~2**<br>The start address of each external I/O bank is calculated as "**BASADDR"** base pointer << 18. Each external I/O bank base address pointer together with the "**SIZE**" bits constitutes the whole address range of each external I/O bank. |
| [18:16] | **SIZE** | **The Size of the External I/O Bank 0~2** |

| SIZE [18:16] | | | Byte |
|---|---|---|---|
| 0 | 0 | 0 | 256K |
| 0 | 0 | 1 | 512K |
| 0 | 1 | 0 | 1M |
| 0 | 1 | 1 | 2M |
| 1 | 0 | 0 | 4M |
| 1 | 0 | 1 | 8M |
| 1 | 1 | 0 | 16M |
| 1 | 1 | 1 | Reserved |

<table>
<tr><th>Bits</th><th>Descriptions</th></tr>
<tr><td>[15]</td><td>ADRS</td><td>Address Bus Alignment for External I/O Bank 0~2<br>When ADRS is set, EBI bus is alignment to byte address format, and ignores DBWD [1:0] setting.</td></tr>
<tr><td>[14:11]</td><td>tACC</td><td>Access Cycles（nOE or nSWE active time）for External I/O Bank 0~2
<table>
<tr><th colspan="4">tACC[14:11]</th><th>MCLK</th><th colspan="4">tACC[14:11]</th><th>MCLK</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>Reversed</td><td>1</td><td>0</td><td>0</td><td>0</td><td>9</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>11</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>2</td><td>1</td><td>0</td><td>1</td><td>0</td><td>13</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>3</td><td>1</td><td>0</td><td>1</td><td>1</td><td>15</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>4</td><td>1</td><td>1</td><td>0</td><td>0</td><td>17</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>5</td><td>1</td><td>1</td><td>0</td><td>1</td><td>19</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>6</td><td>1</td><td>1</td><td>1</td><td>0</td><td>21</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>7</td><td>1</td><td>1</td><td>1</td><td>1</td><td>23</td></tr>
</table></td></tr>
<tr><td>[10:8]</td><td>tCOH</td><td>Chip Selection Hold-On Time on nOE or nWBE for External I/O Bank 0~2
<table>
<tr><th colspan="3">tCOH [10:8]</th><th>MCLK</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Bits</th><th>Descriptions</th><th></th></tr>
<tr><td>[7:5]</td><td>tACS</td><td>Address Set-up Before nECS for External I/O bank 0~2
<table>
<tr><th colspan="3">tACS [7:5]</th><th>MCLK</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7</td></tr>
</table></td></tr>
<tr><td>[4:2]</td><td>tCOS</td><td>Chip Selection Set-up Time on nOE or nWBE for External I/O Bank 0~2<br>When the bank is configured, the access to its bank stretches chip selection time before the nOE or new signal is activated.
<table>
<tr><th colspan="3">tCOS [4:2]</th><th>MCLK</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7</td></tr>
</table></td></tr>
<tr><td>[1:0]</td><td>DBWD</td><td>Programmable Data Bus Width for External I/O Bank 0~2
<table>
<tr><th colspan="2">DBWD [1:0]</th><th>Width of Data Bus</th></tr>
<tr><td>0</td><td>0</td><td>Disable bus</td></tr>
<tr><td>0</td><td>1</td><td>8-bit</td></tr>
<tr><td>1</td><td>0</td><td>16-bit</td></tr>
<tr><td>1</td><td>1</td><td>32-bit</td></tr>
</table></td></tr>
</table>

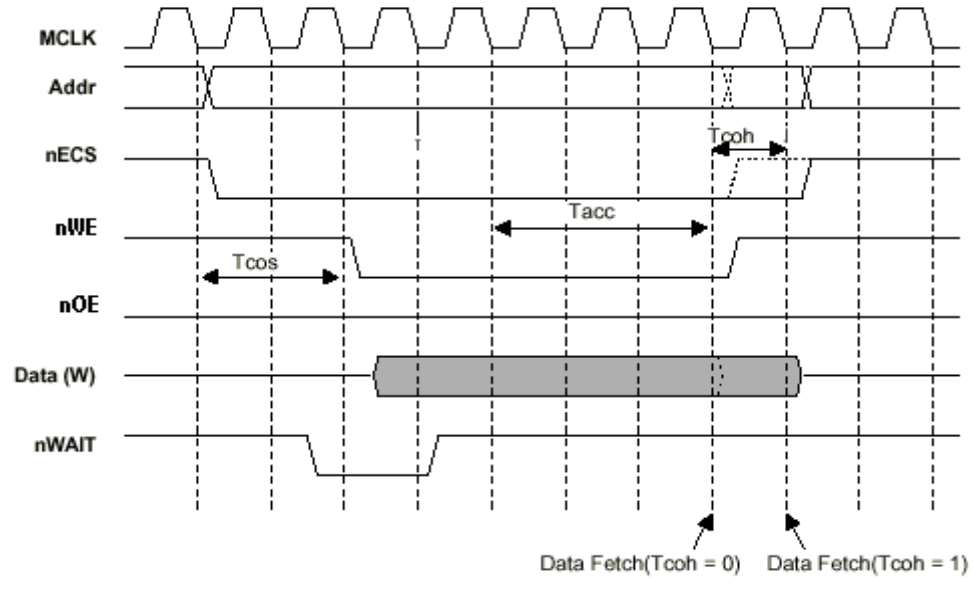



**Fig. External I/O Write operation timing**

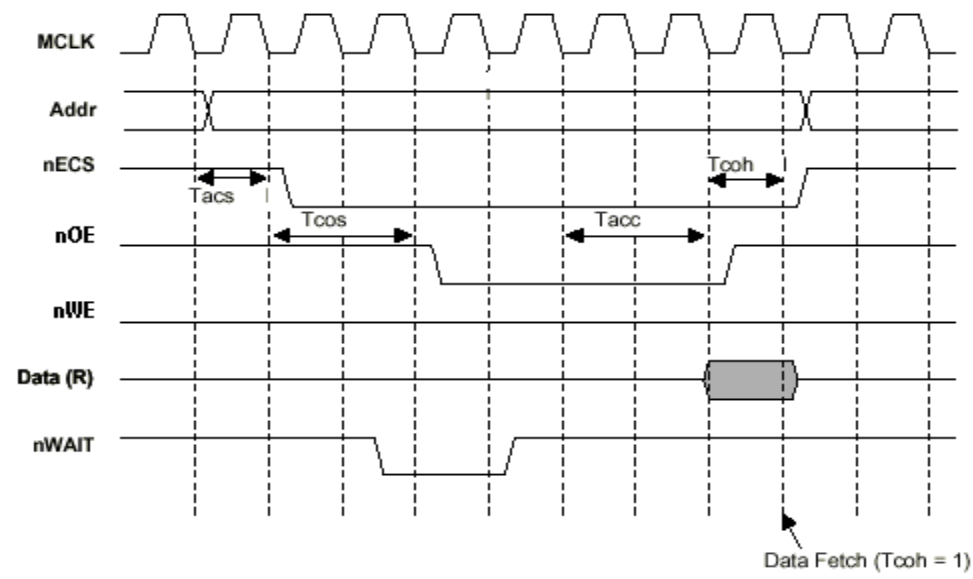



**Fig. External I/O Read operation timing**

### Clock Skew Control Register (CKSKEW)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CKSKEW | 0xB000_102C | R/W | Clock Skew Control Register | 0xXXXX_0048 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | SWPON |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DLH_CLK_SKEW | | | | MCLK_O_D | | | |

| Bits | Descriptions | |
|---|---|---|
| [8] | **SWPON** | **SDRAM Initialization by Software Trigger**<br>Set this bit will issue a SDRAM power on default setting command, this bit will be auto-clear by hardware |
| [7:4] | **DLH_CLK_SKEW** | **Data Latch Clock Skew Adjustment** |
| [3:0] | **MCLK_O_D** | **MCLK Output Delay Adjustment** |

## 7.5 Ethernet MAC Controller

### Overview

This chip provides an Ethernet MAC Controller (EMC) for WAN/LAN application. This EMC has its DMA controller, transmit FIFO, and receive FIFO.

The Ethernet MAC controller consists of IEEE 802.3/Ethernet protocol engine with internal CAM function for Ethernet MAC address recognition, Transmit-FIFO, Receive-FIFO, TX/RX state machine controller and status controller. The EMC only supports RMII (Reduced MII) interface to connect with PHY operating on 50MHz REF_CLK.

## Features

- Supports IEEE Std. 802.3 CSMA/CD protocol.
- Supports both half and full duplex for 10M/100M bps operation.
- Supports RMII interface.
- Supports MII Management function.
- Supports pause and remote pause function for flow control.
- Supports long frame (more than 1518 bytes) and short frame (less than 64 bytes) reception.
- Supports 16 entries CAM function for Ethernet MAC address recognition.
- Supports internal loop back mode for diagnostic.
- Supports 256 bytes embedded transmit and receive FIFO.
- Supports DMA function.

### EMC Descriptors

A link-list data structure named as descriptor is used to keep the control, status and data information of each frame. Through the descriptor, CPU and EMC exchange the information for frame reception and transmission.

Two different descriptors are defined in NUC950ADN. One named as Rx descriptor for frame reception and the other names as Tx descriptor for frame transmission. Each Rx descriptor consists of four words. There is much information kept in the descriptors and details are described as below.

#### 7.5.1.1 Rx Buffer Descriptor

| 31 | 30 | 29 | 16 | 15 | 0 |
|---|---|---|---|---|---|
| O | Rx Status | | | Receive Byte Count | |
| Receive Buffer Starting Address | | | | | BO |
| Reserved | | | | | |
| Next Rx Descriptor Starting Address | | | | | |

### Rx Descriptor Word 0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Owner | | Reserved | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | RP | ALIE | RXGD | PTLE | Reserved | CRCE | RXINTR |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RBC | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RBC | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:30] | **Owner** | **Ownership**<br>The ownership field defines which one, the CPU or EMC, is the owner of each Rx descriptor. Only the owner has right to modify the Rx descriptor and the others can read the Rx descriptor only.<br>00: The owner is CPU<br>01: Undefined<br>10: The owner is EMC<br>11: Undefined<br>If the O=2'b10 indicates the EMC RxDMA is the owner of Rx descriptor and the Rx descriptor is available for frame reception. After the frame reception completed, if the frame needed NAT translation, EMC RxDMA modify ownership field to 2'b11. Otherwise, the ownership field will be modified to 2'b00.<br>If the O=2'b00 indicates the CPU is the owner of Rx descriptor. After the CPU completes processing the frame, it modifies the ownership field to 2'b10 and releases the Rx descriptor to EMC RxDMA. |

| Bits | Descriptions | |
|---|---|---|
| [29:23] | **Rx Status** | **Receive Status**<br>This field keeps the status for frame reception. All status bits are updated by EMC. In the receive status, bits 29 to 23 are undefined and reserved for the future. |
| [22] | **RP** | **Runt Packet**<br>The RP indicates the frame stored in the data buffer pointed by Rx descriptor is a short frame (frame length is less than 64 bytes).<br>1'b0: The frame is not a short frame.<br>1'b1: The frame is a short frame. |
| [21] | **ALIE** | **Alignment Error**<br>The ALIE indicates the frame stored in the data buffer pointed by Rx descriptor is not a multiple of byte.<br>1'b0: The frame is a multiple of byte.<br>1'b1: The frame is not a multiple of byte. |
| [20] | **RXGD** | **Frame Reception Complete**<br>The RXGD indicates the frame reception has completed and stored in the data buffer pointed by Rx descriptor.<br>1'b0: The frame reception not complete yet.<br>1'b1: The frame reception completed. |
| [19] | **PTLE** | **Packet Too Long**<br>The PTLE indicates the frame stored in the data buffer pointed by Rx descriptor is a long frame (frame length is greater than 1518 bytes).<br>1'b0: The frame is not a long frame.<br>1'b1: The frame is a long frame. |
| [17] | **CRCE** | **CRC Error**<br>The CRCE indicates the frame stored in the data buffer pointed by Rx descriptor incurred CRC error.<br>1'b0: The frame doesn't incur CRC error.<br>1'b1: The frame incurred CRC error. |

| Bits | Descriptions | |
|---|---|---|
| [16] | **RXINTR** | **Receive Interrupt**<br>The RXINTR indicates the frame stored in the data buffer pointed by Rx descriptor caused an interrupt condition.<br>1'b0: The frame doesn't cause an interrupt.<br>1'b1: The frame caused an interrupt. |
| [15:0] | **RBC** | **Receive Byte Count**<br>The RBC indicates the byte count of the frame stored in the data buffer pointed by Rx descriptor. The four bytes CRC field is also included in the receive byte count. But if the SPCRC of register MCMDR is enabled, the four bytes CRC field will be excluded from the receive byte count. |

### Rx Descriptor Word 1

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RXBSA | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RXBSA | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RXBSA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXBSA | | | | | | BO | |

| Bits | Descriptions | |
|---|---|---|
| [31:2] | **RXBSA** | **Receive Buffer Starting Address**<br>The RXBSA indicates the starting address of the receive frame buffer. The RXBSA is used to be the bit 31 to 2 of memory address. In other words, the starting address of the receive frame buffer always located at word boundary. |
| [1:0] | **BO** | **Byte Offset**<br>The BO indicates the byte offset from RXBSA where the received frame begins to store. If the BO is 2'b01, the starting address where the received frame begins to store is RXBSA+2'b01, and so on. |

### Rx Descriptor Word 2

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | |

The Rx descriptor word 2 keeps obsolete information for MAC translation. Therefore, these information bits are undefined and should be ignored.

### Rx Descriptor Word 3

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| NRXDSA | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| NRXDSA | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| NRXDSA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| NRXDSA | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **NRXDSA** | **Next Rx Descriptor Starting Address**<br>The Rx descriptor is a link-list data structure. Consequently, NRXDSA is used to keep the starting address of the next Rx descriptor. The bits [1:0] will be ignored by EMC. So, all Rx descriptor must locate at word boundary memory address. |

#### 7.5.1.2 Tx Buffer Descriptor

| 31 | 16 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| O | Reserved | | I | C | P |
| Transmit Buffer Starting Address | | | | | BO |
| Tx Status | | Transmit Byte Count | | | |
| Next Tx Descriptor Starting Address | | | | | |

### Tx Descriptor Word 0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Owner | Reserved | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | IntEn | CRCApp | PadEn |

| Bits | Descriptions | |
|---|---|---|
| [31] | **Owner** | **Ownership**<br>The ownership field defines which one, the CPU or EMC, is the owner of each Tx descriptor. Only the owner has right to modify the Tx descriptor and the other can read the Tx descriptor only.<br>0: The owner is CPU<br>1: The owner is EMC<br>If the O=1'b1 indicates the EMC TxDMA is the owner of Tx descriptor and the Tx descriptor is available for frame transmission. After the frame transmission completed, EMC TxDMA modify ownership field to 1'b0 and return the ownership of Tx descriptor to CPU.<br>If the O=1'b0 indicates the CPU is the owner of Tx descriptor. After the CPU prepares new frame to wait transmission, it modifies the ownership field to 1'b1 and releases the Tx descriptor to EMC TxDMA. |
| [2] | **IntEn** | **Transmit Interrupt Enable**<br>The IntEn controls the interrupt trigger circuit after the frame transmission completed. If the IntEn is enabled, the EMC will trigger interrupt after frame transmission completed. Otherwise, the interrupt doesn't be triggered.<br>1'b0: Frame transmission interrupt is masked.<br>1'b1: Frame transmission interrupt is enabled. |

| Bits | Descriptions | |
|---|---|---|
| [1] | **CRCApp** | **CRC Append**<br>The CRCApp control the CRC append during frame transmission. If CRCApp is enabled, the 4-bytes CRC checksum will be appended to frame at the end of frame transmission.<br>1'b0: 4-bytes CRC appending is disabled.<br>1'b1: 4-bytes CRC appending is enabled. |
| [0] | **PadEN** | **Padding Enable**<br>The PadEN control the PAD bits appending while the length of transmission frame is less than 60 bytes. If PadEN is enabled, EMC does the padding automatically.<br>1'b0: PAD bits appending is disabled.<br>1'b1: PAD bits appending is enabled. |

### Tx Descriptor Word 1

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| TXBSA | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TXBSA | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TXBSA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXBSA | | | | | | BO | |

| Bits | Descriptions | |
|---|---|---|
| [31:2] | **TXBSA** | **Transmit Buffer Starting Address**<br>The TXBSA indicates the starting address of the transmit frame buffer. The TXBSA is used to be the bit 31 to 2 of memory address. In other words, the starting address of the transmit frame buffer always located at word boundary. |
| [1:0] | **BO** | **Byte Offset**<br>The BO indicates the byte offset from TXBSA where the transmit frame begins to read. If the BO is 2'b01, the starting address where the transmit frame begins to read is TXBSA+2'b01, and so on. |

### Tx Descriptor Word 2

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CCNT | | | | Reserved | SQE | PAU | TXHA |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LC | TXABT | NCS | EXDEF | TXCP | Reserved | DEF | TXINTR |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TBC | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TBC | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:28] | **CCNT** | **Collision Count**<br>The CCNT indicates the how many collisions occurred consecutively during a packet transmission. If the packet incurred 16 consecutive collisions during transmission, the CCNT will be 4'h0 and bit TXABT will be set to 1. |
| [26] | **SQE** | **SQE Error**<br>The SQE indicates the SQE error found at end of packet transmission on 10Mbps half-duplex mode. The SQE error check will only be done while both bit EnSQE of MCMDR is enabled and EMC is operating on 10Mbps half-duplex mode.<br>1'b0: No SQE error found at end of packet transmission.<br>1'b0: SQE error found at end of packet transmission. |
| [25] | **PAU** | **Transmission Paused**<br>The PAU indicates the next normal packet transmission process will be paused temporally because EMC received a PAUSE control frame, or S/W set bit SDPZ of MCMDR and make EMC to transmit a PAUSE control frame out.<br>1'b0: Next normal packet transmission process will go on.<br>1'b1: Next normal packet transmission process will be paused. |

| Bits | Descriptions | |
|---|---|---|
| [24] | **TXHA** | **P Transmission Halted**<br>The TXHA indicates the next normal packet transmission process will be halted because the bit TXON of MCMDR is disabled be S/W.<br>1'b0: Next normal packet transmission process will go on.<br>1'b1: Next normal packet transmission process will be halted. |
| [23] | **LC** | **Late Collision**<br>The LC indicates the collision occurred in the outside of 64 bytes collision window. This means after the 64 bytes of a frame has transmitted out to the network, the collision still occurred. The late collision check will only be done while EMC is operating on half-duplex mode.<br>1'b0: No collision occurred in the outside of 64 bytes collision window.<br>1'b1: Collision occurred in the outside of 64 bytes collision window. |
| [22] | **TXABT** | **Transmission Abort**<br>The TXABT indicates the packet incurred 16 consecutive collisions during transmission, and then the transmission process for this packet is aborted. The transmission abort is only available while EMC is operating on half-duplex mode.<br>1'b0: Packet doesn't incur 16 consecutive collisions during transmission.<br>1'b1: Packet incurred 16 consecutive collisions during transmission. |
| [21] | **NCS** | **No Carrier Sense**<br>The NCS indicates the MII I/F signal CRS doesn't active at the start of or during the packet transmission. The NCS is only available while EMC is operating on half-duplex mode.<br>1'b0: CRS signal actives correctly.<br>1'b1: CRS signal doesn't active at the start of or during the packet transmission. |
| [20] | **EXDEF** | **Defer Exceed**<br>The EXDEF indicates the frame waiting for transmission has deferred over 0.32768ms on 100Mbps mode, or 3.2768ms on 10Mbps mode. The deferral exceed check will only be done while bit NDEF of MCMDR is disabled, and EMC is operating on half-duplex mode.<br>1'b0: Frame waiting for transmission has not deferred over 0.32768ms (100Mbps) or 3.2768ms (10Mbps).<br>1'b1: Frame waiting for transmission has deferred over 0.32768ms (100Mbps) or 3.2768ms (10Mbps). |

| Bits | Descriptions | |
|---|---|---|
| [19] | **TXCP** | **Transmission Complete**<br>The TXCP indicates the packet transmission has completed correctly.<br>1'b0: The packet transmission doesn't complete.<br>1'b1: The packet transmission has completed. |
| [17] | **DEF** | **Transmission Deferred**<br>The DEF indicates the packet transmission has deferred once. The DEF is only available while EMC is operating on half-duplex mode.<br>1'b0: Packet transmission doesn't defer.<br>1'b1: Packet transmission has deferred once. |
| [16] | **TXINTR** | **Transmit Interrupt**<br>The TXINTR indicates the packet transmission caused an interrupt condition.<br>1'b0: The packet transmission doesn't cause an interrupt.<br>1'b1: The packet transmission caused an interrupt. |
| [15:0] | **TBC** | **Transmit Byte Count**<br>The TBC indicates the byte count of the frame stored in the data buffer pointed by Tx descriptor for transmission. |

### Tx Descriptor Word 3

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| NTXDSA | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| NTXDSA | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| NTXDSA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| NTXDSA | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **NTXDSA** | **Next Tx Descriptor Starting Address**<br>The Tx descriptor is a link-list data structure. Consequently, NTXDSA is used to keep the starting address of the next Tx descriptor. The bits [1:0] will be ignored by EMC. So, all Tx descriptor must locate at word boundary memory address. |

### 7.5.2 EMC Register Mapping

The EMC implements many registers and the registers are separated into three types, the control registers, the status registers and diagnostic registers. The control registers are used by S/W to pass control information to EMC. The status registers are used to keep EMC operation status for S/W.

**EMC Registers**

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMC_BA = 0xB000_3000** | | | | |
| **Control Registers (44)** | | | | |
| **CAMCMR** | 0xB000_3000 | R/W | CAM Command Register | 0x0000_0000 |
| **CAMEN** | 0xB000_3004 | R/W | CAM Enable Register | 0x0000_0000 |
| **CAM0M** | 0xB000_3008 | R/W | CAM0 Most Significant Word Register | 0x0000_0000 |
| **CAM0L** | 0xB000_300C | R/W | CAM0 Least Significant Word Register | 0x0000_0000 |
| **CAM1M** | 0xB000_3010 | R/W | CAM1 Most Significant Word Register | 0x0000_0000 |
| **CAM1L** | 0xB000_3014 | R/W | CAM1 Least Significant Word Register | 0x0000_0000 |
| **CAM2M** | 0xB000_3018 | R/W | CAM2 Most Significant Word Register | 0x0000_0000 |
| **CAM2L** | 0xB000_301C | R/W | CAM2 Least Significant Word Register | 0x0000_0000 |
| **CAM3M** | 0xB000_3020 | R/W | CAM3 Most Significant Word Register | 0x0000_0000 |
| **CAM3L** | 0xB000_3024 | R/W | CAM3 Least Significant Word Register | 0x0000_0000 |
| **CAM4M** | 0xB000_3028 | R/W | CAM4 Most Significant Word Register | 0x0000_0000 |
| **CAM4L** | 0xB000_302C | R/W | CAM4 Least Significant Word Register | 0x0000_0000 |
| **CAM5M** | 0xB000_3030 | R/W | CAM5 Most Significant Word Register | 0x0000_0000 |
| **CAM5L** | 0xB000_3034 | R/W | CAM5 Least Significant Word Register | 0x0000_0000 |
| **CAM6M** | 0xB000_3038 | R/W | CAM6 Most Significant Word Register | 0x0000_0000 |
| **CAM6L** | 0xB000_303C | R/W | CAM6 Least Significant Word Register | 0x0000_0000 |
| **CAM7M** | 0xB000_3040 | R/W | CAM7 Most Significant Word Register | 0x0000_0000 |
| **CAM7L** | 0xB000_3044 | R/W | CAM7 Least Significant Word Register | 0x0000_0000 |
| **CAM8M** | 0xB000_3048 | R/W | CAM8 Most Significant Word Register | 0x0000_0000 |
| **CAM8L** | 0xB000_304C | R/W | CAM8 Least Significant Word Register | 0x0000_0000 |
| **CAM9M** | 0xB000_3050 | R/W | CAM9 Most Significant Word Register | 0x0000_0000 |
| **CAM9L** | 0xB000_3054 | R/W | CAM9 Least Significant Word Register | 0x0000_0000 |

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAM10M** | 0xB000_3058 | R/W | CAM10 Most Significant Word Register | 0x0000_0000 |
| **CAM10L** | 0xB000_305C | R/W | CAM10 Least Significant Word Register | 0x0000_0000 |
| **CAM11M** | 0xB000_3060 | R/W | CAM11 Most Significant Word Register | 0x0000_0000 |
| **CAM11L** | 0xB000_3064 | R/W | CAM11 Least Significant Word Register | 0x0000_0000 |
| **CAM12M** | 0xB000_3068 | R/W | CAM12 Most Significant Word Register | 0x0000_0000 |
| **CAM12L** | 0xB000_306C | R/W | CAM12 Least Significant Word Register | 0x0000_0000 |
| **CAM13M** | 0xB000_3070 | R/W | CAM13 Most Significant Word Register | 0x0000_0000 |
| **CAM13L** | 0xB000_3074 | R/W | CAM13 Least Significant Word Register | 0x0000_0000 |
| **CAM14M** | 0xB000_3078 | R/W | CAM14 Most Significant Word Register | 0x0000_0000 |
| **CAM14L** | 0xB000_307C | R/W | CAM14 Least Significant Word Register | 0x0000_0000 |
| **CAM15M** | 0xB000_3080 | R/W | CAM15 Most Significant Word Register | 0x0000_0000 |
| **CAM15L** | 0xB000_3084 | R/W | CAM15 Least Significant Word Register | 0x0000_0000 |
| **TXDLSA** | 0xB000_3088 | R/W | Transmit Descriptor Link List Start Address Register | 0xFFFF_FFFC |
| **RXDLSA** | 0xB000_308C | R/W | Receive Descriptor Link List Start Address Reg. | 0xFFFF_FFFC |
| **MCMDR** | 0xB000_3090 | R/W | MAC Command Register | 0x0000_0000 |
| **MIID** | 0xB000_3094 | R/W | MII Management Data Register | 0x0000_0000 |
| **MIIDA** | 0xB000_3098 | R/W | MII Management Control and Address Register | 0x0090_0000 |
| **FFTCR** | 0xB000_309C | R/W | FIFO Threshold Control Register | 0x0000_0101 |
| **TSDR** | 0xB000_30A0 | W | Transmit Start Demand Register | Undefined |
| **RSDR** | 0xB000_30A4 | W | Receive Start Demand Register | Undefined |
| **DMARFC** | 0xB000_30A8 | R/W | Maximum Receive Frame Control Register | 0x0000_0800 |
| **MIEN** | 0xB000_30AC | R/W | MAC Interrupt Enable Register | 0x0000_0000 |
| **Status Registers (11)** | | | | |
| **MISTA** | 0xB000_30B0 | R/W | MAC Interrupt Status Register | 0x0000_0000 |
| **MGSTA** | 0xB000_30B4 | R/W | MAC General Status Register | 0x0000_0000 |
| **MPCNT** | 0xB000_30B8 | R/W | Missed Packet Count Register | 0x0000_7FFF |
| **MRPC** | 0xB000_30BC | R | MAC Receive Pause Count Register | 0x0000_0000 |
| **MRPCC** | 0xB000_30C0 | R | MAC Receive Pause Current Count Register | 0x0000_0000 |
| **MREPC** | 0xB000_30C4 | R | MAC Remote Pause Count Register | 0x0000_0000 |
| **DMARFS** | 0xB000_30C8 | R/W | DMA Receive Frame Status Register | 0x0000_0000 |
| **CTXDSA** | 0xB000_30CC | R | Current Transmit Descriptor Start Address Reg. | 0x0000_0000 |
| **CTXBSA** | 0xB000_30D0 | R | Current Transmit Buffer Start Address Register | 0x0000_0000 |

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRXDSA** | 0xB000_30D4 | R | Current Receive Descriptor Start Address Reg. | 0x0000_0000 |
| **CRXBSA** | 0xB000_30D8 | R | Current Receive Buffer Start Address Register | 0x0000_0000 |

### 7.5.3 EMC Register Details

## CAM Command Register (CAMCMR)

The EMC of NUC950ADN supports CAM function for destination MAC address recognition. The CAMCMR control the CAM comparison function, and unicast, multicast, and broadcast packet reception.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAMCMR | 0xB000_3000 | R/W | CAM Command Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | RMII | ECMP | CCAM | ABP | AMP | AUP |

| Bits | Descriptions | |
|---|---|---|
| [5] | **RMII** | **Enable RMII Input Data Sampled by Negative Edge of REFCLK**<br>1'b0: PHY_CRSDV and PHY_RXD[1:0] are sampled by the positive edge of REFCLK<br>1'b1: PHY_CRSDV and PHY_RxD[1:0] are sampled by the negative edge of REFCLK |
| [4] | **ECMP** | **Enable CAM Compare**<br>The ECMP controls the enable of CAM comparison function for destination MAC address recognition. If S/W wants to receive a packet with specific destination MAC address, configures the MAC address into anyone of 16 CAM entries, then enables that CAM entry and set ECMP to 1.<br>1'b0: Disable CAM comparison function for destination MAC address recognition.<br>1'b1: Enable CAM comparison function for destination MAC address recognition. |

| Bits | Descriptions | |
|---|---|---|
| [3] | **CCAM** | **Complement CAM Compare**<br>The CCAM controls the complement of the CAM comparison result. If the ECMP and CCAM are both enabled, the incoming packet with specific destination MAC address configured in CAM entry will be dropped. And the incoming packet with destination MAC address doesn't configured in any CAM entry will be received.<br>1'b0: The CAM comparison result doesn't be complemented.<br>1'b1: The CAM comparison result will be complemented. |
| [2] | **ABP** | **Accept Broadcast Packet**<br>The ABP controls the broadcast packet reception. If ABP is enabled, EMC receives all incoming packet its destination MAC address is a broadcast address.<br>1'b0: EMC receives packet depends on the CAM comparison result.<br>1'b1: EMC receives all broadcast packets. |
| [1] | **AMP** | **Accept Multicast Packet**<br>The AMP controls the multicast packet reception. If AMP is enabled, EMC receives all incoming packet its destination MAC address is a multicast address.<br>1'b0: EMC receives packet depends on the CAM comparison result.<br>1'b1: EMC receives all multicast packets. |
| [0] | **AUP** | **Accept Unicast Packet**<br>The AUP controls the unicast packet reception. If AUP is enabled, EMC receives all incoming packet its destination MAC address is a unicast address.<br>1'b0: EMC receives packet depends on the CAM comparison result.<br>1'b1: EMC receives all unicast packets. |

**CAMCMR Setting and Comparison Result**

The following table is the address recognition result in different CAMCMR configuration. The column Result shows the incoming packet type that can pass the address recognition in specific CAM configuration. The C, U, M and B represents the:

*C*: It indicates the destination MAC address of incoming packet has been configured in CAM entry.

*U*: It indicates the incoming packet is a unicast packet.

*M*: It indicates the incoming packet is a multicast packet.

*B*: It indicates the incoming packet is a broadcast packet.

| ECMP | CCAM | AUP | AMP | ABP | Result |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | No Packet |
| 0 | 0 | 0 | 0 | 1 | B |
| 0 | 0 | 0 | 1 | 0 | M |
| 0 | 0 | 0 | 1 | 1 | M B |
| 0 | 0 | 1 | 0 | 0 | C U |
| 0 | 0 | 1 | 0 | 1 | C U B |
| 0 | 0 | 1 | 1 | 0 | C U M |
| 0 | 0 | 1 | 1 | 1 | C U M B |
| 0 | 1 | 0 | 0 | 0 | C U M B |
| 0 | 1 | 0 | 0 | 1 | C U M B |
| 0 | 1 | 0 | 1 | 0 | C U M B |
| 0 | 1 | 0 | 1 | 1 | C U M B |
| 0 | 1 | 1 | 0 | 0 | C U M B |
| 0 | 1 | 1 | 0 | 1 | C U M B |
| 0 | 1 | 1 | 1 | 0 | C U M B |
| 0 | 1 | 1 | 1 | 1 | C U M B |
| 1 | 0 | 0 | 0 | 0 | C |
| 1 | 0 | 0 | 0 | 1 | C B |
| 1 | 0 | 0 | 1 | 0 | C M |
| 1 | 0 | 0 | 1 | 1 | C N B |
| 1 | 0 | 1 | 0 | 0 | C U |
| 1 | 0 | 1 | 0 | 1 | C U B |
| 1 | 0 | 1 | 1 | 0 | C U M |
| 1 | 0 | 1 | 1 | 1 | C U M B |
| 1 | 1 | 0 | 0 | 0 | U M B |
| 1 | 1 | 0 | 0 | 1 | U M B |
| 1 | 1 | 0 | 1 | 0 | U M B |
| 1 | 1 | 0 | 1 | 1 | U M B |
| 1 | 1 | 1 | 0 | 0 | C U M B |
| 1 | 1 | 1 | 0 | 1 | C U M B |
| 1 | 1 | 1 | 1 | 0 | C U M B |
| 1 | 1 | 1 | 1 | 1 | C U M B |

## CAM Enable Register (CAMEN)

The CAMEN controls the validation of each CAM entry. Each CAM entry must be enabled first before it can participate in the destination MAC address recognition.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAMEN | 0xB000_3004 | R/W | CAM Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CAM15EN | CAM14EN | CAM13EN | CAM12EN | CAM11EN | CAM10EN | CAM9EN | CAM8EN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CAM7EN | CAM6EN | CAM5EN | CAM4EN | CAM3EN | CAM2EN | CAM1EN | CAM0EN |

| Bits | Descriptions | |
|---|---|---|
| [x] | **CAMxEN** | **CAM Entry x Enable**<br>The CAMxEN controls the validation of CAM entry x. The x can be 0 to 15.<br>The CAM entry 13, 14 and 15 are for PAUSE control frame transmission. If S/W wants to transmit a PAUSE control frame out to network, the enable bits of these three CAM entries all must be enabled first.<br>1'b0: CAM entry x is disabled.<br>1'b1: CAM entry x is enabled. |

## CAM Entry Registers (CAMxx)

In the EMC of NUC950ADN, there are 16 CAM entries. In these 16 CAM entries, 13 entries (entry 0~12) are to keep destination MAC address for packet recognition, and the other 3 entries (entry 13~15) are for PAUSE control frame transmission. Each CAM entry consists of 6 bytes. Consequently, 2 register ports are needed for each CAM entry.

For packet recognition, a register pair {CAMxM, CAMxL} represents a CAM entry and can keep a destination MAC address. The corresponding CAM enable bit CAMxEN of CAMEN register is also needed be enabled. The x can be the 0 to 12.

For PAUSE control frame transmission, first, S/W must configure destination MAC address of control frame into the register pair {CAM13M, CAM13L}, source MAC address into the register pair {CAM14M, CAM14L}, and configure length/type, op-code and operand of control frame into the register pair {CAM15M, CAM15L}. The bit CAM13EN, CAM14EN and CAM15EN of CAMEN register are also needed be enabled. Then, enable the bit SDPZ of MCMDR register.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAM0M | 0xB000_3008 | R/W | CAM0 Most Significant Word Register | 0x0000_0000 |
| CAM0L | 0xB000_300C | | CAM0 Least Significant Word Register | 0x0000_0000 |
| : | : | | : | : |
| CAM15M | 0xB000_3080 | | CAM15 Most Significant Word Register | 0x0000_0000 |
| CAM15L | 0xB000_3084 | | CAM15 Least Significant Word Register | 0x0000_0000 |

## CAMxM

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MAC Address Byte 5 (MSB) | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MAC Address Byte 4 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MAC Address Byte 3 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MAC Address Byte 2 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **CAMxM** | **CAMx Most Significant Word**<br>The CAMxM keeps the bit 47~16 of MAC address. The x can be the 0~14. The register pair {CAMxM, CAMxL} represents a CAM entry and can keep a MAC address. For example, if the MAC address 00-50-BA-33-BA-44 is kept in CAM entry 1, the register CAM1M is 32'h0050_BA33 and CAM1L is 32'hBA44_0000. |

### CAMxL

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MAC Address Byte 1 | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MAC Address Byte 0 (LSB) | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **CAMxL** | **CAMx Least Significant Word**<br>The CAMxL keeps the bit 15~0 of MAC address. The x can be the 0~14. The register pair {CAMxM, CAMxL} represents a CAM entry and can keep a MAC address. For example, if the MAC address 00-50-BA-33-BA-44 is kept in CAM entry 1, the register CAM1M is 32'h0050_BA33 and CAM1L is 32'hBA44_0000. |

### CAM15M

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Length/Type (MSB) | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Length/Type | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| OP-Code (MSB) | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OP-Code | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:16] | **Length/Type** | **Length/Type Field of PAUSE Control Frame**<br>In the PAUSE control frame, a length/type field is defined and will be 16'h8808. |
| [15:0] | **OP-Code** | **OP Code Field of PAUSE Control Frame**<br>In the PAUSE control frame, an op code field is defined and will be 16'h0001. |

## CAM15L

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Operand (MSB) | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Operand | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:16] | **Operand** | **Pause Parameter**<br>In the PAUSE control frame, an operand field is defined and controls how much time the destination Ethernet MAC Controller is paused. The unit of the operand is the slot time, the 512 bits time. |

## Transmit Descriptor Link List Start Address Register (TXDLSA)

The Tx descriptor defined in EMC is a link-list data structure. The TXDLSA keeps the starting address of this link-list. In other words, the TXDLSA keeps the starting address of the 1st Tx descriptor. S/W must configure TXDLSA before enable bit TXON of MCMDR register.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| TXDLSA | 0xB000_3088 | R/W | Transmit Descriptor Link List Start Address Register | 0xFFFF_FFFC |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| TXDLSA | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TXDLSA | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TXDLSA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXDLSA | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **TXDLSA** | **Transmit Descriptor Link-List Start Address**<br>The TXDLSA keeps the start address of transmit descriptor link-list. If the S/W enables the bit TXON of MCMDR register, the content of TXDLSA will be loaded into the current transmit descriptor start address register (CTXDSA). The TXDLSA doesn't be updated by EMC. During the operation, EMC will ignore the bits [1:0] of TXDLSA. This means that each Tx descriptor always must locate at word boundary memory address. |

## Receive Descriptor Link List Start Address Register (RXDLSA)

The Rx descriptor defined in EMC is a link-list data structure. The RXDLSA keeps the starting address of this link-list. In other words, the RXDLSA keeps the starting address of the 1st Rx descriptor. S/W must configure RXDLSA before enable bit RXON of MCMDR register.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| RXDLSA | 0xB000_308C | R/W | Receive Descriptor Link List Start Address Register | 0xFFFF_FFFC |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RXDLSA | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RXDLSA | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RXDLSA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXDLSA | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **RTXDLSA** | **Receive Descriptor Link-List Start Address**<br>The RXDLSA keeps the start address of receive descriptor link-list. If the S/W enables the bit RXON of MCMDR register, the content of RXDLSA will be loaded into the current receive descriptor start address register (CRXDSA). The RXDLSA doesn't be updated by EMC. During the operation, EMC will ignore the bits [1:0] of RXDLSA. This means that each Rx descriptor always must locate at word boundary memory address. |

## MAC Command Register (MCMDR)

The MCMDR provides the control information for EMC. Some command settings affect both frame transmission and reception, such as bit FDUP, the full/half duplex mode selection, or bit OPMOD, the 100/10M bps mode selection. Some command settings control frame transmission and reception separately, likes bit TXON and RXON.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MCMDR | 0xB000_3090 | R/W | MAC Command Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | SWR |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | LBK | OPMOD | EnMDC | FDUP | EnSQE | SDPZ |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | NDEF | TXON |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | SPCRC | AEP | ACP | ARP | ALP | RXON |

| Bits | Descriptions | |
|---|---|---|
| [24] | **SWR** | **Software Reset**<br>The SWR implements a reset function to make the EMC return default state. The SWR is a self-clear bit. This means after the software reset finished, the SWR will be cleared automatically. Enable SWR can also reset all control and status registers, exclusive of these two bits EnRMII and OPMOD of MCMDR register.<br>The EMC re-initial is needed after the software reset completed.<br>1'b0: Software reset completed.<br>1'b1: Enable software reset. |

| Bits | Descriptions | |
|---|---|---|
| [21] | **LBK** | **Internal Loop Back Select**<br>The LBK enables the EMC operating on internal loop-back mode. If the LBK is enabled, the packet transmitted out will be loop-backed to Rx. If the EMC is operating on internal loop-back mode, it also means the EMC is operating on full-duplex mode and the value of FDUP of MCMDR register is ignored. Beside, the LBK doesn't be affected by SWR bit.<br>1'b0: The EMC operates in normal mode.<br>1'b1: The EMC operates in internal loop-back mode. |
| [20] | **OPMOD** | **Operation Mode Select**<br>The OPMOD defines the EMC is operating on 10M or 100M bps mode. The OPMOD doesn't be affected by SWR bit.<br>1'b0: The EMC operates on 10Mbps mode.<br>1'b1: The EMC operates on 100Mbps mode. |
| [19] | **EnMDC** | **Enable MDC Clock Generation**<br>The EnMDC controls the MDC clock generation for MII Management Interface. If the EnMDC is set to 1, the MDC clock generation is enabled. Otherwise, the MDC clock generation is disabled. Consequently, if S/W wants to access the registers of external PHY through MII Management Interface, the EnMDC must be set to high.<br>1'b0: Disable MDC clock generation.<br>1'b1: Enable MDC clock generation. |
| [18] | **FDUP** | **Full Duplex Mode Select**<br>The FDUP controls that EMC is operating on full or half duplex mode.<br>1'b0: The EMC operates on half duplex mode.<br>1'b1: The EMC operates on full duplex mode. |
| [17] | **EnSQE** | **Enable SQE Checking**<br>The EnSQE controls the enable of SQE checking. The SQE checking is only available while EMC is operating on 10M bps and half duplex mode. In other words, the EnSQE cannot affect EMC operation, if the EMC is operating on 100M bps or full duplex mode.<br>1'b0: Disable SQE checking while EMC is operating on 10Mbps and half duplex mode.<br>1'b1: Enable SQE checking while EMC is operating on 10Mbps and half duplex mode. |

| Bits | Descriptions | |
|---|---|---|
| [16] | **SDPZ** | **Send PAUSE Frame**<br>The SDPZ controls the PAUSE control frame transmission.<br>If S/W wants to send a PAUSE control frame out, the CAM entry 13, 14 and 15 must be configured first and the corresponding CAM enable bit of CAMEN register also must be set. Then, set SDPZ to 1 enables the PAUSE control frame transmission.<br>The SDPZ is a self-clear bit. This means after the PAUSE control frame transmission has completed, the SDPZ will be cleared automatically.<br>It is recommended that only enables SPDZ while EMC is operating on full duplex mode.<br>1'b0: The PAUSE control frame transmission has completed.<br>1'b1: Enable EMC to transmit a PAUSE control frame out. |
| [9] | **NDEF** | **No Defer**<br>The NDEF controls the enable of deferral exceed counter. If NDEF is set to high, the deferral exceed counter is disabled. The NDEF is only useful while EMC is operating on half duplex mode.<br>1'b0: The deferral exceed counter is enabled.<br>1'b1: The deferral exceed counter is disabled. |
| [8] | **TXON** | **Frame Transmission ON**<br>The TXON controls the normal packet transmission of EMC. If the TXON is set to high, the EMC starts the packet transmission process, including the Tx descriptor fetching, packet transmission and Tx descriptor modification.<br>It is must to finish EMC initial sequence before enable TXON. Otherwise, the EMC operation is undefined.<br>If the TXON is disabled during EMC is transmitting a packet out, the EMC stops the packet transmission process after the current packet transmission finished.<br>1'b0: The EMC stops packet transmission process.<br>1'b1: The EMC starts packet transmission process. |
| [5] | **SPCRC** | **Strip CRC Checksum**<br>The SPCRC controls if the length of incoming packet is calculated with 4 bytes CRC checksum. If the SPCRC is set to high, 4 bytes CRC checksum is excluded from length calculation of incoming packet.<br>1'b0: The 4 bytes CRC checksum is included in packet length calculation.<br>1'b1: The 4 bytes CRC checksum is excluded in packet length calculation. |

| Bits | Descriptions | |
|---|---|---|
| [4] | **AEP** | **Accept CRC Error Packet**<br>The AEP controls the EMC accepts or drops the CRC error packet. If the AEP is set to high, the incoming packet with CRC error will be received by EMC as a good packet.<br>1'b0: The CRC error packet will be dropped by EMC.<br>1'b1: The CRC error packet will be accepted by EMC. |
| [3] | **ACP** | **Accept Control Packet**<br>The ACP controls the control frame reception. If the ACP is set to high, the EMC will accept the control frame. Otherwise, the control frame will be dropped. It is recommended that S/W only enable AEP while EMC is operating on full duplex mode.<br>1'b0: The control frame will be dropped by EMC.<br>1'b1: The control frame will be accepted by EMC. |
| [2] | **ARP** | **Accept Runt Packet**<br>The ARP controls the runt packet, which length is less than 64 bytes, reception. If the ARP is set to high, the EMC will accept the runt packet.<br>Otherwise, the runt packet will be dropped.<br>1'b0: The runt packet will be dropped by EMC.<br>1'b1: The runt packet will be accepted by EMC. |
| [1] | **ALP** | **Accept Long Packet**<br>The ALP controls the long packet, which packet length is greater than 1518 bytes, reception. If the ALP is set to high, the EMC will accept the long packet.<br>Otherwise, the long packet will be dropped.<br>1'b0: The long packet will be dropped by EMC.<br>1'b1: The long packet will be accepted by EMC. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **RXON** | **Frame Reception ON**<br>The RXON controls the normal packet reception of EMC. If the RXON is set to high, the EMC starts the packet reception process, including the Rx descriptor fetching, packet reception and Rx descriptor modification.<br>It is must to finish EMC initial sequence before enable RXON. Otherwise, the EMC operation is undefined.<br>If the RXON is disabled during EMC is receiving an incoming packet, the EMC stops the packet reception process after the current packet reception finished.<br>1'b0: The EMC stops packet reception process.<br>1'b1: The EMC starts packet reception process. |

## MII Management Data Register (MIID)

The EMC provides MII management function to access the control and status registers of the external PHY. The MIID register is used to store the data that will be written into the registers of external PHY for write command or the data that is read from the registers of external PHY for read command.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MIID | 0xB000_3094 | R/W | MII Management Data Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MIIData | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MIIData | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **MIIData** | **MII Management Data**<br>The MIIData is the 16 bits data that will be written into the registers of external PHY for MII Management write command or the data from the registers of external PHY for MII Management read command. |

## MII Management Control and Address Register (MIIDA)

The EMC provides MII management function to access the control and status registers of the external PHY. The MIIDA register is used to keep the MII management command information, like the register address, external PHY address, MDC clocking rate, read/write etc.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MIIDA | 0xB000_3098 | R/W | MII Management Control and Address Register | 0x0090_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MDCCR | | | | MDCON | PreSP | BUSY | Write |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | PHYAD | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | PHYRAD | | | | |

| Bits | Descriptions |
|---|---|

<table>
<tr><th>Bits</th><th>Descriptions</th><th></th></tr>
<tr><td>[23:20]</td><td>MDCCR</td><td>
MDC Clock Rating (Default Value: 4’h9)<br>
The MDCCR controls the MDC clock rating for MII Management I/F.<br>
Depend on the IEEE Std. 802.3 clause 22.2.2.11, the minimum period for MDC shall be 400ns. In other words, the maximum frequency for MDC is 2.5MHz. The MDC is divided from the AHB bus clock, the HCLK. Consequently, for different HCLK the different ratios are required to generate appropriate MDC clock.<br>
The following table shows relationship between HCLK and MDC clock in different MDCCR configurations. The $T_{HCLK}$ indicates the period of HCLK.
<table>
<tr><th>MDCCR [23:20]</th><th>MDC Clock Period</th><th>MDC Clock Frequency</th></tr>
<tr><td>4’b0000</td><td>4 x $T_{HCLK}$</td><td>HCLK/4</td></tr>
<tr><td>4’b0001</td><td>6 x $T_{HCLK}$</td><td>HCLK/6</td></tr>
<tr><td>4’b0010</td><td>8 x $T_{HCLK}$</td><td>HCLK/8</td></tr>
<tr><td>4’b0011</td><td>12 x $T_{HCLK}$</td><td>HCLK/12</td></tr>
<tr><td>4’b0100</td><td>16 x $T_{HCLK}$</td><td>HCLK/16</td></tr>
<tr><td>4’b0101</td><td>20 x $T_{HCLK}$</td><td>HCLK/20</td></tr>
<tr><td>4’b0110</td><td>24 x $T_{HCLK}$</td><td>HCLK/24</td></tr>
<tr><td>4’b0111</td><td>28 x $T_{HCLK}$</td><td>HCLK/28</td></tr>
<tr><td>4’b1000</td><td>30 x $T_{HCLK}$</td><td>HCLK/30</td></tr>
<tr><td>4’b1001</td><td>32 x $T_{HCLK}$</td><td>HCLK/32</td></tr>
<tr><td>4’b1010</td><td>36 x $T_{HCLK}$</td><td>HCLK/36</td></tr>
<tr><td>4’b1011</td><td>40 x $T_{HCLK}$</td><td>HCLK/40</td></tr>
<tr><td>4’b1100</td><td>44 x $T_{HCLK}$</td><td>HCLK/44</td></tr>
<tr><td>4’b1101</td><td>48 x $T_{HCLK}$</td><td>HCLK/48</td></tr>
<tr><td>4’b1110</td><td>54 x $T_{HCLK}$</td><td>HCLK/54</td></tr>
<tr><td>4’b1111</td><td>60 x $T_{HCLK}$</td><td>HCLK/60</td></tr>
</table>
</td></tr>
<tr><td>[19]</td><td>MDCON</td><td>
MDC Clock ON Always<br>
The MDC controls the MDC clock generation. If the MDCON is set to high, the MDC clock actives always. Otherwise, the MDC will only active while S/W issues a MII management command.<br>
1’b0: The MDC clock will only active while S/W issues a MII management command.<br>
1’b1: The MDC clock actives always.
</td></tr>
</table>

| Bits | Descriptions | |
|---|---|---|
| [18] | **PreSP** | **Preamble Suppress**<br>The PreSP controls the preamble field generation of MII management frame. If the PreSP is set to high, the preamble field generation of MII management frame is skipped.<br>1'b0: Preamble field generation of MII management frame is not skipped.<br>1'b1: Preamble field generation of MII management frame is skipped. |
| [17] | **BUSY** | **Busy Bit**<br>The BUSY controls the enable of the MII management frame generation. If S/W wants to access registers of external PHY, it set BUSY to high and EMC generates the MII management frame to external PHY through MII Management I/F. The BUSY is a self-clear bit. This means the BUSY will be cleared automatically after the MII management command finished.<br>1'b0: The MII management has finished.<br>1'b1: Enable EMC to generate a MII management command to external PHY. |
| [16] | **Write** | **Write Command**<br>The Write defines the MII management command is a read or write.<br>1'b0: The MII management command is a read command.<br>1'b1: The MII management command is a write command. |
| [12:8] | **PHYAD** | **PHY Address**<br>The PHYAD keeps the address to differentiate which external PHY is the target of the MII management command. |
| [4:0] | **PHYRAD** | **PHY Register Address**<br>The PHYRAD keeps the address to indicate which register of external PHY is the target of the MII management command. |

**MII Management Function Frame Format**

In IEEE Std. 802.3 clause 22.2.4, the MII management function is defined. The MII management function is used for the purpose of controlling the PHY and gathering status from the PHY. The MII management frame format is shown as follow.

| | Management frame fields | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **PRE** | **ST** | **OP** | **PHYAD** | **REGAD** | **TA** | **DATA** | **IDLE** |
| **READ** | **1…1** | **01** | **10** | **AAAAA** | **RRRRR** | **Z0** | **DDDDDDDDDDDDDDDD** | **Z** |
| **WRITE** | **1…1** | **01** | **01** | **AAAAA** | **RRRRR** | **10** | **DDDDDDDDDDDDDDDD** | **Z** |

**MII Management Function Configure Sequence**

| Read | Write |
|---|---|
| 1. Set appropriate MDCCR. | 1. Write data to MIID register |
| 2. Set PHYAD and PHYRAD. | 2. Set appropriate MDCCR. |
| 3. Set Write to 1'b0 | 3. Set PHYAD and PHYRAD. |
| 4. Set bit BUSY to 1'b1 to send a MII management frame out. | 4. Set Write to 1'b1 |
| 5. Wait BUSY to become 1'b0. | 5. Set bit BUSY to 1'b1 to send a MII management frame out. |
| 6. Read data from MIID register. | 6. Wait BUSY to become 1'b0. |
| 7. Finish the read command. | 7. Finish the write command. |

## FIFO Threshold Control Register (FFTCR)

The FFTCR defines the high and low threshold of internal FIFO, including TxFIFO and RxFIFO. The threshold of internal FIFO is related to EMC request generation and when the frame transmission starts. The FFTCR also defines the burst length of AHB bus cycle for system memory access.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| FFTCR | 0xB000_309C | R/W | FIFO Threshold Control Register | 0x0000_0101 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | BLength | | Reserved | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | TxTHD | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | RxTHD | |

| Bits | Descriptions | |
|---|---|---|
| [21:20] | **Blength** | **DMA Burst Length**<br>The Blength defines the burst length of AHB bus cycle while EMC accesses system memory.<br>2'b00: 4 words<br>2'b01: 8 words<br>2'b10: 16 words<br>2'b11: 16 words |

| | | |
|---|---|---|
| [9:8] | **TxTHD** | **TxFIFO Low Threshold**<br>Default Value: 2'b01<br>The TxTHD controls when TxDMA requests internal arbiter for data transfer between system memory and TxFIFO. The TxTHD defines not only the low threshold of TxFIFO, but also the high threshold. The high threshold is the twice of low threshold always. During the packet transmission, if the TxFIFO reaches the high threshold, the TxDMA stops generate request to transfer frame data from system memory to TxFIFO. If the frame data in TxFIFO is less than low threshold, TxDMA starts to transfer frame data from system memory to TxFIFO.<br>The TxTHD also defines when the TxMAC starts to transmit frame out to network. The TxMAC starts to transmit the frame out while the TxFIFO first time reaches the high threshold during the transmission of the frame. If the frame data length is less than TxFIFO high threshold, the TxMAC starts to transmit the frame out after the frame data are all inside the TxFIFO.<br>2'b00: Undefined.<br>2'b01: TxFIFO low threshold is 64B and high threshold is 128B.<br>2'b10: TxFIFO low threshold is 80B and high threshold is 160B.<br>2'b11: TxFIFO low threshold is 96B and high threshold is 192B. |
| [1:0] | **RxTHD** | **RxFIFO High Threshold**<br>Default Value: 2'b01<br>The RxTHD controls when RxDMA requests internal arbiter for data transfer between RxFIFO and system memory. The RxTHD defines not only the high threshold of RxFIFO, but also the low threshold. The low threshold is the half of high threshold always. During the packet reception, if the RxFIFO reaches the high threshold, the RxDMA starts to transfer frame data from RxFIFO to system memory. If the frame data in RxFIFO is less than low threshold, RxDMA stops to transfer the frame data to system memory.<br>2'b00: Depend on the burst length setting. If the burst length is 8 words, high threshold is 8 words, too.<br>2'b01: RxFIFO high threshold is 64B and low threshold is 32B.<br>2'b10: RxFIFO high threshold is 128B and low threshold is 64B.<br>2'b11: RxFIFO high threshold is 192B and low threshold is 96B. |

## Transmit Start Demand Register (TSDR)

If the Tx descriptor is not available for use of TxDMA after the TXON of MCMDR register is enabled, the FSM (Finite State Machine) of TxDMA enters the Halt state and the frame transmission is halted. After the S/W has prepared the new Tx descriptor for frame transmission, it must issue a write command to TSDR register to make TxDMA leave Halt state and contiguous frame transmission. The TSDR is a write only register and read from this register is undefined. The write to TSDR register has took effect only while TxDMA stayed at Halt state.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| TSDR | 0xB000_30A0 | W | Transmit Start Demand Register | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| TSD | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TSD | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TSD | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TSD | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **TSD** | **Transmit Start Demand** |

## Receive Start Demand Register (RSDR)

If the Rx descriptor is not available for use of RxDMA after the RXON of MCMDR register is enabled, the FSM (Finite State Machine) of RxDMA enters the Halt state and the frame reception is halted. After the S/W has prepared the new Rx descriptor for frame reception, it must issue a write command to RSDR register to make RxDMA leave Halt state and contiguous frame reception. The RSDR is a write only register and read from this register is undefined. The write to RSDR register has took effect only while RxDMA stayed at Halt state.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| RSDR | 0xB000_30A4 | W | Receive Start Demand Register | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RSD | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RSD | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RSD | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RSD | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | RSD | Receive Start Demand |

## Maximum Receive Frame Control Register (DMARFC)

The DMARFC defines the maximum frame length for a received frame that can be stored in the system memory. It is recommend that only use this register while S/W wants to receive a frame which length is greater than 1518 bytes.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| DMARFC | 0xB000_30A8 | R/W | Maximum Receive Frame Control Register | 0x0000_0800 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RXMS | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXMS | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **RXMS** | **Maximum Receive Frame Length**<br>Default Value: 16'h0800<br>The RXMS defines the maximum frame length for received frame. If the frame length of received frame is greater than RXMS, and bit EnDFO of MIEN register is also enabled, the bit DFOI of MISTA register is set and the Rx interrupt is triggered.<br>It is recommended that only use RXMS to qualify the length of received frame while S/W wants to receive a frame which length is greater than 1518 bytes. |

## MAC Interrupt Enable Register (MIEN)

The MIEN controls the enable of EMC interrupt status to generate interrupt. Two interrupts, RXINTR for frame reception and TXINTR for frame transmission, are generated from EMC to CPU.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MIEN | 0xB000_30AC | R/W | MAC Interrupt Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | EnTxBErr |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| EnTDU | EnLC | EnTXABT | EnNCS | EnEXDEF | EnTXCP | EnTXEMP | EnTXINTR |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | EnCFR | Reserved | | EnRxBErr | EnRDU | EnDEN | EnDFO |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EnMMP | EnRP | EnALIE | EnRXGD | EnPTLE | EnRXOV | EnCRCE | EnRXINTR |

| Bits | Descriptions | |
|---|---|---|
| [24] | **EnTxBErr** | **Enable Transmit Bus Error Interrupt**<br>The EnTxBErr controls the TxBErr interrupt generation. If TxBErr of MISTA register is set, and both EnTxBErr and EnTXINTR are enabled, the EMC generates the Tx interrupt to CPU. If EnTxBErr or EnTXINTR is disabled, no Tx interrupt is generated to CPU even the TxBErr of MISTA register is set.<br>1'b0: TxBErr of MISTA register is masked from Tx interrupt generation.<br>1'b1: TxBErr of MISTA register can participate in Tx interrupt generation. |
| [23] | **EnTDU** | **Enable Transmit Descriptor Unavailable Interrupt**<br>The EnTDU controls the TDU interrupt generation. If TDU of MISTA register is set, and both EnTDU and EnTXINTR are enabled, the EMC generates the Tx interrupt to CPU. If EnTDU or EnTXINTR is disabled, no Tx interrupt is generated to CPU even the TDU of MISTA register is set.<br>1'b0: TDU of MISTA register is masked from Tx interrupt generation.<br>1'b1: TDU of MISTA register can participate in Tx interrupt generation. |

| Bits | Descriptions | |
|---|---|---|
| [22] | **EnLC** | **Enable Late Collision Interrupt**<br>The EnLC controls the LC interrupt generation. If LC of MISTA register is set, and both EnLC and EnTXINTR are enabled, the EMC generates the Tx interrupt to CPU. If EnLC or EnTXINTR is disabled, no Tx interrupt is generated to CPU even the LC of MISTA register is set.<br>1'b0: LC of MISTA register is masked from Tx interrupt generation.<br>1'b1: LC of MISTA register can participate in Tx interrupt generation. |
| [21] | **EnTXABT** | **Enable Transmit Abort Interrupt**<br>The EnTXABT controls the TXABT interrupt generation. If TXABT of MISTA register is set, and both EnTXABT and EnTXINTR are enabled, the EMC generates the Tx interrupt to CPU. If EnTXABT or EnTXINTR is disabled, no Tx interrupt is generated to CPU even the TXABT of MISTA register is set.<br>1'b0: TXABT of MISTA register is masked from Tx interrupt generation.<br>1'b1: TXABT of MISTA register can participate in Tx interrupt generation. |
| [20] | **EnNCS** | **Enable No Carrier Sense Interrupt**<br>The EnNCS controls the NCS interrupt generation. If NCS of MISTA register is set, and both EnNCS and EnTXINTR are enabled, the EMC generates the Tx interrupt to CPU. If EnNCS or EnTXINTR is disabled, no Tx interrupt is generated to CPU even the NCS of MISTA register is set.<br>1'b0: NCS of MISTA register is masked from Tx interrupt generation.<br>1'b1: NCS of MISTA register can participate in Tx interrupt generation. |
| [19] | **EnEXDEF** | **Enable Defer Exceed Interrupt**<br>The EnEXDEF controls the EXDEF interrupt generation. If EXDEF of MISTA register is set, and both EnEXDEF and EnTXINTR are enabled, the EMC generates the Tx interrupt to CPU. If EnEXDEF or EnTXINTR is disabled, no Tx interrupt is generated to CPU even the EXDEF of MISTA register is set.<br>1'b0: EXDEF of MISTA register is masked from Tx interrupt generation.<br>1'b1: EXDEF of MISTA register can participate in Tx interrupt generation. |
| [18] | **EnTXCP** | **Enable Transmit Completion Interrupt**<br>The EnTXCP controls the TXCP interrupt generation. If TXCP of MISTA register is set, and both EnTXCP and EnTXINTR are enabled, the EMC generates the Tx interrupt to CPU. If EnTXCP or EnTXINTR is disabled, no Tx interrupt is generated to CPU even the TXCP of MISTA register is set.<br>1'b0: TXCP of MISTA register is masked from Tx interrupt generation.<br>1'b1: TXCP of MISTA register can participate in Tx interrupt generation. |

| Bits | Descriptions | |
|---|---|---|
| [17] | **EnTXEMP** | **Enable Transmit FIFO Underflow Interrupt**<br>The EnTXEMP controls the TXEMP interrupt generation. If TXEMP of MISTA register is set, and both EnTXEMP and EnTXINTR are enabled, the EMC generates the Tx interrupt to CPU. If EnTXEMP or EnTXINTR is disabled, no Tx interrupt is generated to CPU even the TXEMP of MISTA register is set.<br>1'b0: TXEMP of MISTA register is masked from Tx interrupt generation.<br>1'b1: TXEMP of MISTA register can participate in Tx interrupt generation. |
| [16] | **EnTXINTR** | **Enable Transmit Interrupt**<br>The EnTXINTR controls the Tx interrupt generation.<br>If EnTXINTR is enabled and TXINTR of MISTA register is high, EMC generates the Tx interrupt to CPU. If EnTXINTR is disabled, no Tx interrupt is generated to CPU even the status bits 17~24 of MISTA are set and the corresponding bits of MIEN are enabled. In other words, if S/W wants to receive Tx interrupt from EMC, this bit must be enabled. And, if S/W doesn't want to receive any Tx interrupt from EMC, disables this bit.<br>1'b0: TXINTR of MISTA register is masked and Tx interrupt generation is disabled.<br>1'b1: TXINTR of MISTA register is unmasked and Tx interrupt generation is enabled. |
| [14] | **EnCFR** | **Enable Control Frame Receive Interrupt**<br>The EnCFR controls the CFR interrupt generation. If CFR of MISTA register is set, and both EnCFR and EnTXINTR are enabled, the EMC generates the Rx interrupt to CPU. If EnCFR or EnTXINTR is disabled, no Rx interrupt is generated to CPU even the CFR of MISTA register is set.<br>1'b0: CFR of MISTA register is masked from Rx interrupt generation.<br>1'b1: CFR of MISTA register can participate in Rx interrupt generation. |
| [11] | **EnRxBErr** | **Enable Receive Bus Error Interrupt**<br>The EnRxBErr controls the RxBerr interrupt generation. If RxBErr of MISTA register is set, and both EnRxBErr and EnTXINTR are enabled, the EMC generates the Rx interrupt to CPU. If EnRxBErr or EnTXINTR is disabled, no Rx interrupt is generated to CPU even the RxBErr of MISTA register is set.<br>1'b0: RxBErr of MISTA register is masked from Rx interrupt generation.<br>1'b1: RxBErr of MISTA register can participate in Rx interrupt generation. |

| Bits | Descriptions | |
|---|---|---|
| [10] | **EnRDU** | **Enable Receive Descriptor Unavailable Interrupt**<br>The EnRDU controls the RDU interrupt generation. If RDU of MISTA register is set, and both EnRDU and EnTXINTR are enabled, the EMC generates the Rx interrupt to CPU. If EnRDU or EnTXINTR is disabled, no Rx interrupt is generated to CPU even the RDU of MISTA register is set.<br>1'b0: RDU of MISTA register is masked from Rx interrupt generation.<br>1'b1: RDU of MISTA register can participate in Rx interrupt generation. |
| [9] | **EnDEN** | **Enable DMA Early Notification Interrupt**<br>The EnDEN controls the DENI interrupt generation. If DENI of MISTA register is set, and both EnDEN and EnTXINTR are enabled, the EMC generates the Rx interrupt to CPU. If EnDEN or EnTXINTR is disabled, no Rx interrupt is generated to CPU even the DENI of MISTA register is set.<br>1'b0: DENI of MISTA register is masked from Rx interrupt generation.<br>1'b1: DENI of MISTA register can participate in Rx interrupt generation. |
| [8] | **EnDFO** | **Enable Maximum Frame Length Interrupt**<br>The EnDFO controls the DFOI interrupt generation. If DFOI of MISTA register is set, and both EnDFO and EnTXINTR are enabled, the EMC generates the Rx interrupt to CPU. If EnDFO or EnTXINTR is disabled, no Rx interrupt is generated to CPU even the DFOI of MISTA register is set.<br>1'b0: DFOI of MISTA register is masked from Rx interrupt generation.<br>1'b1: DFOI of MISTA register can participate in Rx interrupt generation. |
| [7] | **EnMMP** | **Enable More Missed Packet Interrupt**<br>The EnMMP controls the MMP interrupt generation. If MMP of MISTA register is set, and both EnMMP and EnTXINTR are enabled, the EMC generates the Rx interrupt to CPU. If EnMMP or EnTXINTR is disabled, no Rx interrupt is generated to CPU even the MMP of MISTA register is set.<br>1'b0: MMP of MISTA register is masked from Rx interrupt generation.<br>1'b1: MMP of MISTA register can participate in Rx interrupt generation. |
| [6] | **EnRP** | **Enable Runt Packet Interrupt**<br>The EnRP controls the RP interrupt generation. If RP of MISTA register is set, and both EnRP and EnTXINTR are enabled, the EMC generates the Rx interrupt to CPU. If EnRP or EnTXINTR is disabled, no Rx interrupt is generated to CPU even the RP of MISTA register is set.<br>1'b0: RP of MISTA register is masked from Rx interrupt generation.<br>1'b1: RP of MISTA register can participate in Rx interrupt generation. |

| Bits | Descriptions | |
|---|---|---|
| [5] | **EnALIE** | **Enable Alignment Error Interrupt**<br>The EnALIE controls the ALIE interrupt generation. If ALIE of MISTA register is set, and both EnALIE and EnTXINTR are enabled, the EMC generates the Rx interrupt to CPU. If EnALIE or EnTXINTR is disabled, no Rx interrupt is generated to CPU even the ALIE of MISTA register is set.<br>1'b0: ALIE of MISTA register is masked from Rx interrupt generation.<br>1'b1: ALIE of MISTA register can participate in Rx interrupt generation. |
| [4] | **EnRXGD** | **Enable Receive Good Interrupt**<br>The EnRXGD controls the RXGD interrupt generation. If RXGD of MISTA register is set, and both EnRXGD and EnTXINTR are enabled, the EMC generates the Rx interrupt to CPU. If EnRXGD or EnTXINTR is disabled, no Rx interrupt is generated to CPU even the RXGD of MISTA register is set.<br>1'b0: RXGD of MISTA register is masked from Rx interrupt generation.<br>1'b1: RXGD of MISTA register can participate in Rx interrupt generation. |
| [3] | **EnPTLE** | **Enable Packet Too Long Interrupt**<br>The EnPTLE controls the PTLE interrupt generation. If PTLE of MISTA register is set, and both EnPTLE and EnTXINTR are enabled, the EMC generates the Rx interrupt to CPU. If EnPTLE or EnTXINTR is disabled, no Rx interrupt is generated to CPU even the PTLE of MISTA register is set.<br>1'b0: PTLE of MISTA register is masked from Rx interrupt generation.<br>1'b1: PTLE of MISTA register can participate in Rx interrupt generation. |
| [2] | **EnRXOV** | **Enable Receive FIFO Overflow Interrupt**<br>The EnRXOV controls the RXOV interrupt generation. If RXOV of MISTA register is set, and both EnRXOV and EnTXINTR are enabled, the EMC generates the Rx interrupt to CPU. If EnRXOV or EnTXINTR is disabled, no Rx interrupt is generated to CPU even the RXOV of MISTA register is set.<br>1'b0: RXOV of MISTA register is masked from Rx interrupt generation.<br>1'b1: RXOV of MISTA register can participate in Rx interrupt generation. |
| [1] | **EnCRCE** | **Enable CRC Error Interrupt**<br>The EnCRCE controls the CRCE interrupt generation. If CRCE of MISTA register is set, and both EnCRCE and EnTXINTR are enabled, the EMC generates the Rx interrupt to CPU. If EnCRCE or EnTXINTR is disabled, no Rx interrupt is generated to CPU even the CRCE of MISTA register is set.<br>1'b0: CRCE of MISTA register is masked from Rx interrupt generation.<br>1'b1: CRCE of MISTA register can participate in Rx interrupt generation. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **EnRXINTR** | **Enable Receive Interrupt**<br>The EnRXINTR controls the Rx interrupt generation.<br>If EnRXINTR is enabled and RXINTR of MISTA register is high, EMC generates the Rx interrupt to CPU. If EnRXINTR is disabled, no Rx interrupt is generated to CPU even the status bits 1~14 of MISTA are set and the corresponding bits of MIEN are enabled. In other words, if S/W wants to receive Rx interrupt from EMC, this bit must be enabled. And, if S/W doesn't want to receive any Rx interrupt from EMC, disables this bit.<br>1'b0: RXINTR of MISTA register is masked and Rx interrupt generation is disabled.<br>1'b1: RXINTR of MISTA register is unmasked and Rx interrupt generation is enabled. |

## MAC Interrupt Status Register (MISTA)

The MISTA keeps much EMC statuses, like frame transmission and reception status, internal FIFO status and also NATA processing status. The statuses kept in MISTA will trigger the reception or transmission interrupt. The MISTA is a write clear register and write 1 to corresponding bit clears the status and also clears the interrupt.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MISTA | 0xB000_30B0 | R/W | MAC Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | TxBErr |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TDU | LC | TXABT | NCS | EXDEF | TXCP | TXEMP | TXINTR |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | CFR | Reserved | | RxBErr | RDU | DENI | DFOI |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MMP | RP | ALIE | RXGD | PTLE | RXOV | CRCE | RXINTR |

| Bits | Descriptions | |
|---|---|---|
| [24] | **TxBErr** | **Transmit Bus Error Interrupt**<br>The TxBErr high indicates the memory controller replies ERROR response while EMC access system memory through TxDMA during packet transmission process. Reset EMC is recommended while TxBErr status is high.<br>If the TxBErr is high and EnTxBErr of MIEN register is enabled, the TxINTR will be high. Write 1 to this bit clears the TxBErr status.<br>1'b0: No ERROR response is received.<br>1'b1: ERROR response is received. |

| Bits | Descriptions | |
|---|---|---|
| [23] | **TDU** | **Transmit Descriptor Unavailable Interrupt**<br>The TDU high indicates that there is no available Tx descriptor for packet transmission and TxDMA will stay at Halt state. Once, the TxDMA enters the Halt state, S/W must issues a write command to TSDR register to make TxDMA leave Halt state while new Tx descriptor is available.<br>If the TDU is high and EnTDU of MIEN register is enabled, the TxINTR will be high. Write 1 to this bit clears the TDU status.<br>1'b0: Tx descriptor is available.<br>1'b1: Tx descriptor is unavailable. |
| [22] | **LC** | **Late Collision Interrupt**<br>The LC high indicates the collision occurred in the outside of 64 bytes collision window. This means after the 64 bytes of a frame has transmitted out to the network, the collision still occurred. The late collision check will only be done while EMC is operating on half-duplex mode. If the LC is high and EnLC of MIEN register is enabled, the TxINTR will be high. Write 1 to this bit clears the LC status.<br>1'b0: No collision occurred in the outside of 64 bytes collision window.<br>1'b1: Collision occurred in the outside of 64 bytes collision window. |
| [21] | **TXABT** | **Transmit Abort Interrupt**<br>The TXABT high indicates the packet incurred 16 consecutive collisions during transmission, and then the transmission process for this packet is aborted. The transmission abort is only available while EMC is operating on half-duplex mode.<br>If the TXABT is high and EnTXABT of MIEN register is enabled, the TxINTR will be high. Write 1 to this bit clears the TXABT status.<br>1'b0: Packet doesn't incur 16 consecutive collisions during transmission.<br>1'b1: Packet incurred 16 consecutive collisions during transmission. |
| [20] | **NCS** | **No Carrier Sense Interrupt**<br>The NCS high indicates the MII I/F signal CRS doesn't active at the start of or during the packet transmission. The NCS is only available while EMC is operating on half-duplex mode. If the NCS is high and EnNCS of MIEN register is enabled, the TxINTR will be high. Write 1 to this bit clears the NCS status.<br>1'b0: CRS signal actives correctly.<br>1'b1: CRS signal doesn't active at the start of or during the packet transmission. |

| Bits | Descriptions | |
|---|---|---|
| [19] | **EXDEF** | **Defer Exceed Interrupt**<br>The EXDEF high indicates the frame waiting for transmission has deferred over 0.32768ms on 100Mbps mode, or 3.2768ms on 10Mbps mode. The deferral exceed check will only be done while bit NDEF of MCMDR is disabled, and EMC is operating on half-duplex mode.<br>If the EXDEF is high and EnEXDEF of MIEN register is enabled, the TxINTR will be high. Write 1 to this bit clears the EXDEF status.<br>1'b0: Frame waiting for transmission has not deferred over 0.32768ms (100Mbps) or 3.2768ms (10Mbps).<br>1'b1: Frame waiting for transmission has deferred over 0.32768ms (100Mbps) or 3.2768ms (10Mbps). |
| [18] | **TXCP** | **Transmit Completion Interrupt**<br>The TXCP indicates the packet transmission has completed correctly.<br>If the TXCP is high and EnTXCP of MIEN register is enabled, the TxINTR will be high. Write 1 to this bit clears the TXCP status.<br>1'b0: The packet transmission doesn't complete.<br>1'b1: The packet transmission has completed. |
| [17] | **TXEMP** | **Transmit FIFO Underflow Interrupt**<br>The TXEMP high indicates the TxFIFO underflow occurred during packet transmission. While the TxFIFO underflow occurred, the EMC will retransmit the packet automatically without S/W intervention. If the TxFIFO underflow occurred often, it is recommended that modify TxFIFO threshold control, the TxTHD of FFTCR register, to higher level.<br>If the TXEMP is high and EnTXEMP of MIEN register is enabled, the TxINTR will be high. Write 1 to this bit clears the TXEMP status.<br>1'b0: No TxFIFO underflow occurred during packet transmission.<br>1'b0: TxFIFO underflow occurred during packet transmission. |

| Bits | Descriptions | |
|---|---|---|
| [16] | **TXINTR** | **Transmit Interrupt**<br>The TXINTR indicates the Tx interrupt status.<br>If TXINTR high and its corresponding enable bit, EnTXINTR of MISTA register, is also high indicates the EMC generates Tx interrupt to CPU. If TXINTR is high but EnTXINTR of MISTA is disabled, no Tx interrupt is generated.<br>The TXINTR is logic OR result of the bits 17~24 in MISTA register do logic AND with the corresponding bits in MIEN register. In other words, if one of the bits 17~24 in MISTA register is high and its corresponding enable bit in MIEN register is also enabled, the TXINTR will be high. Because the TXINTR is a logic OR result, clears bits 17~24 of MISTA register makes TXINTR be cleared, too.<br>1'b0: No status of bits 17~24 in MISTA is set or no enable of bits 17~24 in MIEN is turned on.<br>1'b1: At least one status of bits 17~24 in MISTA is set and its corresponding enable bit is turned on. |
| [14] | **CFR** | **Control Frame Receive Interrupt**<br>The CFR high indicates EMC receives a flow control frame. The CFR only available while EMC is operating on full duplex mode.<br>If the CFR is high and EnCFR of MIEN register is enabled, the RxINTR will be high. Write 1 to this bit clears the CFR status.<br>1'b0: The EMC doesn't receive the flow control frame.<br>1'b1: The EMC receives a flow control frame. |
| [11] | **RxBErr** | **Receive Bus Error Interrupt**<br>The RxBErr high indicates the memory controller replies ERROR response while EMC access system memory through RxDMA during packet reception process. Reset EMC is recommended while RxBErr status is high.<br>If the RxBErr is high and EnRxBErr of MIEN register is enabled, the RxINTR will be high. Write 1 to this bit clears the RxBErr status.<br>1'b0: No ERROR response is received.<br>1'b1: ERROR response is received. |

| Bits | Descriptions | |
|---|---|---|
| [10] | **RDU** | **Receive Descriptor Unavailable Interrupt**<br>The RDU high indicates that there is no available Rx descriptor for packet reception and RxDMA will stay at Halt state. Once, the RxDMA enters the Halt state, S/W must issues a write command to RSDR register to make RxDMA leave Halt state while new Rx descriptor is available.<br>If the RDU is high and EnRDU of MIEN register is enabled, the RxINTR will be high. Write 1 to this bit clears the RDU status.<br>1'b0: Rx descriptor is available.<br>1'b1: Rx descriptor is unavailable. |
| [9] | **DENI** | **DMA Early Notification Interrupt**<br>The DENI high indicates the EMC has received the Length/Type field of the incoming packet.<br>If the DENI is high and EnDENI of MIEN register is enabled, the RxINTR will be high. Write 1 to this bit clears the DENI status.<br>1'b0: The Length/Type field of incoming packet has not received yet.<br>1'b1: The Length/Type field of incoming packet has received. |
| [8] | **DFOI** | **Maximum Frame Length Interrupt**<br>The DFOI high indicates the length of the incoming packet has exceeded the length limitation configured in DMARFC register and the incoming packet is dropped. If the DFOI is high and EnDFO of MIEN register is enabled, the RxINTR will be high. Write 1 to this bit clears the DFOI status.<br>1'b0: The length of the incoming packet doesn't exceed the length limitation configured in DMARFC.<br>1'b1: The length of the incoming packet has exceeded the length limitation configured in DMARFC. |
| [7] | **MMP** | **More Missed Packet Interrupt**<br>The MMP high indicates the MPCNT, Missed Packet Count, has overflow. If the MMP is high and EnMMP of MIEN register is enabled, the RxINTR will be high. Write 1 to this bit clears the MMP status.<br>1'b0: The MPCNT has not rolled over yet.<br>1'b1: The MPCNT has rolled over yet. |

| Bits | Descriptions | |
|---|---|---|
| [6] | **RP** | **Runt Packet Interrupt**<br>The RP high indicates the length of the incoming packet is less than 64 bytes, and the packet is dropped. If the ARP of MCMDR register is set, the short packet is regarded as a good packet and RP will not be set.<br>If the RP is high and EnRP of MIEN register is enabled, the RxINTR will be high. Write 1 to this bit clears the RP status.<br>1'b0: The incoming frame is not a short frame or S/W wants to receive a short frame.<br>1'b1: The incoming frame is a short frame and dropped. |
| [5] | **ALIE** | **Alignment Error Interrupt**<br>The ALIE high indicates the length of the incoming frame is not a multiple of byte. If the ALIE is high and EnALIE of MIEN register is enabled, the RxINTR will be high. Write 1 to this bit clears the ALIE status.<br>1'b0: The frame length is a multiple of byte.<br>1'b1: The frame length is not a multiple of byte. |
| [4] | **RXGD** | **Receive Good Interrupt**<br>The RXGD high indicates the frame reception has completed.<br>If the RXGD is high and EnRXGD of MIEN register is enabled, the RxINTR will be high. Write 1 to this bit clears the RXGD status.<br>1'b0: The frame reception has not complete yet.<br>1'b1: The frame reception has completed. |
| [3] | **PTLE** | **Packet Too Long Interrupt**<br>The PTLE high indicates the length of the incoming packet is greater than 1518 bytes and the incoming packet is dropped. If the ALP of MCMDR register is set, the long packet will be regarded as a good packet and PTLE will not be set.<br>If the PTLE is high and EnPTLE of MIEN register is enabled, the RxINTR will be high. Write 1 to this bit clears the PTLE status.<br>1'b0: The incoming frame is not a long frame or S/W wants to receive a long frame.<br>1'b1: The incoming frame is a long frame and dropped. |

| Bits | Descriptions | |
|---|---|---|
| [2] | **RXOV** | **Receive FIFO Overflow Interrupt**<br>The RXOV high indicates the RxFIFO overflow occurred during packet reception. While the RxFIFO overflow occurred, the EMC drops the current receiving packer. If the RxFIFO overflow occurred often, it is recommended that modify RxFIFO threshold control, the RxTHD of FFTCR register, to higher level.<br>If the RXOV is high and EnRXOV of MIEN register is enabled, the RxINTR will be high. Write 1 to this bit clears the RXOV status.<br>1'b0: No RxFIFO overflow occurred during packet reception.<br>1'b0: RxFIFO overflow occurred during packet reception. |
| [1] | **CRCE** | **CRC Error Interrupt**<br>The CRCE high indicates the incoming packet incurred the CRC error and the packet is dropped. If the AEP of MCMDR register is set, the CRC error packet will be regarded as a good packet and CRCE will not be set.<br>If the CRCE is high and EnCRCE of MIEN register is enabled, the RxINTR will be high. Write 1 to this bit clears the CRCE status.<br>1'b0: The frame doesn't incur CRC error.<br>1'b1: The frame incurred CRC error. |
| [0] | **RXINTR** | **Receive Interrupt**<br>The RXINTR indicates the Rx interrupt status.<br>If RXINTR high and its corresponding enable bit, EnRXINTR of MISTA register, is also high indicates the EMC generates Rx interrupt to CPU. If RXINTR is high but EnRXINTR of MISTA is disabled, no Rx interrupt is generated.<br>The RXINTR is logic OR result of the bits 1~14 in MISTA register do logic AND with the corresponding bits in MIEN register. In other words, if one of the bits 1~14 in MISTA register is high and its corresponding enable bit in MIEN register is also enabled, the RXINTR will be high.<br>Because the RXINTR is a logic OR result, clears bits 1~14 of MISTA register makes RXINTR be cleared, too.<br>1'b0: No status of bits 1~14 in MISTA is set or no enable of bits 1~14 in MIEN is turned on.<br>1'b1: At least one status of bits 1~14 in MISTA is set and its corresponding enable bit is turned on. |

## MAC General Status Register (MGSTA)

The MGSTA also keeps the statuses of EMC. But the statuses in the MGSTA will not trigger any interrupt. The MGSTA is a write clear register and write 1 to corresponding bit clears the status.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MGSTA | 0xB000_30B4 | R/W | MAC General Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | TXHA | SQE | PAU | DEF |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CCNT | | | | Reserved | RFFull | RXHA | CFR |

| Bits | Descriptions | |
|---|---|---|
| [11] | **TXHA** | **Transmission Halted**<br>Default Value: 1'b0<br>The TXHA high indicates the next normal packet transmission process will be halted because the bit TXON of MCMDR is disabled be S/W.<br>1'b0: Next normal packet transmission process will go on.<br>1'b1: Next normal packet transmission process will be halted. |
| [10] | **SQE** | **Signal Quality Error**<br>Default Value: 1'b0<br>The SQE high indicates the SQE error found at end of packet transmission on 10Mbps half-duplex mode. The SQE error check will only be done while both bit EnSQE of MCMDR is enabled and EMC is operating on 10Mbps half-duplex mode.<br>1'b0: No SQE error found at end of packet transmission.<br>1'b0: SQE error found at end of packet transmission. |

| Bits | Descriptions | |
|---|---|---|
| [9] | **PAU** | **Transmission Paused**<br>Default Value: 1'b0<br>The PAU high indicates the next normal packet transmission process will be paused temporally because EMC received a PAUSE control frame, or S/W set bit SDPZ of MCMDR and make EMC to transmit a PAUSE control frame out.<br>1'b0: Next normal packet transmission process will go on.<br>1'b1: Next normal packet transmission process will be paused. |
| [8] | **DEF** | **Deferred Transmission**<br>Default Value: 1'b0<br>The DEF high indicates the packet transmission has deferred once. The DEF is only available while EMC is operating on half-duplex mode.<br>1'b0: Packet transmission doesn't defer.<br>1'b1: Packet transmission has deferred once. |
| [7:4] | **CCNT** | **Collision Count**<br>Default Value: 4'h0<br>The CCNT indicates the how many collisions occurred consecutively during a packet transmission. If the packet incurred 16 consecutive collisions during transmission, the CCNT will be 4'h0 and bit TXABT will be set to 1. |
| [2] | **RFFull** | **RxFIFO Full**<br>Default Value: 1'b0<br>The RFFull indicates the RxFIFO is full due to four 64-byte packets are kept in RxFIFO and the following incoming packet will be dropped.<br>1'b0: The RxFIFO is not full.<br>1'b1: The RxFIFO is full and the following incoming packet will be dropped. |
| [1] | **RXHA** | **Receive Halted**<br>Default Value: 1'b0<br>The RXHA high indicates the next normal packet reception process will be halted because the bit RXON of MCMDR is disabled be S/W.<br>1'b0: Next normal packet reception process will go on.<br>1'b1: Next normal packet reception process will be halted. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **CFR** | **Control Frame Received**<br>Default Value: 1'b0<br>The CFR high indicates EMC receives a flow control frame. The CFR only available while EMC is operating on full duplex mode.<br>1'b0: The EMC doesn't receive the flow control frame.<br>1'b1: The EMC receives a flow control frame. |

## Missed Packet Count Register (MPCNT)

The MPCNT keeps the number of packets that were dropped due to various types of receive errors. The MPCNT is a read clear register. In addition, S/W also can write an initial value to MPCNT and the missed packet counter will start counting from that initial value. If the missed packet counter is overflow, the MMP of MISTA will be set.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MPCNT | 0xB000_30B8 | R/W | Missed Packet Count Register | 0x0000_7FFF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MPC | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MPC | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **MPC** | **Miss Packet Count**<br>Default Value: 16'h7FFF<br>The MPC indicates the number of packets that were dropped due to various types of receive errors. The following type of receiving error makes missed packet counter increase:<br>▪ Incoming packet is incurred RxFIFO overflow.<br>▪ Incoming packet is dropped due to RXON is disabled.<br>Incoming packet is incurred CRC error. |

## MAC Receive Pause Count Register (MRPC)

The EMC of NUC950ADN supports the PAUSE control frame reception and recognition. If EMC received a PAUSE control frame, the operand field of the PAUSE control frame will be extracted and stored in the MRPC register. The MRPC register will keep the same while Tx of EMC is pausing due to the PAUSE control frame is received. The MRPC is read only and write to this register has no effect.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MRPC | 0xB000_30BC | R | MAC Receive Pause Count Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MRPC | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MRPC | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **MRPC** | **MAC Receive Pause Count**<br>Default Value: 16'h0<br>The MRPC keeps the operand field of the PAUSE control frame. It indicates how many slot time (512 bit time) the Tx of EMC will be paused. |

## MAC Receive Pause Current Count Register (MRPCC)

The EMC of NUC950ADN supports the PAUSE control frame reception and recognition. If EMC received a PAUSE control frame, the operand field of the PAUSE control frame will be extracted and stored into a down count timer. The MRPCC shows the current value of that down count timer for S/W to know how long the Tx of EMC will be paused. The MRPCC is read only and write to this register has no effect.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MRPCC | 0xB000_30C0 | R | MAC Receive Pause Current Count Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MRPCC | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MRPCC | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **MRPCC** | **MAC Receive Pause Current Count**<br>Default Value: 16'h0<br>The MRPCC shows the current value of that down count timer. If a new PAUSE control frame is received before the timer count down to zero, the new operand of the PAUSE control frame will be stored into the down count timer and the timer starts count down from the new value. |

## MAC Remote Pause Count Register (MREPC)

The EMC of NUC950ADN supports the PAUSE control frame transmission. After the PAUSE control frame is transmitted out completely, a timer starts to count down from the value of operand of the transmitted PAUSE control frame. The MREPC shows the current value of this down count timer. The MREPC is read only and write to this register has no effect.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MREPC | 0xB000_30C4 | R | MAC Remote Pause Count Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MREPC | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MREPC | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **MREPC** | **MAC Remote Pause Count**<br>Default Value: 16'h0<br>The MREPC shows the current value of the down count timer that starts to count down from the value of operand of the transmitted PAUSE control frame. |

## DMA Receive Frame Status Register (DMARFS)

The DMARFS is used to keep the Length/Type field of each incoming Ethernet packet. This register is write-clear and writes 1 to corresponding bit clears the bit.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| DMARFS | 0xB000_30C8 | R/W | DMA Receive Frame Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RXFLT | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXFLT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **RXFLT** | **Receive Frame Length/Type**<br>Default Value: 16'h0<br>The RXFLT keeps the Length/Type field of each incoming Ethernet packet. If the bit EnDEN of MIEN is enabled and the Length/Type field of incoming packet has received, the bit DENI of MISTA will be set and trigger interrupt. And, the content of Length/Type field will be stored in RXFLT. |

## Current Transmit Descriptor Start Address Register (CTXDSA)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CTXDSA | 0xB000_30CC | R | Current Transmit Descriptor Start Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CTXDSA | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CTXDSA | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CTXDSA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTXDSA | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **CTXDSA** | **Current Transmit Descriptor Start Address**<br>Default Value: 32'h0<br>The CTXDSA keeps the start address of Tx descriptor that is used by TxDMA currently. The CTXDSA is read only and write to this register has no effect. |

## Current Transmit Buffer Start Address Register (CTXBSA)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CTXBSA | 0xB000_30D0 | R | Current Transmit Buffer Start Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CTXBSA | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CTXBSA | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CTXBSA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTXBSA | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **CTXBSA** | **Current Transmit Buffer Start Address**<br>Default Value: 32'h0<br>The CTXDSA keeps the start address of Tx frame buffer that is used by TxDMA currently. The CTXBSA is read only and write to this register has no effect. |

## Current Receive Descriptor Start Address Register (CRXDSA)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRXDSA | 0xB000_30D4 | R | Current Receive Descriptor Start Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CRXDSA | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CRXDSA | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CRXDSA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CRXDSA | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **CRXDSA** | **Current Receive Descriptor Start Address**<br>Default Value: 32'h0<br>The CRXDSA keeps the start address of Rx descriptor that is used by RxDMA currently. The CRXDSA is read only and write to this register has no effect. |

## Current Receive Buffer Start Address Register (CRXBSA)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRXBSA | 0xB000_30D8 | R | Current Receive Buffer Start Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CRXBSA | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CRXBSA | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CRXBSA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CRXBSA | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **CRXBSA** | **Current Receive Buffer Start Address**<br>Default Value: 32'h0<br>The CRXBSA keeps the start address of Rx frame buffer that is used by RxDMA currently. The CRXBSA is read only and write to this register has no effect. |

### 7.5.4 Operation Notes

#### MII Management Interface

The operation mode between EMC and external PHY must be identically. Consequently, S/W has to access control register of external PHY through MII management interface to get operation information of PHY. To issue MII management command to access external PHY, the MIID and MIIDA registers can be used. And, while using MII management interface, the EnMDC of MCMDR register must be set to high.

#### EMC Initial

If S/W wants to enable EMC for packet transmission and reception, the TXON and RXON of MCMDR register must be enabled. But, before enabling TXON and RXON, the following issues must be noted.

For packet transmission, the Tx descriptor link list and Tx frame buffer must be prepared and TXDLSA must be configured.

For incoming packet destination MAC address recognition, the CAMCMR, CAMEN, CAMxM and CAMxL registers must be configured. For incoming packet's buffering, the Rx descriptor link list and Rx frame buffer must be prepared and RXDLSA register must be configured.

Besides, the interrupt status that S/W wants to know must be enabled through MIEN register.

Finally, the EMC operation mode control bits of MCMDR must be configured and TXON and RXON must be enabled.

#### MAC Interrupt Status Register (MISTA)

The MISTA register keeps the status of EMC operation. It is recommended that S/W must enable four interrupt statuses at least. They are TxBErr, RxBErr, TDU and RDU.

While EMC accesses memory, it reports the memory error through TxBErr or TxBErr status. If any of them actives, the reset EMC is recommended.

For packet transmission, a valid Tx descriptor is required, and for packet reception, a valid Rx one is. If EMC cannot find a valid Tx or Rx descriptor, it sets TDU or RDU to high respectively. After S/W releases a valid Tx or Rx descriptor to EMC, writing TSDR or RSDR register to enable packet transmission and reception again is needed.

#### Pause Control Frame Transmission

The EMC supports the PAUSE control frame transmission for flow control while EMC is operating on full-duplex mode. The register CAM13M, CAM13L, CAM14M, CAM14L, CAM15M and CAM15L are designed for this purpose.

For PAUSE control frame transmission, first, S/W must configure destination MAC address of control frame into the register pair {CAM13M, CAM13L}, source MAC address into the register pair {CAM14M, CAM14L}, and configure length/type, op-code and operand of control frame into the register pair {CAM15M, CAM15L}. The bit CAM13EN, CAM14EN and CAM15EN of CAMEN register are also needed be enabled. Then, set bit SDPZ of MCMDR register to high to enable PAUSE control frame transmission. After the PAUSE control frame transmission completed, the SDPZ will be cleared automatically.

#### Internal Loop-back

If the LBK of MCMDR register is set, the EMC operates on internal loop-back mode. While EMC operates on internal loop-back mode, it also means EMC operates on full-duplex mode, and the value of FDUP of

MCMDR register is ignored.

## 7.6 GDMA Controller

### 7.6.1 Overview & Features

The chip has a two-channel general DMA controller with or without descriptor fetch operation, called the GDMA. The two-channel GDMA performs the following data transfers without the CPU intervention:

- Memory-to-memory (memory to/from memory)
- Memory –to – IO
- IO- to –memory

The on-chip GDMA can be started by the software. Software can also be used to restart the GDMA operation after it has been stopped. The CPU can recognize the completion of a GDMA operation by software polling or when it receives an internal GDMA interrupt. The GDMA controller can increment source or destination address, decrement them as well, and conduct 8-bit (byte), 16-bit (half-word), or 32-bit (word) data transfers.

### 7.6.2 GDMA Non-Descriptor Functional Description

The GDMA directly transfers data between source and destination. The GDMA starts to transfer data after it receives service requests from software. When the entire data have been transferred completely, the GDMA becomes idle. Nevertheless, if another transfer is needed, then the GDMA must be programmed again.

### 7.6.3 GDMA Descriptor Functional Description

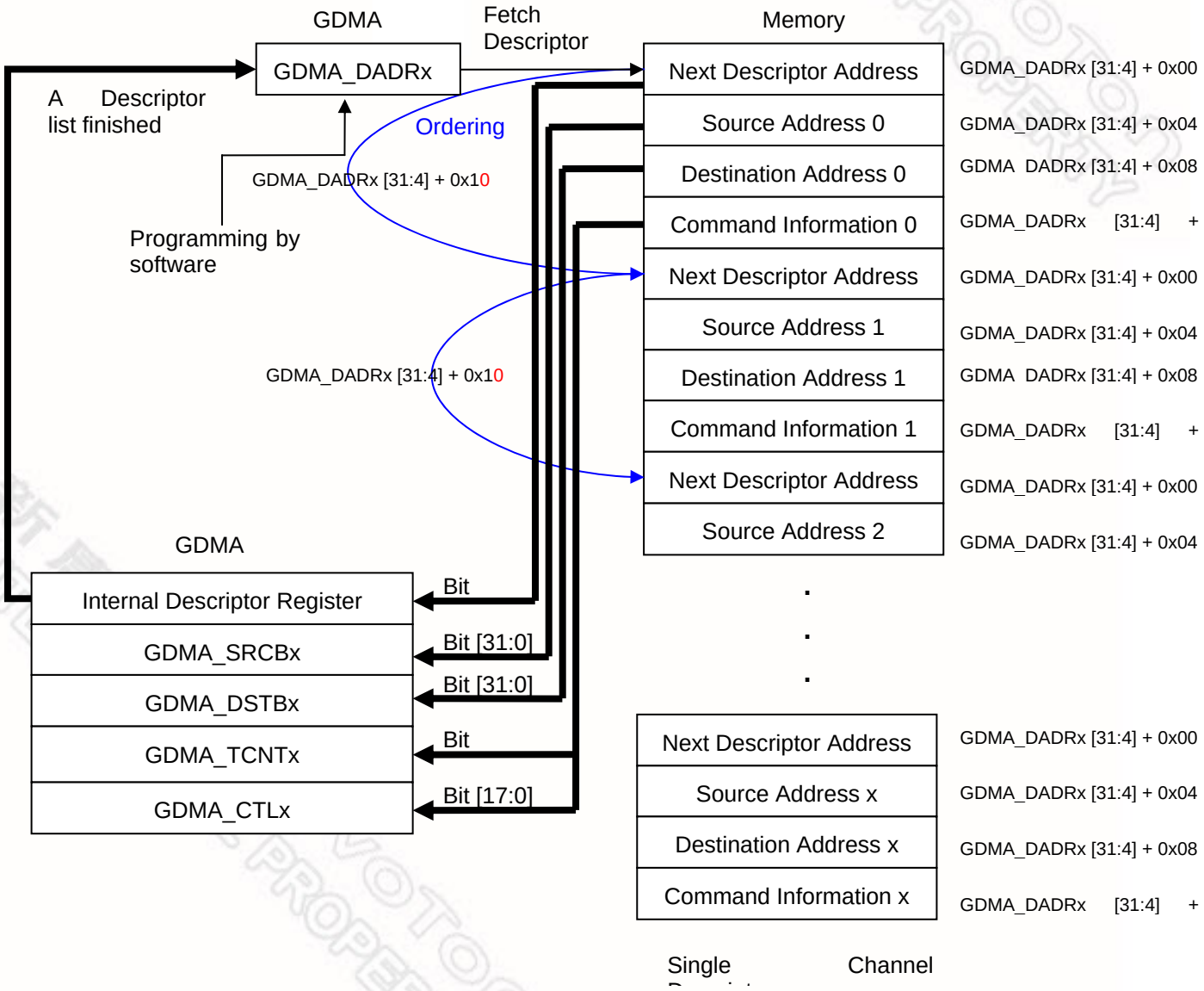
The descriptor-fetch function works when run-bit (bit-3) is set and non-dsptrmode-bit (bit-2) is cleared in Descriptor Register (GDMA_DADRx) and the GDMA_CTLx bit setting as following table. The Non-descriptor-fetch function works when software triggers the [softreq] bit (bit-16) and the [gdmaen] bit (bit-0) in GDMA_CTLx Register. If the [softreq] set to zero and the [GDMAMS] (bit2-3) set as 01 or 10 will start the I/O to memory function. Software can also be used to restart the GDMA operation after it has been stopped. The CPU can recognize the completion of a GDMA operation by software polling or when it receives an internal GDMA interrupt. The GDMA controller can increase source or destination address, decrease them as well, and conduct 8-bit (byte), 16-bit (half-word), or 32-bit (word) data transfers.

| Operation Mode | Enable bit |
|---|---|
| Non-Descriptor Mode with SW Enable | GDMA_CTLx : gdmaen[0] softreq[16] gdmams[3:2] |
| Non-Descriptor Mode with I/O Enable | GDMA_CTLx : gdmaen[0] gdmams[3:2] |

| | |
|---|---|
| Descriptor Mode with SW Enable | GDMA_DADRx : run[3] non-dsptrmode[2];<br>GDMA_CTLx in Descriptor List : gdmaen[0] gdmams[3:2] |
| Descriptor Mode with I/O Enable | GDMA_DADRx : run[3] non-dsptrmode[2];<br>GDMA_CTLx in Descriptor List : gdmaen[0] gdmams[3:2] |

#### 7.6.3.1 Descriptor Fetch Function

**The Illustration of Descriptor list fetches:**



**Descriptor-based function (GDMA_DADRx [NON_DSPTRMODE] = 0) operate in the following condition:**

**Memory to Memory**

1. Software can write a value 0x04 to current GDMA_DADRx register to reset the register and disable Descriptor based function first.
2. Then software can program the bits of [Descriptor Address], [RUN], [NON_DSPTRMODE] and [ORDEN] to the GDMA_DADRx register to enable Descriptor based function. (The Descriptor can only work when the [RUN] [3] is set and [NON_DSPTRMODE] [2] bit is cleared properly.)
3. After sets current GDMA_DADRx register, the GDMA will fetch four-word information from memory immediately which contains the next Descriptor address, Source Address, Destination Address and Command information. (Command information consists of control and counter registers)

NOTE: GDMA will read the descriptor list from memory such the diagram above and write back to GDMA internal register (next GDMA_DADRx), GDMA_SRCBx, GDMA_DSTBx, GDMA_CTLx and GDMA_TCNTx registers. The most important one of write back is command information, which will separate some bits of command information into control and counter registers respectively. The first fourteen bits of the MSB of the Command information in Descriptor list will be written back to GDMA_TCNTx register, and the others bits of the Command information will be written back to GDMA_CTLx register. The control register part of the Command information will update the GDMA_CTLx register during every descriptor fetch. The allocation of command information is described at GDMA Register Descriptions.

**The Allocation of Command Information in Descriptor List:**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GDMA_TCNTx[13:6] ← Command Info[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GDMA_TCNTx[5:0] ← Command Info[23:18] | | | | | | BLOCK | SOFTREQ |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TWS | | RESERVED | | D_INTS | D_INTS | RESERVED | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SAFIX | DAFIX | SADIR | DADIR | GDMAMS | | BME | GDMAEN |

4. GDMA will depend on the information to request a bus ownership and start the data transfer when GDMA has gotten a bus grant from the arbiter, otherwise, it will wait until get bus grant. The data transfer direction is dependent on the Control register.

5. The GDMA transfers data and releases bus at every burst transfer. The GDMA will stop transfer for current descriptor when the counter is decreased to zero. The current GDMA_DADRx will be updated by next GDMA_DADRx at end of each descriptor transfer.

6. The GDMA is running consecutively unless the next GDMA_DADRx[RUN] bit is zero or interrupt status bit of GDMA_INTCS register is cleared. The CPU can recognize the completion of a GDMA descriptor fetch operation by polling the current GDMA_DADRx[NON_DSPTRMODE] bit or set the GDMA_CTLx[D_INTS] to receive a interrupt from GDMA.(Note: The recommendation is the [NON_DSPTRMODE] bit in list is set at

the same time)

7. When an error occurs in the descriptor operation, GDMA will clear [RUN] bit and stop channel operation immediately. Software can reset the channel, and sets the current GDMA_DADRx [RUN] register to start again.

**Memory to I/O and I/O to Memory**

1. Software must set the [REQ_ATV], [ACK_ATV] and [GDMAMS] bits in GDMA_CTLx register corresponding to I/O pin with pull high or pull low properly first, and then set the current GDMA_DADRx to start the I/O to Memory with descriptor fetch transfer.
2. The descriptor lists stop transfer until the RUN bit was zero in descriptor list when external I/O request triggered once. The RUN bit can be set when external I/O request triggered again under the NON_DSPTRMODE bit was zero in descriptor list. The trigger period of the external I/O has a timing limitation whatever the GDMA was in single or burst mode, and the periodic trigger of the external I/O must be less than 38 MCLK.
3. Each GDMA lists can operate after clearing interrupt status. The descriptor lists stop transfer until the RUN bit was zero or interrupt status was set.
4. The next Descriptor address, Source Address, Destination Address and Command information must be set properly in every Descriptor list. Especially, every bit of the Command information will update the GDMA_CTLx and GDMA_TCNTx registers at every initiation of descriptor list.

NOTE: The [BLOCK] bit of GDMA_CTLx register is disabled when the descriptor mode of the I/O to memory is enabled.

NOTE: GDMA can change mode with following description:

Descriptor-fetch of each channel can be stopped until the current transfer list done. Software can change Descriptor mode to Non-Descriptor mode by writing 0x04 to GDMA_DADRx register during the current descriptor transfer operating.

Non-Descriptor fetch can be stopped until current transfer count finished when software programs the GDMA_CTLx register with gdmaen bit cleared or softreq cleared.

NOTE: Once software programs the current GDMA_DADRx register, GDMA will fetch the descriptor list from memory and fill the data to next GDMA_DADRx, current GDMA_SRCBx, current GDMA_DSTBx, current GDMA_CTLx and current GDMA_TCNTx registers automatically. The fourth word in descriptor list includes the information for GDMA_CTLx and GDMA_TCNTx registers.

NOTE: The descriptor fetch function only occurs when current GDMA_DADRx [RUN] bit is set and GDMA_DADRx [NON_DSPTRMODE] is cleared. The current GDMA_DADRx will be updated by next GDMA_DADRx at every descriptor stops.

#### 7.6.3.2 Ordering function in Descriptor fetch mode

This function determines the source of next descriptor address. If [ORDEN] is set, the GDMA controller fetches the next descriptor from current GDMA_DADRx [Descriptor Address] + 16 bytes.

If this bit is cleared, GDMA fetches the next descriptor from the current GDMA_DADRx [Descriptor Address].

GDMA_DADRx [ORDEN] is only relevant to descriptor-fetch function (GDMA_DADRx [NON_DSPTRMODE] = 0).

#### 7.6.3.3 Channel Reset

The Channel reset is turned on when the bit-0 of GDMA_DADRx is set. This function will clear all status and stop the descriptor based function relative to individual channel. The GDMA_DADRx register value is 0x05h when reset bit is set.

#### 7.6.3.4 Non-Descriptor Fetch Function

The non-descriptor-fetch function will take place when current GDMA_DADRx [NON_DSPTRMODE] is set and the GDMA_DADRx register will have no any intention for the GDMA controller.

The default value of GDMA_DADRx is 0x04. Software can clear GDMA_DADRx with value 0x04 as well. In this mode, software should write a valid source address to the GDMA_SRCBx register, a destination address to the GDMA_DSTBx register, and a transfer count to the GDMA_TCNTx register. Next, the GDMA_CTLx of [gdmaen] and [softreq] bits must be set. A non-descriptor fetch is performed when bus granted. After transferring a number of bytes or words correspond with burst mode or not, the channel either waits for the next request or continues with the data transfer until the GDMA_CTCNTx reaches zero. When GDMA_CTCNTx reaches zero, the channel stops operation.

When an error occurs during the GDMA operation, the channel stops unless software clears the error condition and sets the GDMA_CTLx of [gdmaen] and [softreq] bits field to start again.

### 7.6.4 GDMA Register Map

**R**: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GDMA_BA = 0xB000_4000** | | | | |
| **GDMA_CTL0** | 0xB000_4000 | R/W | Channel 0 Control Register | 0x0000_0000 |
| **GDMA_SRCB0** | 0xB000_4004 | R/W | Channel 0 Source Base Address Register | 0x0000_0000 |
| **GDMA_DSTB0** | 0xB000_4008 | R/W | Channel 0 Destination Base Address Register | 0x0000_0000 |
| **GDMA_TCNT0** | 0xB000_400C | R/W | Channel 0 Transfer Count Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **GDMA_CSRC0** | 0xB000_4010 | R | Channel 0 Current Source Address Register | 0x0000_0000 |
| **GDMA_CDST0** | 0xB000_4014 | R | Channel 0 Current Destination Address Reg. | 0x0000_0000 |
| **GDMA_CTCNT0** | 0xB000_4018 | R | Channel 0 Current Transfer Count Register | 0x0000_0000 |
| **GDMA_DADR0** | 0xB000_401C | R/W | Channel 0 Descriptor Address Register | 0x0000_0004 |
| **GDMA_CTL1** | 0xB000_4020 | R/W | Channel 1 Control Register | 0x0000_0000 |
| **GDMA_SRCB1** | 0xB000_4024 | R/W | Channel 1 Source Base Address Register | 0x0000_0000 |
| **GDMA_DSTB1** | 0xB000_4028 | R/W | Channel 1 Destination Base Address Register | 0x0000_0000 |
| **GDMA_TCNT1** | 0xB000_402C | R/W | Channel 1 Transfer Count Register | 0x0000_0000 |
| **GDMA_CSRC1** | 0xB000_4030 | R | Channel 1 Current Source Address Register | 0x0000_0000 |
| **GDMA_CDST1** | 0xB000_4034 | R | Channel 1 Current Destination Address Reg. | 0x0000_0000 |
| **GDMA_CTCNT1** | 0xB000_4038 | R | Channel 1 Current Transfer Count Register | 0x0000_0000 |
| **GDMA_DADR1** | 0xB000_403C | R/W | Channel 1 Descriptor Address Register | 0x0000_0004 |
| **GDMA_INTBUF0** | 0xB000_4080 | R | GDMA Internal Buffer Word 0 | 0x0000_0000 |
| **GDMA_INTBUF1** | 0xB000_4084 | R | GDMA Internal Buffer Word 1 | 0x0000_0000 |
| **GDMA_INTBUF2** | 0xB000_4088 | R | GDMA Internal Buffer Word 2 | 0x0000_0000 |
| **GDMA_INTBUF3** | 0xB000_408C | R | GDMA Internal Buffer Word 3 | 0x0000_0000 |
| **GDMA_INTBUF4** | 0xB000_4090 | R | GDMA Internal Buffer Word 4 | 0x0000_0000 |
| **GDMA_INTBUF5** | 0xB000_4094 | R | GDMA Internal Buffer Word 5 | 0x0000_0000 |
| **GDMA_INTBUF6** | 0xB000_4098 | R | GDMA Internal Buffer Word 6 | 0x0000_0000 |
| **GDMA_INTBUF7** | 0xB000_409C | R | GDMA Internal Buffer Word 7 | 0x0000_0000 |
| **GDMA_INTCS** | 0xB000_40A0 | R/W | Interrupt Control and Status Register (2 Channels) | 0x0000_0000 |

## Channel 0 / 1 Control Register (GDMA_CTL0, GDMA_CTL1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GDMA_CTL0 | 0xB000_4000 | R/W | Channel 0 Control Register | 0x0000_0000 |
| GDMA_CTL1 | 0xB000_4020 | R/W | Channel 1 Control Register | 0x0000_0000 |

The control registers has two formats for descriptor fetch and non-descriptor fetch function respectively. The functionality of each control bit is described in following table.

**1. Non-Descriptor fetches Mode**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | RESERVED | RESERVED |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | SABNDERR | DABNDERR | RESERVED | AUTOIEN | RESERVED | BLOCK | SOFTREQ |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | TWS | | RESERVED | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SAFIX | DAFIX | SADIR | DADIR | GDMAMS | | BME | GDMAEN |

**2. Descriptor fetches Mode**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | RESERVED | RESERVED |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | SABNDERR | DABNDERR | RESERVED | | | BLOCK | SOFTREQ |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | TWS | | RESERVED | D_INTS | RESERVED | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SAFIX | DAFIX | SADIR | DADIR | GDMAMS | | BME | GDMAEN |

NOTE:

- The bit [REQ_ATV] and [ACK_ATV] must be set first before using I/O to Memory mode with Descriptor fetch transfer. These two bits cannot do any setup in command information within descriptor list configuration. The [SABNDERR], [DABNDERR], [GDMAERR] can also be read at descriptor fetch mode.
- Regardless of GDMA operate in descriptor mode or non-descriptor mode, when transfer width is 16-bit (half word) and the address with decrement function enable for starting source address or destination address or both are used should set the least two bit of addresses is 0xF.

**Control Register of Non-Descriptor fetches Mode:**

| Bits | Descriptions | |
|---|---|---|
| [22] | **SABNDERR** | **Source Address Boundary Alignment Error Flag**<br>If TWS [13:12]=10, GDMA_SRCB [1:0] should be 00<br>If TWS [13:12]=01, GDMA_SRCB [0] should be 0<br>Except the SADIR function enabled.<br>The address boundary alignment should be depended on TWS [13:12].<br>0 = the GDMA_SRCB is on the boundary alignment.<br>1 = the GDMA_SRCB not on the boundary alignment<br>The SABNDERR register bits just can be read only. |
| [21] | **DABNDERR** | **Destination Address Boundary Alignment Error Flag**<br>If TWS [13:12]=10, GDMA_DSTB [1:0] should be 00<br>If TWS [13:12]=01, GDMA_DSTB [0] should be 0<br>Except the SADIR function enabled.<br>The address boundary alignment should be depended on TWS [13:12].<br>0 = the GDMA_DSTB is on the boundary alignment.<br>1 = the GDMA_DSTB not on the boundary alignment<br>The DABNDERR register bits just can be read only. |
| [19] | **AUTOIEN** | **Auto initialization Enable**<br>0 = Disables auto initialization<br>1 = Enables auto initialization, the GDMA_CSRC0/1, GDMA_CDST0/1, and GDMA_CTCNT0/1 registers are updated by the GDMA_SRC0/1, GDMA_DST0/1, and GDMA_TCNT0/1 registers automatically when transfer is complete.<br>GDMA will start another transfer when SOFTREQ set again. |
| [17] | **BLOCK** | **Bus Lock**<br>0 = Unlocks the bus during the period of transfer<br>1 = locks the bus during the period of transfer |

| Bits | Descriptions | |
|---|---|---|
| [16] | **SOFTREQ** | **Software Triggered GDMA Request**<br>Software can request the GDMA transfer service by setting this bit to 1. This bit is automatically cleared by hardware when the transfer is completed. This bit is available only while GDMAMS [3:2] register bits are set on software mode (memory to memory and memory to I/O). |
| [13:12] | **TWS** | **Transfer Width Select**<br>00 = One byte (8 bits) is transferred for every GDMA operation<br>01 = One half-word (16 bits) is transferred for every GDMA operation<br>10 = One word (32 bits) is transferred for every GDMA operation<br>11 = Reserved<br>The GDMA_SCRB and GDMA_DSTB should be alignment under the TWS selection |
| [7] | **SAFIX** | **Source Address Fixed**<br>0 = Source address is changed during the GDMA operation<br>1 = Do not change the destination address during the GDMA operation. This feature can be used when data were transferred from a single source to multiple destinations. |
| [6] | **DAFIX** | **Destination Address Fixed**<br>0 = Destination address is changed during the GDMA operation<br>1 = Do not change the destination address during the GDMA operation. This feature can be used when data were transferred from multiple sources to a single destination. |
| [5] | **DADIR** | **Source Address Direction**<br>0 = Source address is incremented successively<br>1 = Source address is decremented successively |
| [4] | **DADIR** | **Destination Address Direction**<br>0 = Destination address is incremented successively<br>1 = Destination address is decremented successively |
| [3:2] | **GDMAMS** | **GDMA Mode Select**<br>00 = Software mode<br>01 = Reserved<br>10 = Reserved<br>11 = Reserved |

| Bits | Descriptions | |
|---|---|---|
| [1] | **BME** | **Burst Mode Enable**<br>0 = Disables the 8-data burst mode<br>1 = Enables the 8-data burst mode<br>If there are 8 words to be transferred, and the BME [1] =1, the GDMA_TCNTx should be 0x01. However, if BME [1] =0, the GDMA_TCNTx should be 0x08.<br>It has to set BME [1] = 0 for I/O device access. |
| [0] | **GDMAEN** | **GDMA Enable**<br>0 = Disables the GDMA operation<br>1 = Enables the GDMA operation; this bit will be clear automatically when the transfer is complete on AUTOIEN [19] register bit is on Disable mode.<br>Note:<br>When operate in Non-Descriptor mode, this bit determines the Memory-to Memory, Memory-to-I/O and I/O-to-Memory operation or not.<br>When operate in Descriptor mode, this bit is determined in descriptor list.<br>Note: Channel reset will clear this bit. |

**Descriptor fetches mode of Control Register:**

| Bits | Descriptions | |
|---|---|---|
| [22] | **SABNDERR** | **Source Address Boundary Alignment Error Flag**<br>If TWS [13:12]=10, GDMA_SRCB [1:0] should be 00<br>If TWS [13:12]=01, GDMA_SRCB [0] should be 0<br>Except the SADIR function enabled.<br>The address boundary alignment should be depended on TWS [13:12].<br>0 = the GDMA_SRCB is on the boundary alignment.<br>1 = the GDMA_SRCB not on the boundary alignment<br>The SABNDERR register bits just can be read only. |
| [21] | **DABNDERR** | **Destination Address Boundary Alignment Error Flag**<br>If TWS [13:12]=10, GDMA_DSTB [1:0] should be 00<br>If TWS [13:12]=01, GDMA_DSTB [0] should be 0<br>Except the DADIR function enabled.<br>The address boundary alignment should be depended on TWS [13:12].<br>0 = the GDMA_DSTB is on the boundary alignment.<br>1 = the GDMA_DSTB not on the boundary alignment<br>The DABNDERR register bits just can be read only. |
| [17] | **BLOCK** | **Bus Lock**<br>0 = Unlocks the bus during the period of transfer<br>1 = locks the bus during the period of transfer |
| [13:12] | **TWS** | **Transfer Width Select**<br>00 = One byte (8 bits) is transferred for every GDMA operation<br>01 = One half-word (16 bits) is transferred for every GDMA operation<br>10 = One word (32 bits) is transferred for every GDMA operation<br>11 = Reserved<br>The GDMA_SCRB and GDMA_DSTB should be alignment under the TWS selection |
| [10] | **D_INTS** | **Descriptor Fetch Mode Interrupt Select**<br>0 = The interrupt will take place at every end of descriptor fetch transfer.<br>1 = The interrupt only take place at the last descriptor fetch transfer.<br>NOTE: this bit is only available in descriptor mode and lists intention. |

| Bits | Descriptions | |
|---|---|---|
| [7] | **SAFIX** | **Source Address Fixed**<br>0 = Source address is changed during the GDMA operation<br>1 = Do not change the source address during the GDMA operation. This feature can be used when data were transferred from a single source to multiple destinations. |
| [6] | **DAFIX** | **Destination Address Fixed**<br>0 = Destination address is changed during the GDMA operation<br>1 = Do not change the destination address during the GDMA operation. This feature can be used when data were transferred from multiple sources to a single destination. |
| [5] | **SADIR** | **Source Address Direction**<br>0 = Source address is incremented successively<br>1 = Source address is decremented successively |
| [4] | **DADIR** | **Destination Address Direction**<br>0 = Destination address is incremented successively<br>1 = Destination address is decremented successively |
| [3:2] | **GDMAMS** | **GDMA Mode Select**<br>00 = Software mode<br>01 = Reserved<br>10 = Reserved<br>11 = Reserved |
| [1] | **BME** | **Burst Mode Enable**<br>0 = Disables the 8-data burst mode<br>1 = Enables the 8-data burst mode<br>If there are 8 words to be transferred, and the BME [1] =1, the GDMA_TCNTx should be 0x01. However, if BME [1] =0, the GDMA_TCNTx should be 0x08.<br>It has to set BME [1] = 0 for I/O device access. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **GDMAEN** | **GDMA Enable**<br>0 = Disables the GDMA operation<br>1 = Enables the GDMA operation; this bit will be clear automatically when the transfer is complete on AUTOIEN [19] register bit is on Disable mode.<br>When operate in Non-Descriptor mode, this bit determines the Memory-to-Memory, Memory-to-I/O and I/O-to-Memory operation or not.<br>When operate in Descriptor mode, this bit determines the I/O-to-Memory operation or not.<br>Channel reset will clear this bit. |

## Channel 0 / 1 Source Base Address Register (GDMA_SRCB0, GDMA_SRCB1)

| Register | Address | R / W | Description | Reset Value |
|---|---|---|---|---|
| GDMA_SRCB0 | 0xB000_4004 | R/W | Channel 0 Source Base Address Register | 0x0000_0000 |
| GDMA_SRCB1 | 0xB000_4024 | R/W | Channel 1 Source Base Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| SRC_BASE_ADDR [31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SRC_BASE_ADDR [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SRC_BASE_ADDR [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SRC_BASE_ADDR [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **SRC_BASE_ADDR** | **32-bit Source Base Address**<br>The GDMA channel starts reading its data from the source address as defined in this source base address register. |

## Channel 0 / 1 Destination Base Address Register (GDMA_DSTB0, GDMA_DSTB1)

| Register | Address | R / W | Description | Reset Value |
|---|---|---|---|---|
| GDMA_DSTB0 | 0xB000_4008 | R/W | Channel 0 Destination Base Address Register | 0x0000_0000 |
| GDMA_DSTB1 | 0xB000_4028 | R/W | Channel 1 Destination Base Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DST_BASE_ADDR [31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DST_BASE_ADDR [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DST_BASE_ADDR [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DST_BASE_ADDR [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **DST_BASE_ADDR** | **32-bit Destination Base Address**<br>The GDMA channel starts writing its data to the destination address as defined in this destination base address register. During a block transfer, the GDMA determines successive destination addresses by adding to or subtracting from the destination base address. |

## Channel 0 / 1 Transfer Count Register (GDMA_TCNT0, GDMA_TCNT1)

| Register | Address | R / W | Description | Reset Value |
|---|---|---|---|---|
| GDMA_TCNT0 | 0xB000_400C | R/W | Channel 0 Transfer Count Register | 0x0000_0000 |
| GDMA_TCNT1 | 0xB000_402C | R/W | Channel 1 Transfer Count Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TFR_CNT [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TFR_CNT [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TFR_CNT [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | **TFR_CNT** | **Transfer Count**<br>**Non-Descriptor Mode:24-bit TFR_CNT [23:0]**<br>The TFR_CNT represents the required number of GDMA transfers. The maximum transfer count is 16M –1.<br><br>**Descriptor Mode: 14-bit TFR_CNT [13:0]**<br>The TFR_CNT represents the required number of GDMA transfers. The maximum transfer count is 16K –1. |

## Channel 0 / 1 Current Source Register (GDMA_CSRC0, GDMA_CSRC1)

| Register | Address | R / W | Description | Reset Value |
|---|---|---|---|---|
| GDMA_CSRC0 | 0xB000_4010 | R | Channel 0 Current Source Address Register | 0x0000_0000 |
| GDMA_CSRC1 | 0xB000_4030 | R | Channel 1 Current Source Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CURRENT_SRC_ADDR [31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CURRENT_SRC_ADDR [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CURRENT_SRC_ADDR [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CURRENT_SRC_ADDR [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **CURRENT_SRC_ADDR** | **32-bit Current Source Address**<br>The CURRENT_SRC_ADDR indicates the source address where the GDMA transfer is just occurring. During a block transfer, the GDMA determines the successive source addresses by adding to or subtracting from the source base address. Depending on the settings you make to the control register, the current source address will remain the same or will be incremented or decremented. |

### Channel 0/1 Current Destination Register (GDMA_CDST0, GDMA_CDST1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GDMA_CDST0 | 0xB000_4014 | R | Channel 0 Current Destination Address Register | 0x0000_0000 |
| GDMA_CDST1 | 0xB000_4034 | R | Channel 1 Current Destination Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CURRENT_DST_ADDR [31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CURRENT_DST_ADDR [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CURRENT_DST_ADDR [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CURRENT_DST_ADDR [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **CURRENT_DST_ADDR** | **32-bit Current Destination Address**<br>The CURRENT_DST_ADDR indicates the destination address where the GDMA transfer is just occurring. During a block transfer, the GDMA determines the successive destination addresses by adding to or subtracting from the destination base address. Depending on the settings you make to the control register, the current destination address will remain the same or will be incremented or decremented. |

## Channel 0 / 1 Current Transfer Count Register (GDMA_CTCNT0, GDMA_CTCNT1)

| Register | Address | R / W | Description | Reset Value |
|---|---|---|---|---|
| GDMA_CTCNT0 | 0xB000_4018 | R | Channel 0 Current Transfer Count Register | 0x0000_0000 |
| GDMA_CTCNT1 | 0xB000_4038 | R | Channel 1 Current Transfer Count Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CURENT_TFR_CNT [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CURRENT_TFR_CNT [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CURRENT_TFR_CNT [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | **CURRENT_TFR_CNT** | **Current Transfer Count**<br>The Current transfer count register indicates the number of transfer being performed.<br>**Non-Descriptor Mode: 24-bit CURENT_TFR_CNT [23:0]**<br>**Descriptor Mode : 14-bit CURENT_TFR_CNT [13:0]** |

## Channel 0/1 Descriptor Register (GDMA_DADR0/1)

| | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GDMA_DADR0** | 0xB000_401C | R/W | Channel 0 Control Register | 0x0000_0004 |
| **GDMA_DADR1** | 0xB000_403C | R/W | Channel 1 Control Register | 0x0000_0004 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Descriptor Address[31:24]** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Descriptor Address[23:16]** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Descriptor Address[15:8]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Descriptor Address[7:4]** | | | | **RUN** | **NON_DSPTRMODE** | **ORDEN** | **RESET** |

| Bits | Descriptions | |
|---|---|---|
| [31:4] | **Descriptor Address** | **Descriptor Address**<br>Contains address of next descriptor. |
| [3] | **RUN** | **Run**<br>The RUN bit can be cleared during descriptor data transfer, and set RUN bit to starts the stopped channel under [Descriptor Address] and [Non-DSPTRMODE] bits are set properly. When RUN bit is cleared and the NON_DSPTRMODE bit is set that non-descriptor fetch occurs whether a valid descriptor address is written to register GDMA_DADRx or not. This bit will reset automatically when each descriptor transfer stopped or the bit in descriptor list is zero. The Descriptor interrupt is determined by bit-10 of the GDMA_CTLx Register.<br>0 = Stops the channel.<br>1 = Starts the channel.<br>Note: must co-operate to [NON_DSPTRMODE] to start the channel with Descriptor fetch function. |

| Bits | Descriptions | |
|---|---|---|
| [2] | **NON_DSPTRMODE** | **Non-Descriptor-Fetch**<br>When NON_DSPTRMODE is set, the channel is considered as a channel with no descriptors. In this mode, the GDMA does not initiate descriptor fetching and software can program the SCRBx, DSTBx, CTRx and TCNTx registers to transfer data until the TCNTx reaches zero. The GDMA_DADRx register is not used in non-descriptor mode. If NON_DSPTRMDOE is cleared under [RUN] and [Descriptor Address] are set properly, GDMA controller initiates descriptor-fetching. The descriptor fetch transfer stops when the counter for the current transfer reaches zero, [RUN] bit is cleared and [NON_DSPTRMODE] is set base on the bits of the descriptor list.<br>0 = Descriptor-fetch transfer<br>1 = NON-descriptor-fetch transfer<br>**Note: this bit = 1 will disable Descriptor function regardless of the RUN bit is 1 or not.** |
| [1] | **ORDEN** | **Enable Ordering Execution for Descriptor List**<br>The GDMA_DADRx [ORDEN] determine which the next descriptor address will be fetched. If [ORDEN] is set, the GDMA controller fetches the next descriptor from Current GDMA_DADRx [Descriptor Address] + 16 bytes.<br>If this bit is cleared, GDMA fetches the next descriptor address from the current GDMA_DADRx [Descriptor Address] register.<br>GDMA_DADRx [ORDEN] is relevant only for descriptor-fetch function (GDMA_DADRx [NON_DSPTRMODE] = 0).<br>0 = Disable descriptor ordering. Fetch the next descriptor from register GDMA_DDADRx [Descriptor Address].<br>1 = Enable descriptor ordering. |
| [0] | **RESET** | **Reset Channel**<br>0 = Disable channel reset.<br>1 = Enable channel status reset and disable descriptor based function. |

## Channel 0/1 GDMA Internal Buffer Register (GDMA_INTBUF0/1)

Software can set the [17-16] bit of GDMA_INTCS to select channels and watch the value which has read from memory.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GDMA_INTBUF0 | 0xB000_4080 | R | GDMA Internal Buffer Word 0 | 0x0000_0000 |
| GDMA_INTBUF1 | 0xB000_4084 | R | GDMA Internal Buffer Word 1 | 0x0000_0000 |
| GDMA_INTBUF2 | 0xB000_4088 | R | GDMA Internal Buffer Word 2 | 0x0000_0000 |
| GDMA_INTBUF3 | 0xB000_408C | R | GDMA Internal Buffer Word 3 | 0x0000_0000 |
| GDMA_INTBUF4 | 0xB000_4090 | R | GDMA Internal Buffer Word 4 | 0x0000_0000 |
| GDMA_INTBUF5 | 0xB000_4094 | R | GDMA Internal Buffer Word 5 | 0x0000_0000 |
| GDMA_INTBUF6 | 0xB000_4098 | R | GDMA Internal Buffer Word 6 | 0x0000_0000 |
| GDMA_INTBUF7 | 0xB000_409C | R | GDMA Internal Buffer Word 7 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DATA_BUFFER [31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DATA_BUFFER [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DATA_BUFFER [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA_BUFFER [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | DATA_BUFFER | **Internal Buffer Register**<br>Each channel has its own internal buffer from Word 0 to Word 7. The [17-16] bit of GDMA_INTCS will determine the values of channels mapping to GDMA_INTBUF0~7.<br>**NOTE: The GDMA_INTBUF0~7 are available when burst mode used, otherwise, only the GDMA_INTBUF0 available.** |

## Channel 0/1 GDMA Interrupt Control and Status Register (GDMA_INTCS)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GDMA_INTCS | 0xB000_40A0 | R/W | Interrupt Control and Status Register (2 Chs) | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | BUF_RD_SEL | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | TERR1F | TC1F | TERR0F | TC0F |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | TERR1EN | TC1EN | TERR0EN | TC0EN |

| Bits | Descriptions | |
|---|---|---|
| [17:16] | **BUF_RD_SEL** | **Internal Buffer Read Select**<br>00 = Read Internal Buffer for Channel 0<br>01 = Read Internal Buffer for Channel 1<br>10 = RESERVED<br>11 = RESERVED |
| [11] | **TERR1F** | **Channel 1 Transfer Error**<br>O = No error occurs<br>1 = Hardware sets this bit on a GDMA transfer failure<br>This bit will be cleared when write logic 1.<br>Transfer error will generate GDMA interrupt |
| [10] | **TC1F** | **Channel 1 Terminal Count**<br>0 = Channel does not expire<br>1 = Channel expires; this bit is set only by GDMA hardware, and clear by software to write logic 1.<br>TC1 is the GDMA interrupt flag. TC1 or GDMATERR1 will generate interrupt |

| Bits | Descriptions | |
|---|---|---|
| [9] | **TERR0F** | **Channel 0 Transfer Error**<br>O = No error occurs<br>1 = Hardware sets this bit on a GDMA transfer failure<br>This bit will be cleared when write logic 1.<br>Transfer error will generate GDMA interrupt |
| [8] | **TC0F** | **Channel 0 Terminal Count**<br>0 = Channel does not expire<br>1 = Channel expires; this bit is set only by GDMA hardware, and clear by software to write logic 1.<br>TC0 is the GDMA interrupt flag. TC0 or GDMATERR0 will generate interrupt |
| [3] | **TEER1EN** | **Channel 1 Interrupt Enable for Transfer Error**<br>0 = Disable Interrupt<br>1 = Enable Interrupt |
| [2] | **TC1EN** | **Channel 1 Interrupt Enable for Terminal Count**<br>0 = Disable Interrupt<br>1 = Enable Interrupt |
| [1] | **TEER0EN** | **Channel 0 Interrupt Enable for Transfer Error**<br>0 = Disable Interrupt<br>1 = Enable Interrupt |
| [0] | **TC0EN** | **Channel 0 Interrupt Enable for Terminal Count**<br>0 = Disable Interrupt<br>1 = Enable Interrupt |

## 7.7 USB Host Controller (USBH)

The **Universal Serial Bus (USB)** is a fast, bi-directional, isochronous, low-cost, dynamically attachable serial interface standard intended for USB devices. The USB is a 4-wire serial cable bus that supports serial data exchange between a Host Controller and a network of peripheral devices. The attached peripherals share USB bandwidth through a host-scheduled, token-based protocol. Peripherals may be attached, configured, used, and detached, while the host and other peripherals continue operation (i.e. hot plug and unplug is supported).

A major design goal of the USB standard was to allow flexible, plug-and-play networks of USB devices. In any USB network, there will be only one host, but there can be many devices and hubs.

The USB Host Controller includes the following features

- Fully compliant with USB Revision 2.0 specification.
- Enhanced Host Controller Interface (EHCI) Revision 1.0 compatible.
- Open Host Controller Interface (OHCI) Revision 1.0 compatible.
- Supports high-speed (480Mbps), full-speed (12Mbps) and low-speed (1.5Mbps) USB devices.
- Supports Control, Bulk, Interrupt, Isochronous and Split transfers.
- Integrated a port routing logic to route full/low speed device to OHCI controller.
- Built-in DMA for real-time data transfer.

### 7.7.1 Register Mapping

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **Capability Registers (USBH_BA = 0xB000_5000)** | | | | |
| EHCVNR | 0xB000_5000 | R | EHCI Version Number Register | 0x0095_0020 |
| EHCSPR | 0xB000_5004 | R | EHCI Structural Parameters Register | 0x0000_0012 |
| EHCCPR | 0xB000_5008 | R | EHCI Capability Parameters Register | 0x0000_0000 |
| **Operational Registers** | | | | |
| UCMDR | 0xB000_5020 | R/W | USB Command Register | 0x0008_0000 |
| USTSR | 0xB000_5024 | R/W | USB Status Register | 0x0000_1004 |
| UIENR | 0xB000_5028 | R/W | USB Interrupt Enable Register | 0x0000_0000 |
| UFINDR | 0xB000_502C | R/W | USB Frame Index Register | 0x0000_0000 |
| UPFLBAR | 0xB000_5034 | R/W | USB Periodic Frame List Base Address Register | 0x0000_0000 |
| UCALAR | 0xB000_5038 | R/W | USB Current Asynchronous List Address Register | 0x0000_0000 |
| UASSTR | 0xB000_503C | R/W | USB Asynchronous Schedule Sleep Timer Register | 0x0000_0BD6 |
| UCFGR | 0xB000_5060 | R/W | USB Configure Flag Register | 0x0000_0000 |
| UPSCR0 | 0xB000_5064 | R/W | USB Port 0 Status and Control Register | 0x0000_2000 |
| UPSCR1 | 0xB000_5068 | R/W | USB Port 1 Status and Control Register | 0x0000_2000 |
| **Miscellaneous Registers** | | | | |
| USBPCR0 | 0xB000_50C4 | R/W | USB PHY 0 Control Register | 0x0000_0060 |

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| USBPCR1 | 0xB000_50C8 | R/W | USB PHY 1 Control Register | 0x0000_0020 |
| **OHCI Registers (USBO_BA = 0xB000_7000)** | | | | |
| HcRev | 0xB000_7000 | R | Host Controller Revision Register | 0x0000_0010 |
| HcControl | 0xB000_7004 | R/W | Host Controller Control Register | 0x0000_0000 |
| HcComSts | 0xB000_7008 | R/W | Host Controller Command Status Register | 0x0000_0000 |
| HcIntSts | 0xB000_700C | R/W | Host Controller Interrupt Status Register | 0x0000_0000 |
| HcIntEn | 0xB000_7010 | R/W | Host Controller Interrupt Enable Register | 0x0000_0000 |
| HcIntDis | 0xB000_7014 | R/W | Host Controller Interrupt Disable Register | 0x0000_0000 |
| HcHCCA | 0xB000_7018 | R/W | Host Controller Communication Area Register | 0x0000_0000 |
| HcPerCED | 0xB000_701C | R/W | Host Controller Period Current ED Register | 0x0000_0000 |
| HcCtrHED | 0xB000_7020 | R/W | Host Controller Control Head ED Register | 0x0000_0000 |
| HcCtrCED | 0xB000_7024 | R/W | Host Controller Control Current ED Register | 0x0000_0000 |
| HcBlkHED | 0xB000_7028 | R/W | Host Controller Bulk Head ED Register | 0x0000_0000 |
| HcBlkCED | 0xB000_702C | R/W | Host Controller Bulk Current ED Register | 0x0000_0000 |
| HcDoneH | 0xB000_7030 | R/W | Host Controller Done Head Register | 0x0000_0000 |
| HcFmIntv | 0xB000_7034 | R/W | Host Controller Frame Interval Register | 0x0000_2EDF |
| HcFmRem | 0xB000_7038 | R | Host Controller Frame Remaining Register | 0x0000_0000 |
| HcFNum | 0xB000_703C | R | Host Controller Frame Number Register | 0x0000_0000 |
| HcPerSt | 0xB000_7040 | R/W | Host Controller Periodic Start Register | 0x0000_0000 |
| HcLSTH | 0xB000_7044 | R/W | Host Controller Low Speed Threshold Register | 0x0000_0628 |
| HcRhDeA | 0xB000_7048 | R/W | Host Controller Root Hub Descriptor A Register | 0x0100_0002 |
| HcRhDeB | 0xB000_704C | R/W | Host Controller Root Hub Descriptor B Register | 0x0000_0000 |
| HcRhSts | 0xB000_7050 | R/W | Host Controller Root Hub Status Register | 0x0000_0000 |
| HcRhPrt1 | 0xB000_7054 | R/W | Host Controller Root Hub Port Status [1] | 0x0000_0000 |
| HcRhPrt2 | 0xB000_7058 | R/W | Host Controller Root Hub Port Status [2] | 0x0000_0000 |
| **OHCI USB Configuration Register** | | | | |
| OpModEn | 0xB000_7204 | R/W | USB Operational Mode Enable Register | 0X0000_0000 |

### 7.7.2 Register Details

## EHCI Version Number Register (EHCVNR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| EHCVNR | 0xB000_5000 | R | EHCI Version Number Register | 0x0095_0020 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Version | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Version | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CR_Length | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:16] | **Version** | **Host Controller Interface Version Number**<br>This is a two-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. |
| [7:0] | **CR_Length** | **Capability Registers Length**<br>This register is used as an offset to add to register base to find the beginning of the Operational Register Space. |

## EHCI Structural Parameters Register (EHCSPR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| EHCSPR | 0xB000_5004 | R | EHCI Structural Parameters Register | 0x0000_0012 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| N_CC | | | | N_PCC | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | PPC | N_PORTS | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:12] | **N_CC** | **Number of Companion Controller**<br>This field indicates the number of companion controllers associated with this USB 2.0 host controller.<br>A zero in this field indicates there are no companion host controllers. Port-ownership hand-off is not supported. Only high-speed devices are supported on the host controller root ports.<br>A value larger than zero in this field indicates there are companion USB 1.1 host controller(s). Port-ownership hand-offs are supported. High, Full- and Low-speed devices are supported on the host controller root ports. |
| [11:8] | **N_PCC** | **Number of Ports per Companion Controller**<br>This field indicates the number of ports supported per companion host controller. It is used to indicate the port routing configuration to system software.<br>For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2.<br>The number in this field must be consistent with N_PORTS and N_CC. |

| Bits | Descriptions | |
|---|---|---|
| [4] | **PPC** | **Port Power Control**<br>This field indicates whether the host controller implementation includes port power control. A one in this bit indicates the ports have port power switches. A zero in this bit indicates the port do not have port power stitches. The value of this field affects the functionality of the *Port Power* field in each port status and control register. |
| [3:0] | **N_PORTS** | **Number of Physical Downstream Ports**<br>This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register Space. Valid values are in the range of 1H to FH. A zero in this field is undefined. |

## EHCI Capability Parameters Register (EHCCPR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| EHCCPR | 0xB000_5008 | R | EHCI Capability Parameters Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| EECP | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ISO_SCH_TH | | | | Reserved | ASPC | PFList | 64B |

| Bits | Descriptions | |
|---|---|---|
| [15:8] | **EECP** | **EHCI Extended Capabilities Pointer (EECP)**<br>8'h0: No extended capabilities are implemented. |
| [7:4] | **ISO_SCH_TH** | **Isochronous Scheduling Threshold** |
| [2] | **ASPC** | **Asynchronous Schedule Park Capability**<br>1'b0: This EHCI host controller doesn't support park feature of high-speed queue heads in the Asynchronous Schedule. |
| [1] | **PFList** | **Programmable Frame List Flag**<br>1'b0: System software must use a frame list length of 1024 elements with this EHCI host controller. |
| [0] | **64B** | **64-bit Addressing Capability**<br>1'b0: Data structure using 32-bit address memory pointers. |

## USB Command Register (UCMDR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| UCMDR | 0xB000_5020 | R/W | USB Command Register | 0x0008_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| INT_TH_CTL | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | AsynADB | ASEN | PSEN | FLSize | | HCRESET | RunStop |

| Bits | Descriptions | |
|---|---|---|
| [23:16] | **INT_TH_CTL** | **Interrupt Threshold Control (R/W)**<br>This field is used by system software to select the maximum rate at which the host controller will issue interrupts. The only valid values are defined below. If software writes an invalid value to this register, the results are undefined. **Value Maximum Interrupt Interval**<br>00h Reserved<br>01h 1 micro-frame<br>02h 2 micro-frames<br>04h 4 micro-frames<br>08h 8 micro-frames (default, equates to 1 ms)<br>10h 16 micro-frames (2 ms)<br>20h 32 micro-frames (4 ms)<br>40h 64 micro-frames (8 ms)<br>Any other value in this register yields undefined results.<br>Software modifications to this bit while HCHalted bit is equal to zero results in undefined behavior. |

| Bits | Descriptions | |
|---|---|---|
| [6] | **AsynADB** | **Interrupt on Async Advance Doorbell (R/W)**<br>This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell.<br>When the host controller has evicted all appropriate cached schedule state, it sets the *Interrupt on Async Advance* status bit in the USBSTS register. If the *Interrupt on Async Advance Enable* bit in the USBINTR register is a one then the host controller will assert an interrupt at the next interrupt threshold.<br>The host controller sets this bit to a zero after it has set the *Interrupt on Async Advance* status bit in the USBSTS register to a one.<br>Software should not write a one to this bit when the asynchronous schedule is disabled. Doing so will yield undefined results. |
| [5] | **ASEN** | **Asynchronous Schedule Enable (R/W)**<br>This bit controls whether the host controller skips processing the Asynchronous Schedule. Values mean:<br>0b Do not process the Asynchronous Schedule<br>1b Use the ASYNCLISTADDR register to access the Asynchro-nous Schedule |
| [4] | **PSEN** | **Periodic Schedule Enable (R/W)**<br>This bit controls whether the host controller skips processing the Periodic Schedule. Values mean:<br>0b Do not process the Periodic Schedule<br>1b Use the PERIODICLISTBASE register to access the Periodic Schedule |
| [3:2] | **FLSize** | **Frame List Size (R/W or RO)**<br>This field is R/W only if *Programmable Frame List Flag* in the HCCPARAMS registers is set to a one. This field specifies the size of the frame list. The size the frame list controls which bits in the Frame Index Register should be used for the Frame List Current index. Values mean:<br>00b 1024 elements (4096 bytes) Default value<br>01b 512 elements (2048 bytes)<br>10b 256 elements (1024 bytes) – for resource-constrained environment<br>11b **Reserved** |

| Bits | Descriptions | |
|---|---|---|
| [1] | **HCRESET** | **Host Controller Reset (HCRESET) (R/W)**<br>This control bit is used by software to reset the host controller. The effects of this on Root Hub registers are similar to a Chip Hardware Reset.<br>When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines, etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports.<br>All operational registers, including port registers and port state machines are set to their initial values. Port ownership reverts to the companion host controller(s), with the side effects. Software must reinitialize the host controller in order to return the host controller to an operational state.<br>This bit is set to zero by the Host Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register.<br>Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior. |
| [0] | **RunStop** | **Run/Stop (R/W)**<br>1=Run. 0=Stop. When set to a 1, the Host Controller proceeds with execution of the schedule. The Host Controller continues execution as long as this bit is set to a 1. When this bit is set to 0, the Host Controller completes the current and any actively pipelined transactions on the USB and then halts. The Host Controller must halt within 16 micro-frames after software clears the Run bit. The HC Halted bit in the status register indicates when the Host Controller has finished its pending pipelined transactions and has entered the stopped state. Software must not write a one to this field unless the host controller is in the Halted state (i.e. HCHalted in the USBSTS register is a one). Doing so will yield undefined results. |

## USB Status Register (USTSR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| USTSR | 0xB000_5024 | R/W | USB Status Register | 0x0000_1000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ASSTS | PSSTS | RECLA | HCHalted | Reserved | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | IntAsynA | HSERR | FLROVER | PortCHG | UERRINT | USBINT |

| Bits | Descriptions | |
|---|---|---|
| [15] | **ASSTS** | **Asynchronous Schedule Status (RO)**<br>The bit reports the current real status of the Asynchronous Schedule. If this bit is a zero then the status of them Asynchronous Schedule is disabled. If this bit is a one then the status of the Asynchronous Schedule is enabled. The Host Controller is not required to i*mmediately* disable or enable the Asynchronous Schedule when software transitions the *Asynchronous Schedule Enable* bit in the USBCMD register. When this bit and the *Asynchronous Schedule Enable* bit are the same value, the Asynchronous Schedule is either r enabled (1) or disabled (0). |
| [14] | **PSSTS** | **Periodic Schedule Status (RO)**<br>The bit reports the current real status of the Periodic Schedule. If this bit is a zero then the status of the Periodic Schedule is disabled. If this bit is a one then the status of the Periodic Schedule is enabled. The Host Controller is not required to *immediately* disable or enable the Periodic Schedule when software transitions the *Periodic Schedule Enable* bit in the USBCMD register. When this bit and the *Periodic Schedule Enable* bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). |
| [13] | **RECLA** | **Reclamation (RO)**<br>This is a read-only status bit, which is used to detect an empty asynchronous schedule. |

| Bits | Descriptions | |
|---|---|---|
| [12] | **HCHalted** | **HCHalted (RO)**<br>This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing as a result of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error). |
| [5] | **IntAsynA** | **Interrupt on Async Advance (R/WC)**<br>System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the *Interrupt on Async Advance Doorbell* bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. |
| [4] | **HSERR** | **Host System Error (R/WC)**<br>The Host Controller sets this bit to 1 when a serious error occurs during a host system access involving the Host Controller module. |
| [3] | **FLROVER** | **Frame List Rollover (R/WC)**<br>The Host Controller sets this bit to a one when the *Frame List Index* rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example, if the frame list size (as programmed in the *Frame List Size* field of the USBCMD register) is 1024, the *Frame Index Register* rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FRINDEX [12] toggles. |
| [2] | **PortCHG** | **Port Change Detect (R/WC)**<br>The Host Controller sets this bit to a one when any port for which the Port Owner bit is set to zero has a change bit transition from a zero to a one or a *Force Port Resume* bit transition from a zero to a one as a result of a J-K transition detected on a suspended port. This bit will also be set as a result of the *Connect Status Change* being set to a one after system software has relinquished ownership of a connected port by writing a one to a port's *Port Owner* bit.<br>This bit is allowed to be maintained in the Auxiliary power well. Alternatively, it is also acceptable that on a D3 to D0 transition of the EHCI HC device, this bit is loaded with the OR of all of the PORTSC change bits (including: Force port resume, over-current change, enable/disable change and connect status change). |
| [1] | **UERRINT** | **USB Error Interrupt (USBERRINT) (R/WC)**<br>The Host Controller sets this bit to 1 when completion of a USB transaction results in an error condition (e.g., error counter underflow). If the TD on which the error interrupt occurred also had its IOC bit set, both this bit and USBINT bit are set. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **USBINT** | **USB Interrupt (USBINT) (R/WC)**<br>The Host Controller sets this bit to 1 on the completion of a USB transaction, which results in the retirement of a Transfer Descriptor that had its IOC bit set.<br>The Host Controller also sets this bit to 1 when a short packet is detected (actual number of bytes received was less than the expected number of bytes). |

## USB Interrupt Enable Register (UIENR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| UIENR | 0xB000_5028 | R/W | USB Interrupt Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | AsynAEN | HSERREN | FLREN | PCHGEN | UERREN | USBIEN |

| Bits | Descriptions | |
|---|---|---|
| [5] | **AsynAEN** | **Interrupt on Async Advance Enable**<br>When this bit is a one, and the *Interrupt on Async Advance* bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the *Interrupt on Async Advance* bit. |
| [4] | **HSERREN** | **Host System Error Enable**<br>When this bit is a one, and the Host System Error Status bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Host System Error bit. |
| [3] | **FLREN** | **Frame List Rollover Enable**<br>When this bit is a one, and the Frame List Rollover bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Frame List Rollover bit. |
| [2] | **PCHGEN** | **Port Change Interrupt Enable**<br>When this bit is a one, and the Port Change Detect bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port Change Detect bit. |

| Bits | Descriptions | |
|---|---|---|
| [1] | **UERREN** | **USB Error Interrupt Enable**<br>When this bit is a one, and the USBERRINT bit in the USBSTS register is a one, the host t controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit. |
| [0] | **USBIEN** | **USB Interrupt Enable**<br>When this bit is a one, and the USBINT bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit. |

## USB Frame Index Register (UFINDR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| UFINDR | 0xB000_502C | R/W | USB Frame Index Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | FrameIND | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FrameIND | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [13:0] | **FrameIND** | **Frame Index**<br>The value in this register increment at the end of each time frame (e.g. micro-frame). Bits [N: 3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register. |

## USB Periodic Frame List Base Address Register (UPFLBAR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| UPFLBAR | 0xB000_5034 | R/W | USB Periodic Frame List Base Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| BADDR | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BADDR | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| BADDR | | | | Reserved | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:12] | **BADDR** | **Base Address (Low)**<br>These bits correspond to memory address signals [31:12], respectively. |

## USB Current Asynchronous List Address Register (UCALAR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| UCALAR | 0xB000_5038 | R/W | USB Current Asynchronous List Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| LPL | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LPL | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LPL | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LPL | | | Reserved | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:5] | **LPL** | **Link Pointer Low (LPL)**<br>These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH). |

## USB Asynchronous Schedule Sleep Timer Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| UASSTR | 0xB000_503C | R/W | USB Asynchronous Schedule Sleep Timer Register | 0x0000_0BD6 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | ASTMR | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ASTMR | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [11:0] | **ASSTMR** | **Asynchronous Schedule Sleep Timer**<br>This field defines the AsyncSchedSleepTime of EHCI spec.<br>The asynchronous schedule sleep timer is used to control how often the host controller fetches asynchronous schedule list from system memory while the asynchronous schedule is empty.<br>The default value of this timer is 12'hBD6. Because this timer is implemented in UTMI clock (30MHz) domain, the default sleeping time will be about 100us. |

## USB Configure Flag Register (UCFGR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| UCFGR | 0xB000_5060 | R/W | USB Configure Flag Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | CF |

| Bits | Descriptions | |
|---|---|---|
| [0] | **CF** | **Configure Flag (CF)**<br>Host software sets this bit as the last action in its process of configuring the Host Controller. This bit controls the default port-routing control logic. Bit values and side-effects are listed below.<br>0b Port routing control logic default-routes each port to an implementation dependent classic host controller.<br>1b Port routing control logic default-routes all ports to this host controller. |

## USB Port 0 Status and Control Register (UPSCR0)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| UPSCR0 | 0xB000_5064 | R/W | USB Port 0 Status and Control Register | 0x0000_2000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | PO | PP | LStatus | | Reserved | PRST |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Suspend | FPResum | OCCHG | OCACT | PENCHG | PEN | CSCHG | CSTS |

| Bits | Descriptions | |
|---|---|---|
| [13] | **PO** | **Port Owner (R/W)**<br>This bit unconditionally goes to a 0b when the Configured bit in the CONFIGFLAG register makes a 0b to 1b transition. This bit unconditionally goes to 1b whenever the Configured bit is zero.<br>System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that a companion host controller owns and controls the port. |
| [12] | **PP** | **Port Power (PP)**<br>Host controller has port power control switches. This bit represents the Current setting of the switch (0 = off, 1 = on). When power is not available on a port (i.e. *PP* equals a 0), the port is nonfunctional and will not report attaches, detaches, etc.<br>When an over-current condition is detected on a powered port and *PPC* is a one, the *PP* bit in each affected port may be transitioned by the host controller from a 1 to 0 (removing power from the port). |

| Bits | Descriptions | |
|---|---|---|
| [11:10] | **LStatus** | **Line Status (RO)**<br>These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. These bits are used for detection of low-speed USB devices prior to the port reset and enable sequence. This field is valid only when the port enable bit is zero and the current connect status bit is set to a one.<br>The encoding of the bits are:<br>**Bits[11:10] USB State Interpretation**<br>00b SE0 Not Low-speed device, perform EHCI reset<br>10b J-state Not Low-speed device, perform EHCI reset<br>01b K-state Low-speed device, release ownership of port<br>11b Undefined Not Low-speed device, perform EHCI reset.<br>This value of this field is undefined if Port Power is zero. |
| [8] | **PRST** | **Port Reset (R/W)**<br>1=Port is in Reset. 0=Port is not in Reset. Default = 0. When software writes a one to this bit (from a zero), the bus reset sequence as defined in the USB Specification Revision 2.0 is started. Software writes a zero to this bit to terminate the bus reset sequence. Software must keep this bit at a one long enough to ensure the reset sequence, as specified in the USB Specification Revision 2.0, completes. Note: when software writes this bit to a one, it must also write a zero to the Port Enable bit.<br>Note that when software writes a zero to this bit there may be a delay before the bit status changes to a zero. The bit status will not read as a zero until after the reset has completed. If the port is in high-speed mode after reset is complete, the host controller will automatically enable this port (e.g. set the Port Enable bit to a one). A host controller must terminate the reset and stabilize the state of the port within 2 milliseconds of software transitioning this bit from a one to a zero. For example: if the port detects that the attached device is high-speed during reset, then the host controller must have the port in the enabled state within 2ms of software writing this bit to a zero.<br>The HCHalted bit in the USBSTS register should be a zero before software attempts to use this bit. The host controller may hold Port Reset asserted to a one when the HCHalted bit is a one.<br>This field is zero if Port Power is zero. |

| Bits | Descriptions | |
|---|---|---|
| [7] | **Suspend** | **Suspend (R/W)**<br>1=Port in suspend state. 0=Port not in suspend state. Default = 0. Port Enabled Bit and Suspend bit of this register define the port states as follows:<br>**Bits [Port Enabled, Suspend] Port State**<br>0X Disable<br>10 Enable<br>11 Suspend<br>When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction, if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB.<br>A write of zero to this bit is ignored by the host controller. The host controller will unconditionally set this bit to a zero when:<br>Software sets the Force Port Resume bit to a zero (from a one).<br>Software sets the Port Reset bit to a one (from a zero).<br>If host software sets this bit to a one when the port is not enabled (i.e. Port enabled bit is a zero) the results are undefined.<br>This field is zero if Port Power is zero. |

| Bits | Descriptions | |
|---|---|---|
| [6] | **FPResum** | **Force Port Resume (R/W)**<br>1= Resume detected/driven on port. 0=No resume (Kstate) detected/driven on port. Default = 0. This functionality defined for manipulating this bit depends on the value of the Suspend bit. For example, if the port is not suspended (Suspend and Enabled bits are a one) and software transitions this bit to a one, then the effects on the bus are undefined.<br>Software sets this bit to a 1 to drive resume signaling. The Host Controller sets this bit to a 1 if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to a one. If software sets this bit to a one, the host controller must not set the Port Change Detect bit.<br>Note that when the EHCI controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. Software must appropriately time the Resume and set this bit to a zero when the appropriate amount of time has elapsed. Writing a zero (from one) causes the port to return to high-speed mode (forcing the bus below the port into a high-speed idle). This bit will remain a one until the port has switched to the high-speed idle. The host controller must complete this transition within 2 milliseconds of software setting this bit to a zero.<br>This field is zero if Port Power is zero. |
| [5] | **OCCHG** | **Over-current Change (R/WC)**<br>Default = 0. 1=This bit gets set to a one when there is a change to Over-current Active. Software clears this bit by writing a one to this bit position. |
| [4] | **OCACT** | **Over-current Active (RO)**<br>Default = 0. 1=This port currently has an over current condition. 0=This port does not have an over-current condition. This bit will automatically transition from a one to a zero when the over current condition is removed. |
| [3] | **PENCHG** | **Port Enable/Disable Change (R/WC)**<br>1=Port enabled/disabled status has changed. 0=No change. Default = 0. For the root hub, this bit gets set to a one only when a port is disabled due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification for the definition of a Port Error). Software clears this bit by writing a 1 to it.<br>This field is zero if Port Power is zero. |

| Bits | Descriptions | |
|---|---|---|
| [2] | **PEN** | **Port Enabled/Disabled (R/W)**<br>1=Enable. 0=Disable. Default = 0. Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. The host controller will only set this bit to a one when the reset sequence determines that the attached device is a high-speed device.<br>Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events.<br>When the port is disabled (0b) downstream propagation of data is blocked on this port, except for reset.<br>This field is zero if Port Power is zero. |
| [1] | **CSCHG** | **Connect Status Change (R/W)**<br>1=Change in Current Connect Status. 0=No change. Default = 0. Indicates a change has occurred in the port's Current Connect Status. The host controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be "setting" an already-set bit (i.e., the bit will remain set).Software sets this bit to 0 by writing a 1 to it.<br>This field is zero if Port Power is zero. |
| [0] | **CSTS** | **Current Connect Status (RO)**<br>1=Device is present on port. 0=No device is present. Default = 0. This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set.<br>This field is zero if Port Power is zero. |

## USB Port 1 Status and Control Register (UPSCR1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| UPSCR1 | 0xB000_5068 | R/W | USB Port 1 Status and Control Register | 0x0000_2000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | Reserved | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | PO | PP | LStatus | | Reserved | PRST |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Suspend | FPResum | OCCHG | OCACT | PENCHG | PEN | CSCHG | CSTS |

| Bits | Descriptions | |
|---|---|---|
| [13] | **PO** | **Port Owner (R/W)**<br>This bit unconditionally goes to a 0b when the Configured bit in the CONFIGFLAG register makes a 0b to 1b transition. This bit unconditionally goes to 1b whenever the Configured bit is zero.<br>System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that a companion host controller owns and controls the port. |
| [12] | **PP** | **Port Power (PP)**<br>Host controller has port power control switches. This bit represents the Current setting of the switch (0 = off, 1 = on). When power is not available on a port (i.e. *PP* equals a 0), the port is nonfunctional and will not report attaches, detaches, etc.<br>When an over-current condition is detected on a powered port and *PPC* is a one, the *PP* bit in each affected port may be transitioned by the host controller from a 1 to 0 (removing power from the port). |

| Bits | Descriptions | |
|---|---|---|
| [11:10] | **LStatus** | **Line Status (RO)**<br>These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. These bits are used for detection of low-speed USB devices prior to the port reset and enable sequence. This field is valid only when the port enable bit is zero and the current connect status bit is set to a one.<br>The encoding of the bits are:<br>**Bits[11:10] USB State Interpretation**<br>00b SE0 Not Low-speed device, perform EHCI reset<br>10b J-state Not Low-speed device, perform EHCI reset<br>01b K-state Low-speed device, release ownership of port<br>11b Undefined Not Low-speed device, perform EHCI reset.<br>This value of this field is undefined if Port Power is zero. |
| [8] | **PRST** | **Port Reset (R/W)**<br>1=Port is in Reset. 0=Port is not in Reset. Default = 0. When software writes a one to this bit (from a zero), the bus reset sequence as defined in the USB Specification Revision 2.0 is started. Software writes a zero to this bit to terminate the bus reset sequence. Software must keep this bit at a one long enough to ensure the reset sequence, as specified in the USB Specification Revision 2.0, completes. Note: when software writes this bit to a one, it must also write a zero to the Port Enable bit.<br>Note that when software writes a zero to this bit there may be a delay before the bit status changes to a zero. The bit status will not read as a zero until after the reset has completed. If the port is in high-speed mode after reset is complete, the host controller will automatically enable this port (e.g. set the Port Enable bit to a one). A host controller must terminate the reset and stabilize the state of the port within 2 milliseconds of software transitioning this bit from a one to a zero. For example: if the port detects that the attached device is high-speed during reset, then the host controller must have the port in the enabled state within 2ms of software writing this bit to a zero.<br>The HCHalted bit in the USBSTS register should be a zero before software attempts to use this bit. The host controller may hold Port Reset asserted to a one when the HCHalted bit is a one.<br>This field is zero if Port Power is zero. |

| Bits | Descriptions | |
|---|---|---|
| [7] | **Suspend** | **Suspend (R/W)**<br>1=Port in suspend state. 0=Port not in suspend state. Default = 0. Port Enabled Bit and Suspend bit of this register define the port states as follows:<br>**Bits [Port Enabled, Suspend] Port State**<br>0X Disable<br>10 Enable<br>11 Suspend<br>When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction, if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB.<br>A write of zero to this bit is ignored by the host controller. The host controller will unconditionally set this bit to a zero when:<br>Software sets the Force Port Resume bit to a zero (from a one).<br>Software sets the Port Reset bit to a one (from a zero).<br>If host software sets this bit to a one when the port is not enabled (i.e. Port enabled bit is a zero) the results are undefined.<br>This field is zero if Port Power is zero. |

| Bits | Descriptions | |
|---|---|---|
| [6] | **FPResum** | **Force Port Resume (R/W)**<br>1= Resume detected/driven on port. 0=No resume (Kstate) detected/driven on port. Default = 0. This functionality defined for manipulating this bit depends on the value of the Suspend bit. For example, if the port is not suspended (Suspend and Enabled bits are a one) and software transitions this bit to a one, then the effects on the bus are undefined.<br>Software sets this bit to a 1 to drive resume signaling. The Host Controller sets this bit to a 1 if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to a one. If software sets this bit to a one, the host controller must not set the Port Change Detect bit.<br>Note that when the EHCI controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. Software must appropriately time the Resume and set this bit to a zero when the appropriate amount of time has elapsed. Writing a zero (from one) causes the port to return to high-speed mode (forcing the bus below the port into a high-speed idle). This bit will remain a one until the port has switched to the high-speed idle. The host controller must complete this transition within 2 milliseconds of software setting this bit to a zero.<br>This field is zero if Port Power is zero. |
| [5] | **OCCHG** | **Over-current Change (R/WC)**<br>Default = 0. 1=This bit gets set to a one when there is a change to Over-current Active. Software clears this bit by writing a one to this bit position. |
| [4] | **OCACT** | **Over-current Active (RO)**<br>Default = 0. 1=This port currently has an over current condition. 0=This port does not have an over-current condition. This bit will automatically transition from a one to a zero when the over current condition is removed. |
| [3] | **PENCHG** | **Port Enable/Disable Change (R/WC)**<br>1=Port enabled/disabled status has changed. 0=No change. Default = 0. For the root hub, this bit gets set to a one only when a port is disabled due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification for the definition of a Port Error). Software clears this bit by writing a 1 to it.<br>This field is zero if Port Power is zero. |

| Bits | Descriptions | |
|---|---|---|
| [2] | **PEN** | **Port Enabled/Disabled (R/W)**<br>1=Enable. 0=Disable. Default = 0. Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. The host controller will only set this bit to a one when the reset sequence determines that the attached device is a high-speed device.<br>Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events.<br>When the port is disabled (0b) downstream propagation of data is blocked on this port, except for reset.<br>This field is zero if Port Power is zero. |
| [1] | **CSCHG** | **Connect Status Change (R/W)**<br>1=Change in Current Connect Status. 0=No change. Default = 0. Indicates a change has occurred in the port's Current Connect Status. The host controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be "setting" an already-set bit (i.e., the bit will remain set).Software sets this bit to 0 by writing a 1 to it.<br>This field is zero if Port Power is zero. |
| [0] | **CSTS** | **Current Connect Status (RO)**<br>1=Device is present on port. 0=No device is present. Default = 0. This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set.<br>This field is zero if Port Power is zero. |

## USB PHY 0 Control Register (USBPCR0)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| USBPCR0 | 0xB000_50C4 | R/W | USB PHY 0 Control Register | 0x0000_0060 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | ClkValid | Reserved | | Suspend |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLK48 | REFCLK | CLK_SEL | | XO_ON | SIDDQ | Reserved | |

| Bits | Descriptions | |
|---|---|---|
| [11] | **ClkValid** | **UTMI Clock Valid**<br>This bit is a flag to indicate if the UTMI clock from USB 2.0 PHY is ready. S/W program must prevent to write other control registers before this UTMI clock valid flag is active.<br>1'b0: UTMI clock is not valid<br>1'b1: UTMI clock is valid |
| [8] | **Suspend** | **Suspend Assertion**<br>This bit controls the suspend mode of USB PHY 0.<br>While PHY was suspended, all circuits of PHY were powered down and outputs are tri-stated.<br>This bit is 1'b0 in default. This means the USB PHY 0 is suspended in default. It is necessary to set this bit 1'b1 to make USB PHY 0 leave suspend mode before doing configuration of USB host.<br>1'b0: USB PHY 0 was suspended.<br>1'b1: USB PHY 0 was not suspended. |

| Bits | Descriptions | |
|---|---|---|
| [7] | **CLK48** | **Digital Logic Clock Select**<br>This bit controls the input signal clk48m_sel of USB PHY 0.<br>This signal selects Power-Save mode.<br>1'b0: Non-Power-Save mode. The PLL and the phase interpolator are powered up. The digital logic uses a 480MHz clock. Non-Power-Save mode is valid in all modes and speeds of operation.<br>1'b1: Power-Save mode. The PLL and the phase interpolator are powered down. The digital logic uses a 48MHz clock. Power-Save mode is valid for only FS-Only operation. The CLK_SEL should be set to 2'b10 (48MHz) when this bit is high. |
| [6] | **REFCLK** | **Reference Clock Source Select**<br>This bit has to set to 1. |
| [5:4] | **CKL_SEL** | **Reference Clock Frequency Select**<br>This field has to set to 2'b10; |
| [3] | **XO_ON** | **Force XO Block on During a Suspend**<br>This bit controls the input signal xo_on of USB PHY 0.<br>1'b0: If all ports are suspended, the XO block is powered up, and the test_clk48m signal is available.<br>1'b1: This bit is inactive, and the XO block is powered down when all ports are suspended. |
| [2] | **SIDDQ** | **IDDQ Test Enable**<br>This bit controls the input signal siddq of USB PHY 0.<br>This signal powers down all analog blocks.<br>1'b0: The analog blocks are in normal operation.<br>1'b1: The analog blocks are powered down. |

## USB PHY 1 Control Register (USBPCR1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| USBPCR1 | 0xB000_50C8 | R/W | USB PHY 1 Control Register | 0x0000_0020 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | XO_SEL | Reserved | | Suspend |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLK48 | REFCLK | CLK_SEL | | XO_ON | SIDDQ | Reserved | |

| Bits | Descriptions | |
|---|---|---|
| [11] | **XO_SEL** | **Clock Select for XO Block**<br>This bit should be set to 0 to define the XO block uses a 48MHz external clock supplied from PHY 0 |
| [8] | **Suspend** | **Suspend Assertion**<br>This bit controls the suspend mode of USB PHY 1.<br>While PHY was suspended, all circuits of PHY were powered down and outputs are tri-stated.<br>This bit is 1'b0 in default. This means the USB PHY 1 is suspended in default. It is necessary to set this bit 1'b1 to make USB PHY 1 leave suspend mode before doing configuration of USB host.<br>1'b0: USB PHY 1 was suspended.<br>1'b1: USB PHY 1 was not suspended. |

| Bits | Descriptions | |
|---|---|---|
| [7] | **CLK48** | **Digital Logic Clock Select**<br>This bit controls the input signal clk48m_sel of USB PHY 1.<br>This signal selects Power-Save mode.<br>1'b0: Non-Power-Save mode. The PLL and the phase interpolator are powered up. The digital logic uses a 480MHz clock. Non-Power-Save mode is valid in all modes and speeds of operation.<br>1'b1: Power-Save mode. The PLL and the phase interpolator are powered down. The digital logic uses a 48MHz clock. Power-Save mode is valid for only FS-Only operation. The CLK_SEL should be set to 2'b10 (48MHz) when this bit is high. |
| [6] | **REFCLK** | **Reference Clock Source Select**<br>This bit has to set to 0. |
| [5:4] | **CKL_SEL** | **Reference Clock Frequency Select**<br>This field has to set to 2'b10. |
| [3] | **XO_ON** | **Force XO Block on During a Suspend**<br>This bit controls the input signal xo_on of USB PHY 1.<br>1'b0: If all ports are suspended, the XO block is powered up, and the test_clk48m signal is available.<br>1'b1: This bit is inactive, and the XO block is powered down when all ports are suspended. |
| [2] | **SIDDQ** | **IDDQ Test Enable**<br>This bit controls the input signal siddq of USB PHY 1.<br>This signal powers down all analog blocks.<br>1'b0: The analog blocks are in normal operation.<br>1'b1: The analog blocks are powered down. |

## Host Controller Revision Register (HcRev)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcRev | 0xB000_7000 | R | Host Controller Revision Register | 0x0000_0010 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Rev | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [7:0] | **Rev** | **Revision**<br>Indicates the Open HCI Specification revision number implemented by the Hardware. Host Controller supports 1.0 specification.<br>(X.Y = XYh) |

## Host Controller Control Register (HcControl)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcControl | 0xB000_7004 | R/W | Host Controller Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | RWakeEn | RWake | IntRoute |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HcFunc | | BlkEn | CtrlEn | ISOEn | PeriEn | CtrlBlkRatio | |

| Bits | Descriptions | |
|---|---|---|
| [10] | **RWakeEn** | **Remote Wakeup Connected Enable**<br>If a remote wakeup signal is supported, this bit enables that operation. Since there is no remote wakeup signal supported, this bit is ignored. |
| [9] | **RWake** | **Remote Wakeup Connected**<br>This bit indicated whether the HC supports a remote wakeup signal. This implementation does not support any such signal. The bit is hard-coded to '0.' |
| [8] | **IntRoute** | **Interrupt Routing**<br>This bit is used for interrupt routing:<br>0: Interrupts routed to normal interrupt mechanism (INT).<br>1: Interrupts routed to SMI. |
| [7:6] | **HcFunc** | **Host Controller Functional State**<br>This field sets the Host Controller state. The Controller may force a state change from USBSUSPEND to USBRESUME after detecting resume signaling from a downstream port. States are:<br>00: USBRESET<br>01: USBRESUME<br>10: USBOPERATIONAL<br>11: USBSUSPEND |

| Bits | Descriptions | |
|---|---|---|
| [5] | **BlkEn** | **Bulk List Enable**<br>When set this bit enables processing of the Bulk list. |
| [4] | **CtrlEn** | **Control List Enable**<br>When set this bit enables processing of the Control list. |
| [3] | **ISOEn** | **Isochronous List Enable**<br>When clear, this bit disables the Isochronous List when the Periodic List is enabled (so Interrupt EDs may be serviced). While processing the Periodic List, the Host Controller will check this bit when it finds an isochronous ED. |
| [2] | **PeriEn** | **Periodic List Enable**<br>When set, this bit enables processing of the Periodic (interrupt and isochronous) list. The Host Controller checks this bit prior to attempting any periodic transfers in a frame. |
| [1:0] | **CtrlBlkRatio** | **Control Bulk Service Ratio**<br>Specifies the number of Control Endpoints serviced for every Bulk Endpoint. Encoding is N-1 where N is the number of Control Endpoints (i.e. '00' = 1 Control Endpoint; '11' = 3 Control Endpoints) |

## Host Controller Command Status Register (HcComSts)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcComSts | 0xB000_7008 | R/W | Host Controller Command Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | SchOverRun | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | OCReq | BlkFill | CtrlFill | HCReset |

| Bits | Descriptions | |
|---|---|---|
| [17:16] | **SchOverRun** | **Schedule Overrun Count**<br>This field is increment every time the **SchedulingOverrun** bit in *HcInterruptStatus* is set. The count wraps from '11' to '00.' |
| [3] | **OCReq** | **Ownership Chang Request**<br>When set by software, this bit sets the **OwnershipChange** field in *HcInterruptStatus*. The bit is cleared by software. |
| [2] | **BlkFill** | **Bulk List Filled**<br>Set to indicate there is an active ED on the Bulk List. The bit may be set by either software or the Host Controller and cleared by the Host Controller each time it begins processing the head of the Bulk List. |
| [1] | **CtrlFill** | **Control List Filled**<br>Set to indicate there is an active ED on the Control List. It may be set by either software or the Host Controller and cleared by the Host Controller each time it begins processing the head of the Control List. |
| [0] | **HCReset** | **Host Controller Reset**<br>This bit is set to initiate the software reset. This bit is cleared by the Host Controller, upon completed of the reset operation. |

## Host Controller Interrupt Status Register (HcIntSts)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcIntSts | 0xB000_700C | R/W | Host Controller Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | OC | Reserved | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | RHSC | FNOF | UnRecErr | Resume | SOF | WBDnHD | SchOR |

| Bits | Descriptions | |
|---|---|---|
| [30] | **OC** | **Ownership Change**<br>This bit is set when the **OwnershipChangeRequest** bit of *HcCommandStatus* is set. |
| [6] | **RHSC** | **Root Hub Status Change**<br>This bit is set when the content of *HcRhStatus* or the content of any *HcRhPortStatus* register has changed. |
| [5] | **FNOF** | **Frame Number Overflow**<br>Set when bit 15 of **FrameNumber** changes value. |
| [4] | **UnRecErr** | **Unrecoverable Error**<br>This event is not implemented and is hard-coded to '0.' Writes are ignored. |
| [3] | **Resume** | **Resume Detected**<br>Set when Host Controller detects resume signaling on a downstream port. |
| [2] | **SOF** | **Start Of Frame**<br>Set when the Frame Management block signals a 'Start of Frame' event. |
| [1] | **WBDnHD** | **Write Back Done Head**<br>Set after the Host Controller has written *HcDoneHead* to *HccaDoneHead*. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **SchOR** | **Scheduling Overrun**<br>Set when the List Processor determines a Schedule Overrun has occurred. |

## Host Controller Interrupt Enable Register (HcIntEn)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcIntEn | 0xB000_7010 | R/W | Host Controller Interrupt Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IntEn | OCEn | Reserved | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | RHSCEn | FNOFEn | URErrEn | ResuEn | SOFEn | WBDHEn | SchOREn |

| Bits | Descriptions | |
|---|---|---|
| [31] | **IntEn** | **Master Interrupt Enable**<br>This bit is a global interrupt enable. A write of '1' allows interrupts to be enabled via the specific enable bits listed above. |
| [30] | **OCEn** | **Ownership Change Enable**<br>0: Ignore<br>1: Enables interrupt generation due to Ownership Change. |
| [6] | **RHSCEn** | **Root Hub Status Change Enable**<br>0: Ignore<br>1: Enables interrupt generation due to Root Hub Status Change. |
| [5] | **FNOFEn** | **Frame Number Overflow Enable**<br>0: Ignore<br>1: Enables interrupt generation due to Frame Number Overflow. |
| [4] | **URErrEn** | **Unrecoverable Error Enable**<br>This event is not implemented. All writes to this bit are ignored. |
| [3] | **ResuEn** | **Resume Detected Enable**<br>0: Ignore<br>1: Enables interrupt generation due to Resume Detected. |

| Bits | Descriptions | |
|---|---|---|
| [2] | **SOFEn** | **Start Of Frame Enable**<br>0: Ignore<br>1: Enables interrupt generation due to Start of Frame. |
| [1] | **WBDHEn** | **Write Back Done Head Enable**<br>0: Ignore<br>1: Enables interrupt generation due to Write-back Done Head. |
| [0] | **SchOREn** | **Scheduling Overrun Enable**<br>0: Ignore<br>1: Enables interrupt generation due to Scheduling Overrun. |

## Host Controller Interrupt Disable Register (HcIntDis)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcIntDis | 0xB000_7014 | R/W | Host Controller Interrupt Disable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IntDis | OCDis | Reserved | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | RHSCDis | FNOFDis | URErrDis | ResuDis | SOFDis | WBDHDis | SchORDis |

| Bits | Descriptions | |
|---|---|---|
| [31] | **IntDis** | **Master Interrupt Disable**<br>Global interrupt disable. A write of '1' disables all interrupts. |
| [30] | **OCDis** | **Ownership Change Disable**<br>0: Ignore<br>1: Disables interrupt generation due to Ownership Change. |
| [6] | **RHSCDis** | **Root Hub Status Change Disable**<br>0: Ignore<br>1: Disables interrupt generation due to Root Hub Status Change. |
| [5] | **FNOFDis** | **Frame Number Overflow Disable**<br>0: Ignore<br>1: Disables interrupt generation due to Frame Number Overflow. |
| [4] | **URErrDis** | **Unrecoverable Error Disable**<br>This event is not implemented. All writes to this bit are ignored. |
| [3] | **ResuDis** | **Resume Detected Disable**<br>0: Ignore<br>1: Disables interrupt generation due to Resume Detected. |

| Bits | Descriptions | |
|---|---|---|
| [2] | **SOFDis** | **Start Of Frame Disable**<br>0: Ignore<br>1: Disables interrupt generation due to Start of Frame. |
| [1] | **WBDHDis** | **Write Back Done Head Disable**<br>0: Ignore<br>1: Disables interrupt generation due to Write-back Done Head. |
| [0] | **SchORDis** | **Scheduling Overrun Disable**<br>0: Ignore<br>1: Disables interrupt generation due to Scheduling Overrun. |

## Host Controller Communication Area Register (HcHCCA)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcHCCA | 0xB000_7018 | R/W | Host Controller Communication Area Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| HCCA | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HCCA | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HCCA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:7] | **HCCA** | **Host Controller Communication Area**<br>Pointer to HCCA base address. |

## Host Controller Period Current ED Register (HcPerCED)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcPerCED | 0xB000_701C | R/W | Host Controller Period Current ED Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PeriCED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PeriCED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PeriCED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PeriCED | | | | Reserved | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:4] | **PeriCED** | **Periodic Current ED**<br>Pointer to the current Periodic List ED. |

## Host Controller Control Head ED Register (HcCtrHED)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcCtrHED | 0xB000_7020 | R/W | Host Controller Control Head ED Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CtrlHED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CtrlHED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CtrlHED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CtrlHED | | | | Reserved | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:4] | **CtrlHED** | **Control Head ED**<br>Pointer to the Control List Head ED. |

## Host Controller Control Current ED Register (HcCtrCED)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcCtrCED | 0xB000_7024 | R/W | Host Controller Control Current ED Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CtrlCED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CtrlCED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CtrlCED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CtrlCED | | | | Reserved | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:4] | **CtrlCED** | **Control Current Head ED**<br>Pointer to the current Control List Head ED. |

## Host Controller Bulk Head ED Register (HcBlkHED)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcBlkHED | 0xB000_7028 | R/W | Host Controller Bulk Head ED Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| BlkHED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BlkHED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| BlkHED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BlkHED | | | | Reserved | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:4] | **BlkHED** | **Bulk Head ED**<br>Pointer to the Bulk List Head ED. |

### Host Controller Bulk Current Head ED Register (HcBlkCED)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcBlkCED | 0xB000_702C | R/W | Host Controller Bulk Current ED Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| BlkCED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BlkCED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| BlkCED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BlkCED | | | | Reserved | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:4] | **BlkCED** | **Bulk Current Head ED**<br>Pointer to the current Bulk List Head ED. |

### Host Controller Done Head Register (HcDoneH)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcDoneH | 0xB000_7030 | R/W | Host Controller Done Head Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DoneH | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DoneH | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DoneH | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DoneH | | | | Reserved | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:4] | **DoneH** | **Done Head**<br>Pointer to the current Done List Head ED. |

## Host Controller Frame Interval Register (HcFmIntv)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcFmIntv | 0xB000_7034 | R/W | Host Controller Frame Interval Register | 0x0000_2EDF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| FmIntvT | FSDPktCnt | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FSDPktCnt | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | FmInterval | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FmInterval | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31] | **FmIntvT** | **Frame Interval Toggle**<br>This bit is toggled by HCD when it loads a new value into **FrameInterval**. |
| [30: 16] | **FSDPktCnt** | **FS Largest Data Packet**<br>This field specifies a value that is loaded into the Largest Data Packet Counter at the beginning of each frame. |
| [13:0] | **FmInterval** | **Frame Interval**<br>This field specifies the length of a frame as (bit times - 1). For 12,000 bit times in a frame, a value of 11,999 is stored here. |

## Host Controller Frame Remaining Register (HcFmRem)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcFmRem | 0xB000_7038 | R | Host Controller Frame Remaining Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| FmRemT | Reserved | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | FmRemain | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FmRemain | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31] | **FmRemT** | **Frame Remaining Toggle**<br>Loaded with **FrameIntervalToggle** when **FrameRemaining** is loaded. |
| [13:0] | **FmRemain** | **Frame Remaining**<br>When the Host Controller is in the USBOPERATIONAL state, this 14-bit field decrements each 12 MHz clock period. When the count reaches 0, (end of frame) the counter reloads with **FrameInterval**. In addition, the counter loads when the Host Controller transitions into USBOPERATIONAL. |

## Host Controller Frame Number Register (HcFNum)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcFNum | 0xB000_703C | R | Host Controller Frame Number Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FmNum | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FmNum | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **FmNum** | **Frame Number**<br>This 16-bit incrementing counter field is incremented coincident with the loading of **FrameRemaining**. The count rolls over from 'FFFFh' to '0h.' |

## Host Controller Periodic Start Register (HcPerSt)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcPerSt | 0xB000_7040 | R/W | Host Controller Periodic Start Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | PeriStart | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PeriStart | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [13:0] | **PeriStart** | **Periodic Start**<br>This field contains a value used by the List Processor to determine where in a frame the Periodic List processing must begin. |

## Host Controller Root Hub Descriptor A Register (HcRhDeA)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcRhDeA | 0xB000_7048 | R/W | Host Controller Root Hub Descriptor A Register | 0x0100_0002 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PwrGDT | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | NOCP | OCPM | DevType | NPS | PSM |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DPortNum | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:24] | **PwrGDT** | **Power On to Power Good Time**<br>This field value is represented as the number of 2 ms intervals, which ensuring that the power switching is effective within 2 ms. Only bits [25:24] are implemented as R/W. The remaining bits are read only as '0'. It is not expected that these bits be written to anything other than 1h, but limited adjustment is provided. This field should be written to support system implementation. This field should always be written to a non-zero value. |
| [12] | **NOCP** | **No Over Current Protection**<br>This bit should be written to support the external system port over-current implementation.<br>0 = Over-current status is reported<br>1 = Over-current status is not reported |
| [11] | **OCPM** | **Over Current Protection Mode**<br>This bit should be written 0 and is only valid when **NOCP** bit is cleared.<br>0 = Global Over-Current<br>1 = Individual Over-Current |
| [10] | **DevType** | **Device Type** |

| Bits | Descriptions | |
|---|---|---|
| [9] | **NPS** | **No Power Switching**<br>This bit should be written to support the external system port power switching implementation.<br>0 = Ports are power switched.<br>1 = Ports are always powered on. |
| [8] | **PSM** | **Power Switching Mode**<br>This bit is only valid when **NoPowerSwitching** is cleared. This bit should be written '0'.<br>0 = Global Switching<br>1 = Individual Switching |
| [7:0] | **DPortNum** | **Number Downstream Ports** |

## Host Controller Root Hub Descriptor B Register (HcRhDeB)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcRhDeB | 0xB000_704C | R/W | Host Controller Root Hub Descriptor B Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PPCM | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PPCM | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DevRemove | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DevRemove | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:16] | **PPCM** | **Port Power Control Mask**<br>Global-power switching. This field is only valid if **NoPowerSwitching** is cleared and **PowerSwitchingMode** is set (individual port switching). When set, the port only responds to individual port power switching commands (**Set**/**ClearPortPower**). When cleared, the port only responds to global power switching commands (**Set**/**ClearGlobalPower**).<br>0 = Device not removable<br>1 = Global-power mask<br>Port Bit relationship - Unimplemented ports are reserved, read/write '0'.<br>0 : Reserved<br>1 : Port 1<br>2 : Port 2<br>...<br>15 : Port 15 |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **DevRemove** | **Device Removable**<br>table of none-4 ports default to removable devices.<br>0 = Device not removable<br>1 = Device removable<br>Port Bit relationship<br>0 : Reserved<br>1 : Port 1<br>2 : Port 2<br>...<br>15 : Port 15<br>Unimplemented ports are reserved, read/write '0'. |

## Host Controller Root Hub Status Register (HcRhSts)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcRhSts | 0xB000_7050 | R/W | Host Controller Root Hub Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RWEClr | Reserved | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | OCIC | LPSC |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DRWEn | Reserved | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | OC | LPS |

| Bits | Descriptions | |
|---|---|---|
| [31] | **RWEClr** | **Clear Remote Wakeup Enable**<br>Writing a '1' to this bit clears **DeviceRemoteWakeupEnable**. Writing a '1' has no effect. |
| [17] | **OCIC** | **Over Current Indicator Change**<br>This bit is set when **OverCurrentIndicator** changes. Writing a '1' clears this bit. Writing a '0' has no effect. |
| [16] | **LPSC** | (Read) **LocalPowerStatusChange**<br>Not supported. Always read '0'.<br>(Write) **SetGlobalPower**<br>Write a '1' issues a **SetGlobalPower** command to the ports. Writing a '0' has no effect. |
| [15] | **DRWEn** | (Read) **DeviceRemoteWakeupEnable**<br>This bit enables ports' **ConnectStatusChange** as a remote wakeup event.<br>0 = disabled<br>1 = enabled<br>(Write) **SetRemoteWakeupEnable**<br>Writing a '1' sets **DeviceRemoteWakeupEnable**. Writing a '0' has no effect. |

| Bits | Descriptions | |
|---|---|---|
| [1] | **OC** | **Over Current Indicator**<br>This bit reflects the state of the OVRCUR pin. This field is only valid if **NoOverCurrentProtection** and **OverCurrentProtectionMode** are cleared.<br>0 = No over-current condition<br>1 = Over-current condition |
| [0] | **LPS** | (Read) **LocalPowerStatus**<br>Not Supported. Always read '0'.<br>(Write) **ClearGlobalPower**<br>Writing a '1' issues a **ClearGlobalPower** command to the ports. Writing a '0' has no effect. |

## Host Controller Root Hub Port Status (HcRhPrt [1: 2])

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcRhPrt1 | 0xB000_7054 | R/W | Host Controller Root Hub Port Status [1] | 0x0000_0000 |
| HcRhPrt2 | 0xB000_7058 | R/W | Host Controller Root Hub Port Status [2] | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | PRSC | POCIC | PSSC | PESC | CSC |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | LSDev | PPS |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | PR | POC | PS | PE | CC |

| Bits | Descriptions | |
|---|---|---|
| [20] | **PRSC** | **Port Reset Status Change**<br>This bit indicates that the port reset signal has completed.<br>0 = Port reset is not complete.<br>1 = Port reset is complete. |
| [19] | **POCIC** | **Port Over Current Indicator Change**<br>This bit is set when **OverCurrentIndicator** changes. Writing a '1' clears this bit. Writing a '0' has no effect. |
| [18] | **PSSC** | **Port Suspend Status Change**<br>This bit indicates the completion of the selective resume sequence for the port.<br>0 = Port is not resumed.<br>1 = Port resume is complete. |
| [17] | **PESC** | **Port Enable Status Change**<br>This bit indicates that the port has been disabled due to a hardware event (cleared **PortEnableStatus**).<br>0 = Port has not been disabled.<br>1 = PortEnableStatus has been cleared. |

| Bits | Descriptions | |
|---|---|---|
| [16] | **CSC** | **Connect Status Change**<br>This bit indicates connect or disconnect event has been detected. Writing a '1' clears this bit. Writing a '0' has no effect.<br>0 = No connect/disconnect event.<br>1 = Hardware detection of connect/disconnect event.<br>Note: If DeviceRemoveable is set, this bit resets to '1'. |
| [9] | **LSDev** | (Read) **LowSpeedDeviceAttached**<br>This bit defines the speed (and bud idle) of the attached device. It is only valid when **CurrentConnectStatus** is set.<br>0 = Full Speed device<br>1 = Low Speed device<br>(Write) **ClearPortPower**<br>Writing a '1' clears **PortPowerStatus**. Writing a '0' has no effect |
| [8] | **PPS** | (Read) **PortPowerStatus**<br>This bit reflects the power state of the port regardless of the power switching mode.<br>0 = Port power is off.<br>1 = Port power is on.<br>Note: If **NoPowerSwitching** is set, this bit is always read as '1'.<br>(Write) **SetPortPower**<br>Writing a '1' sets **PortPowerStatus**. Writing a '0' has no effect. |
| [4] | **PR** | (Read) **PortResetStatus**<br>0 = Port reset signal is not active.<br>1 = Port reset signal is active.<br>(Write) **SetPortReset**<br>Writing a '1' sets **PortResetStatus**. Writing a '0' has no effect. |

| Bits | Descriptions | |
|---|---|---|
| [3] | **POC** | (Read) **PortOverCurrentIndicator**<br>table of none-2 supports global over-current reporting. This bit reflects the state of the OVRCUR pin dedicated to this port. This field is only valid if **NoOverCurrentProtection** is cleared and **OverCurrentProtectionMode** is set.<br>0 = No over-current condition<br>1 = Over-current condition<br>(Write) **ClearPortSuspend**<br>Writing a '1' initiates the selective resume sequence for the port. Writing a '0' has no effect. |
| [2] | **PS** | (Read) **PortSuspendStatus**<br>0 = Port is not suspended<br>1 = Port is selectively suspended<br>(Write) **SetPortSuspend**<br>Writing a '1' sets **PortSuspendStatus**. Writing a '0' has no effect. |
| [1] | **PE** | (Read) **PortEnableStatus**<br>0 = Port disabled.<br>1 = Port enabled.<br>(Write) **SetPortEnable**<br>Writing a '1' sets **PortEnableStatus**. Writing a '0' has no effect. |
| [0] | **CC** | (Read) **CurrentConnectStatus**<br>0 = No device connected.<br>1 = Device connected.<br>NOTE: If **DeviceRemoveable** is set (not removable) this bit is always '1'.<br>(Write) **ClearPortEnable**<br>Writing '1' a clears **PortEnableStatus**. Writing a '0' has no effect. |

## USB Operational Mode Enable Register (OpModEn)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| OpModEn | 0xB000_7204 | R/W | USB Operational Mode Enable Register | 0X0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | SIEPDis |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | OCALow | Reserved | ABORT | DBR16 |

| Bits | Descriptions | |
|---|---|---|
| [8] | **SIEPDis** | **SIE Pipeline Disable**<br>When set, waits for all USB bus activity to complete prior to returning completion status to the List Processor. This is a failsafe mechanism to avoid potential problems with the clk_dr transition between 1.5 MHz and 12 MHz. |
| [3] | **OCALow** | **Over Current Active Low**<br>This bit controls the polarity of over current flag from external power IC.<br>0: Over current flag is high active<br>1: Over current flag is low active |
| [1] | **ABORT** | **AHB Bus ERROR Response**<br>This bit indicates there is an ERROR response received in AHB bus.<br>0: No ERROR response received<br>1: ERROR response received |
| [0] | **DBR16** | **Data Buffer Region 16**<br>When set, the size of the data buffer region is 16 bytes. Otherwise, the size is 32 bytes. |

## 7.8 USB 2.0 Device Controller

The NUC950ADN USB Device Controller is compliant to the USB Specification version 2.0. It also supports the software control for device remote-wakeup and 6 configurable endpoints in addition to Control Endpoint. Each of these endpoints can be Isochronous, Bulk or Interrupt and they can be either of IN or OUT direction with maximum packet size up to 1024 bytes. Three different modes of operation (Auto validation mode, manual validation mode and Fly mode) are supported for IN-endpoint.

### 7.8.1 USB Device Register Group Summary

| Register Groups | Description |
|---|---|
| **Main Control Registers** | These set of registers control the global enable of interrupts and maintain the status of the interrupts |
| **USB Control Registers** | These set of registers control the USB related events to/from the USB host and hold the status of the USB events. |
| **Control Endpoint Registers** | These set of registers direct the control endpoint in handling the USB requests from the host and hold the status information of the transactions. |
| **Non control Endpoint Registers** | These set of registers configure, control and exhibit the status of the non-control endpoints' operation |
| **DMA Registers** | These registers are responsible for the DMA related operations |

### 7.8.2 USB Device Control Registers Map

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_BA = 0xB000_6000** | | | | |
| **IRQ_STAT** | 0xB000_6000 | R | Interrupt Register | 0x0000_0000 |
| **IRQ_ENB_L** | 0xB000_6008 | R/W | Interrupt Enable Low Register | 0x0000_0001 |
| **USB_IRQ_STAT** | 0xB000_6010 | R/W | USB Interrupt Status register | 0x0000_0000 |

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| USB_IRQ_ENB | 0xB000_6014 | R/W | USB Interrupt Enable register | 0x0000_0040 |
| USB_OPER | 0xB000_6018 | R/W | USB operational register | 0x0000_0002 |
| USB_FRAME_CNT | 0xB000_601C | R | USB frame count register | 0x0000_0000 |
| USB_ADDR | 0xB000_6020 | R/W | USB address register | 0x0000_0000 |
| CEP_DATA_BUF | 0xB000_6028 | R/W | Control-ep Data Buffer | 0x0000_0000 |
| CEP_CTRL_STAT | 0xB000_602C | R/W | Control-ep Control and Status | 0x0000_0000 |
| CEP_IRQ_ENB | 0xB000_6030 | R/W | Control-ep Interrupt Enable | 0x0000_0000 |
| CEP_IRQ_STAT | 0xB000_6034 | R/W | Control-ep Interrupt Status | 0x0000_1000 |
| IN_TRNSFR_CNT | 0xB000_6038 | R/W | In-transfer data count | 0x0000_0000 |
| OUT_TRNSFR_CNT | 0xB000_603C | R | Out-transfer data count | 0x0000_0000 |
| CEP_CNT | 0xB000_6040 | R | Control-ep data count | 0x0000_0000 |
| SETUP1_0 | 0xB000_6044 | R | Setupbyte1 & byte0 | 0x0000_0000 |
| SETUP3_2 | 0xB000_6048 | R | Setupbyte3 & byte2 | 0x0000_0000 |
| SETUP5_4 | 0xB000_604C | R | Setupbyte5 & byte4 | 0x0000_0000 |
| SETUP7_6 | 0xB000_6050 | R | Setupbyte7 & byte6 | 0x0000_0000 |
| CEP_START_ADDR | 0xB000_6054 | R/W | Control EP RAM start address | 0x0000_0000 |
| CEP_END_ADDR | 0xB000_6058 | R/W | Control EP RAM end address | 0x0000_0000 |
| DMA_CTRL_STS | 0xB000_605C | R/W | DMA control and status register | 0x0000_0000 |
| DMA_CNT | 0xB000_6060 | R/W | DMA count register | 0x0000_0000 |
| EPA_DATA_BUF | 0xB000_6064 | R/W | Endpoint A data register | 0x0000_0000 |
| EPA_IRQ_STAT | 0xB000_6068 | R/W | Endpoint A Interrupt status register | 0x0000_0002 |
| EPA_IRQ_ENB | 0xB000_606C | R/W | Endpoint A Interrupt enable register | 0x0000_0000 |
| EPA_DATA_CNT | 0xB000_6070 | R | Data count available in endpoint A buffer | 0x0000_0000 |
| EPA_RSP_SC | 0xB000_6074 | R/W | Endpoint A response register set/clear | 0x0000_0000 |
| EPA_MPS | 0xB000_6078 | R/W | Endpoint A maximum packet size register | 0x0000_0000 |
| EPA_CNT | 0xB000_607C | R/W | Endpoint A transfer count register | 0x0000_0000 |
| EPA_CFG | 0xB000_6080 | R/W | Endpoint A configuration register | 0x0000_0012 |
| EPA_START_ADDR | 0xB000_6084 | R/W | Endpoint A RAM start address | 0x0000_0000 |
| EPA_END_ADDR | 0xB000_6088 | R/W | Endpoint A RAM end address | 0x0000_0000 |

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| EPB_DATA_BUF | 0xB000_608C | R/W | Endpoint B data register | 0x0000_0000 |
| EPB_IRQ_STAT | 0xB000_6090 | R/W | Endpoint B Interrupt status register | 0x0000_0002 |
| EPB_IRQ_ENB | 0xB000_6094 | R/W | Endpoint B Interrupt enable register | 0x0000_0000 |
| EPB_DATA_CNT | 0xB000_6098 | R | Data count available in endpoint B buffer | 0x0000_0000 |
| EPB_RSP_SC | 0xB000_609C | R/W | Endpoint B response register set/clear | 0x0000_0000 |
| EPB_MPS | 0xB000_60A0 | R/W | Endpoint B maximum packet size register | 0x0000_0000 |
| EPB_TRF_CNT | 0xB000_60A4 | R/W | Endpoint B transfer count register | 0x0000_0000 |
| EPB_CFG | 0xB000_60A8 | R/W | Endpoint B configuration register | 0x0000_0022 |
| EPB_START_ADDR | 0xB000_60AC | R/W | Endpoint B RAM start address | 0x0000_0000 |
| EPB_END_ADDR | 0xB000_60B0 | R/W | Endpoint B RAM end address | 0x0000_0000 |
| EPC_DATA_BUF | 0xB000_60B4 | R/W | Endpoint C data register | 0x0000_0000 |
| EPC_IRQ_STAT | 0xB000_60B8 | R/W | Endpoint C Interrupt status register | 0x0000_0002 |
| EPC_IRQ_ENB | 0xB000_60BC | R/W | Endpoint C Interrupt enable register | 0x0000_0000 |
| EPC_DATA_CNT | 0xB000_60C0 | R | Data count available in endpoint C buffer | 0x0000_0000 |
| EPC_RSP_SC | 0xB000_60C4 | R/W | Endpoint C response register set/clear | 0x0000_0000 |
| EPC_MPS | 0xB000_60C8 | R/W | Endpoint C maximum packet size register | 0x0000_0000 |
| EPC_TRF_CNT | 0xB000_60CC | R/W | Endpoint C transfer count register | 0x0000_0000 |
| EPC_CFG | 0xB000_60D0 | R/W | Endpoint C configuration register | 0x0000_0032 |
| EPC_START_ADDR | 0xB000_60D4 | R/W | Endpoint C RAM start address | 0x0000_0000 |
| EPC_END_ADDR | 0xB000_60D8 | R/W | Endpoint C RAM end address | 0x0000_0000 |
| EPD_DATA_BUF | 0xB000_60DC | R/W | Endpoint D data register | 0x0000_0000 |
| EPD_IRQ_STAT | 0xB000_60E0 | R/W | Endpoint D Interrupt status register | 0x0000_0002 |
| EPD_IRQ_ENB | 0xB000_60E4 | R/W | Endpoint D Interrupt enable register | 0x0000_0000 |
| EPD_DATA_CNT | 0xB000_60E8 | R | Data count available in endpoint D buffer | 0x0000_0000 |
| EPD_RSP_SC | 0xB000_60EC | R/W | Endpoint D response register set/clear | 0x0000_0000 |
| EPD_MPS | 0xB000_60F0 | R/W | Endpoint D maximum packet size register | 0x0000_0000 |

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| EPD_TRF_CNT | 0xB000_60F4 | R/W | Endpoint D transfer count register | 0x0000_0000 |
| EPD_CFG | 0xB000_60F8 | R/W | Endpoint D configuration register | 0x0000_0042 |
| EPD_START_ADDR | 0xB000_60FC | R/W | Endpoint D RAM start address | 0x0000_0000 |
| EPD_END_ADDR | 0xB000_6100 | R/W | Endpoint D RAM end address | 0x0000_0000 |
| EPE_DATA_BUF | 0xB000_6104 | R/W | Endpoint E data register | 0x0000_0000 |
| EPE_IRQ_STAT | 0xB000_6108 | R/W | Endpoint E Interrupt status register | 0x0000_0002 |
| EPE_IRQ_ENB | 0xB000_610C | R/W | Endpoint E Interrupt enable register | 0x0000_0000 |
| EPE_DATA_CNT | 0xB000_6110 | R | Data count available in endpoint E buffer | 0x0000_0000 |
| EPE_RSP_SC | 0xB000_6114 | R/W | Endpoint E response register set/clear | 0x0000_0000 |
| EPE_MPS | 0xB000_6118 | R/W | Endpoint E maximum packet size register | 0x0000_0000 |
| EPE_TRF_CNT | 0xB000_611C | R/W | Endpoint E transfer count register | 0x0000_0000 |
| EPE_CFG | 0xB000_6120 | R/W | Endpoint E configuration register | 0x0000_0052 |
| EPE_START_ADDR | 0xB000_6124 | R/W | Endpoint E RAM start address | 0x0000_0000 |
| EPE_END_ADDR | 0xB000_6128 | R/W | Endpoint E RAM end address | 0x0000_0000 |
| EPF_DATA_BUF | 0xB000_612C | R/W | Endpoint F data register | 0x0000_0000 |
| EPF_IRQ_STAT | 0xB000_6130 | R/W | Endpoint F Interrupt status register | 0x0000_0002 |
| EPF_IRQ_ENB | 0xB000_6134 | R/W | Endpoint F Interrupt enable register | 0x0000_0000 |
| EPF_DATA_CNT | 0xB000_6138 | R | Data count available in endpoint F buffer | 0x0000_0000 |
| EPF_RSP_SC | 0xB000_613C | R/W | Endpoint F response register set/clear | 0x0000_0000 |
| EPF_MPS | 0xB000_6140 | R/W | Endpoint F maximum packet size register | 0x0000_0000 |
| EPF_TRF_CNT | 0xB000_6144 | R/W | Endpoint F transfer count register | 0x0000_0000 |
| EPF_CFG | 0xB000_6148 | R/W | Endpoint F configuration register | 0x0000_0062 |
| EPF_START_ADDR | 0xB000_614C | R/W | Endpoint F RAM start address | 0x0000_0000 |
| EPF_END_ADDR | 0xB000_6150 | R/W | Endpoint F RAM end address | 0x0000_0000 |
| USB_DMA_ADDR | 0xB000_6700 | R/W | AHB_DMA address register | 0x0000_0000 |
| USB_PHY_CTL | 0xB000_6704 | R/W | USB PHY control register | 0x0000_0260 |

### 7.8.3 USB Device Control Registers

## Interrupt Register (IRQ)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| IRQ | 0xB000_6000 | R | Interrupt Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EPF_INT | EPE_INT | EPD_INT | EPC_INT | EPB_INT | EPA_INT | CEP_INT | USB_INT |

| Bits | Descriptions | |
|---|---|---|
| [7] | **EPF_INT** | **This bit conveys the interrupt for Endpoints F.**<br>When set, the corresponding Endpoint F's interrupt status register should be read to determine the cause of the interrupt. |
| [6] | **EPE_INT** | **This bit conveys the interrupt for Endpoints E.**<br>When set, the corresponding Endpoint E's interrupt status register should be read to determine the cause of the interrupt. |
| [5] | **EPD_INT** | **This bit conveys the interrupt for Endpoints D.**<br>When set, the corresponding Endpoint D's interrupt status register should be read to determine the cause of the interrupt. |
| [4] | **EPC_INT** | **This bit conveys the interrupt for Endpoints C.**<br>When set, the corresponding Endpoint C's interrupt status register should be read to determine the cause of the interrupt. |

| Bits | Descriptions | |
|---|---|---|
| [3] | **EPB_INT** | T**his bit conveys the interrupt for Endpoints B.**<br>When set, the corresponding Endpoint B's interrupt status register should be read to determine the cause of the interrupt. |
| [2] | **EPA_INT** | **This bit conveys the interrupt for Endpoints A**.<br>When set, the corresponding Endpoint A's interrupt status register should be read to determine the cause of the interrupt. |
| [1] | **CEP_INT** | **Control Endpoint Interrupt.**<br>This bit conveys the interrupt status for control endpoint. When set, Control-ep's interrupt status register should be read to determine the cause of the interrupt. |
| [0] | **USB_INT** | **USB Interrupt.**<br>The interrupt status for USB specific events endpoint. When set, USB interrupt status register should be read to determine the cause of the interrupt. |

## Interrupt Enable Low Register (IRQ_ENB_L)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| IRQ_ENB_L | 0xB000_6008 | R/W | Interrupt Enable Low Register | 0x0000_0001 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EPF_IE | EPE_IE | EPD_IE | EPC_IE | EPB_IE | EPA_IE | CEP_IE | USB_IE |

| Bits | Descriptions | |
|---|---|---|
| [7] | **EPF_IE** | **Interrupt Enable for Endpoint F.**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint F |
| [6] | **EPE_IE** | **Interrupt Enable for Endpoint E.**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint E |
| [5] | **EPD_IE** | **Interrupt Enable for Endpoint D.**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint D |
| [4] | **EPC_IE** | **Interrupt Enable for Endpoint C.**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint C |
| [3] | **EPB_IE** | **Interrupt Enable for Endpoint B.**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint B |

| Bits | Descriptions | |
|---|---|---|
| [2] | **EPA_IE** | **Interrupt Enable for Endpoint A.**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint A. |
| [1] | **CEP_IE** | **Control Endpoint Interrupt Enable.**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the control endpoint. |
| [0] | **USB_IE** | **USB Interrupt Enable.**<br>When set, this bit enables a local interrupt to be generated when a USB event occurs on the bus. |

## USB Interrupt Status Register (USB_IRQ_STAT)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| USB_IRQ_STAT | 0xB000_6010 | R/W | USB Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | TCLKOK_IS | DMACOM_IS | HISPD_IS | SUS_IS | RUM_IS | RST_IS | SOF_IS |

| Bits | Descriptions | |
|---|---|---|
| [6] | **TCLKOK_IS** | **Usable Clock Interrupt.**<br>This bit is set when usable clock is available from the transceiver. Writing '1" clears this bit. |
| [5] | **DMACOM_IS** | **DMA Completion Interrupt.**<br>This bit is set when the DMA transfer is over. Writing '1" clears this bit. |
| [4] | **HISPD_IS** | **High Speed Settle.**<br>This bit is set when the valid high-speed reset protocol is over and the device has settled is high-speed. Writing '1" clears this bit. |
| [3] | **SUS_IS** | **Suspend Request.**<br>This bit is set as default and it has to be cleared by writing '1' before the USB reset. This bit is also set when a USB Suspend request is detected from the host. Writing '1' clears this bit. |
| [2] | **RUM_IS** | **Resume.**<br>When set, this bit indicates that a device resume has occurred. Writing a '1' clears this bit. |

| Bits | Descriptions | |
|---|---|---|
| [1] | **RST_IS** | **Reset Status.**<br>When set, this bit indicates that either the USB root port reset is end. Writing a '1' clears this bit. |
| [0] | **SOF_IS** | **SOF.**<br>This bit indicates when a start-of-frame packet has been received. Writing a '1' clears this bit. |

## USB Interrupt Enable Register (USB_IRQ_ENB)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| USB_IRQ_ENB | 0xB000_6014 | R/W | USB Interrupt Enable Register | 0x0000_0040 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | TCLKOK_IE | DMACOM_IE | HISPD_IE | SUS_IE | RUM_IE | RST_IE | SOF_IE |

| Bits | Descriptions | |
|---|---|---|
| [6] | **TCLKOK_IE** | **Usable Clock Interrupt.**<br>This bit enables the usable clock interrupt. |
| [5] | **DMACOM_IE** | **DMA Completion Interrupt.**<br>This bit enables the DMA completion interrupt |
| [4] | **HISPD_IE** | **High Speed Settle.**<br>This bit enables the high-speed settle interrupt. |
| [3] | **SUS_IE** | **Suspend Request**.<br>This bit enables the Suspend interrupt. |
| [2] | **RUM_IE** | **Resume.**<br>This bit enables the Resume interrupt. |
| [1] | **RST_IE** | **Reset Status.**<br>This bit enables the USB-Reset interrupt. |
| [0] | **SOF_IE** | **SOF Interrupt.** This bit enables the SOF interrupt. |

## USB Operational Register (USB_OPER)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| USB_OPER | 0xB000_6018 | R/W | USB Operational Register | 0x0000_0002 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | CUR_SPD | SET_HISPD | GEN_RUM |

| Bits | Descriptions | |
|---|---|---|
| [2] | **CUR_SPD** | **USB Current Speed.**<br>When set, this bit indicates that the DEVICE CONTROLLER has settled in High Speed and a zero indicates that the device has settled in Full Speed |
| [1] | **SET_HISPD** | **USB High Speed.**<br>When set to one, this bit indicates the DEVICE CONTROLLER to initiate a chirp-sequence during reset protocol. When set to zero, it indicates the DEVICE CONTROLLER to suppress the chirp-sequence during reset protocol, thereby allowing the DEVICE CONTROLLER to settle in full-speed, even though it is connected to a USB2.0 Host. |
| [0] | **GEN_RUM** | **Generate Resume.**<br>Writing a 1 to this bit causes a Resume sequence to be initiated to the host if device remote wakeup is enabled. This bit is self-clearing. |

## USB Frame Count Register (USB_FRAME_CNT)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| USB_FRAME_CNT | 0xB000_601C | R | USB Frame Count Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | FRAME_CNT | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FRAME_CNT | | | | | MFRAME_CNT | | |

| Bits | Descriptions | |
|---|---|---|
| [13:3] | **FRAME_CNT** | **FRAME COUNTER.**<br>This field contains the frame count from the most recent start-of-frame packet. |
| [2:0] | **MFRAME_CNT** | **MICRO FRAME COUNTER.**<br>This field contains the micro-frame number for the frame number in the frame counter field. |

## USB Address Register (USB_ADDR)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| USB_ADDR | 0xB000_6020 | R/W | USB Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | ADDR | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [6:0] | **ADDR** | This field contains the current USB address of the device. This field is cleared when a root port reset is detected. |

## Control-ep Data Buffer (CEP_DATA_BUF)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| CEP_DATA_BUF | 0xB000_6028 | RW | Control-ep Data Buffer | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DATA_BUF | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA_BUF | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **DATA_BUF** | **Control-ep Data Buffer.**<br>Bits [15:8] of this register provide the high order byte and bits [7:0] of this register provide the lower order byte for the buffer transaction (read or write). |

## Control-ep Control and Status (CEP_CTRL_STAT)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| CEP_CTRL_STAT | 0xB000_602C | RW | Control-ep Control and Status | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | FLUSH | ZEROLEN | STLALL | NAK_CLEAR |

| Bits | Descriptions | |
|---|---|---|
| [3] | **FLUSH** | **CEP-FLUSH Bit.**<br>Writing 1 to this bit cause the packet buffer and its corresponding CEP_AVL_CNT register to be cleared. This bit is self-cleaning. |
| [2] | **ZEROLEN** | **ZEROLEN Bit.**<br>This bit is valid for auto validation mode only. When this bit is set, DEVICE CONTROLLER can send a zero length packet to the host during Data stage to an IN token. This bit gets cleared once the zero length data packet is sent. So, the local CPU need not write again to clear this bit. |
| [1] | **STLALL** | **STALL.**<br>This bit is a read/write bit. When this stall bit is set, the control endpoint sends a stall handshake in response to any in or out token thereafter. This is typically used for response to invalid/unsupported requests. When this bit is being set the NAK clear bit has to be cleared at the same time since the NAK clear bit has highest priority than STALL. It is automatically cleared on receipt of a next setup-token. So, the local CPU need not write again to clear this bit.<br>NOTE: ONLY when CPU write data [1:0] is 2'b10 or 2'b00, this bit can be updated. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **NAK_CLEAR** | **NAK_CLEAR.** This is a read/write bit. This bit plays a crucial role in any control transfer. It bit is set to one by the DEVICE CONTROLLER, whenever a setup token is received. The local CPU can take its own time to finish off any house-keeping work based on the request and then clear this bit. Unless the bit is being cleared by the local CPU by writing zero, the DEVICE CONTROLLER will be responding with NAKs for the subsequent status phase. This mechanism holds the host from moving to the next request, until the local CPU is also ready to process the next request.<br>NOTE: ONLY when CPU write data [1:0] is 2'b10 or 2'b00, this bit can be updated. |

### Control Endpoint Interrupt Enable (CEP_IRQ_ENABLE)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| CEP_IRQ_ENABLE | 0xB000_6030 | R/W | Control Endpoint Interrupt Enable | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | EMPTY_IE | FULL_IE | STACOM_IE | ERR_IE | STALL_IE |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| NAK_IE | DATA_RxED_IE | DATA_TxED_IE | PING_IE | IN_TK_IE | OUT_TK_IE | SETUP_PK_IE | SETUP_TK_IE |

| Bits | Descriptions | |
|---|---|---|
| [12] | **EMPTY_IE** | **Buffer Empty Interrupt.**<br>This bit enables the buffer empty interrupt. |
| [11] | **FULL_IE** | **Buffer Full Interrupt.**<br>This bit enables the buffer full interrupt. |
| [10] | **STACOM_IE** | **Status Completion Interrupt.**<br>This bit enables the Status Completion interrupt. |
| [9] | **ERR_IE** | **USB Error Interrupt.**<br>This bit enables the USB Error interrupt. |
| [8] | **STALL_IE** | **STALL Sent Interrupt.**<br>This bit enables the STALL sent interrupt |
| [7] | **NAK_IE** | **NAK Sent Interrupt.**<br>This bit enables the NAK sent interrupt. |
| [6] | **DATA_RxED_IE** | **Data Packet Received Interrupt.**<br>This bit enables the data received interrupt. |

| Bits | Descriptions | |
|---|---|---|
| [5] | **DATA_TxED_IE** | **Data Packet Transmitted Interrupt.**<br>This bit enables the data packet transmitted interrupt. |
| [4] | **PING_IE** | **Ping Token Interrupt.**<br>This bit enables the ping token interrupt. |
| [3] | **IN_TK_IE** | **In Token Interrupt.**<br>This bit enables the in token interrupt |
| [2] | **OUT_TK_IE** | **Out Token Interrupt.**<br>This bit enables the out token interrupt. |
| [1] | **SETUP_PK_IE** | **Setup Packet Interrupt.**<br>This bit enables the setup packet interrupt. |
| [0] | **SETUP_TK_IE** | **Setup Token Interrupt Enable.**<br>This bit enables the setup token interrupt. |

## Control-Endpoint Interrupt Status (CEP_IRQ_STAT)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| CEP_IRQ_STAT | 0xB000_6034 | R/W | Control-ep Interrupt Status | 0x0000_1000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | EMPTY_IS | FULL_IS | STACOM_IS | ERR_IS | STALL_IS |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| NAK_IS | DATA_RxED_IS | DATA_TxED_IS | PING_IS | IN_TK_IS | OUT_TK_IS | SETUP_PK_IS | SETUP_TK_IS |

| Bits | Descriptions | |
|---|---|---|
| [12] | **EMPTY_IS** | **Buffer Empty Interrupt.** (Read Only)<br>This bit is set when the control-ednpt buffer is empty. |
| [11] | **FULL_IS** | **Buffer Full Interrupt.** (Write "1" Clear)<br>This bit is set when the control-endpt buffer is full. |
| [10] | **STACOM_IS** | **Status Completion Interrupt.** (Write "1" Clear)<br>This bit is set when the status stage of a USB transaction has completed successfully. |
| [9] | **ERR_IS** | **USB Error Interrupt.** (Write "1" Clear)<br>This bit is set when an error had occurred during the transaction. |
| [8] | **STALL_IS** | **STALL Sent Interrupt.** (Write "1" Clear)<br>This bit is set when a stall-token is sent in response to an in/out token |
| [7] | **NAK_IS** | **NAK Sent Interrupt.** (Write "1" Clear)<br>This bit is set when a nak-token is sent in response to an in/out token |

| Bits | Descriptions | |
|---|---|---|
| [6] | **DATA_RxED_IS** | **Data Packet Received Interrupt**. (Write "1" Clear)<br>This bit is set when a data packet is successfully received from the host for an out-token and an ack is sent to the host. |
| [5] | **DATA_TxED_IS** | **Data Packet Transmitted Interrupt.** (Write "1" Clear)<br>This bit is set when a data packet is successfully transmitted to the host in response to an in-token and an ack-token is received for the same. |
| [4] | **PING_IS** | **Ping Token Interrupt.** (Write "1" Clear)<br>This bit is set when the control-endpt receives a ping token from the host. |
| [3] | **IN_TK_IS** | **In Token Interrupt.** (Write "1" Clear)<br>This bit is set when the control-endpt receives an in token from the host. |
| [2] | **OUT_TK_IS** | **Out Token Interrupt.** (Write "1" Clear)<br>This bit is set when the control-endpoint receives an out token from the host. |
| [1] | **SETUP_PK_IS** | **Setup Packet Interrupt.** (Write "1" Clear)<br>This bit is set when a setup packet has been received from the host. This bit must be cleared (by writing a 1) before the next setup packet can be received. If the bit is not cleared, then the successive setup packets will be overwritten in the setup packet buffer. |
| [0] | **SETUP_TK_IS** | **Setup Token Interrupt.** (Write "1" Clear)<br>This bit indicates when a setup token is received. Writing a 1 clears this status bit |

## In-transfer data count (IN_TRF_CNT)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| IN_TRF_CNT | 0xB000_6038 | R/W | In-transfer data count | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IN_TRF_CNT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [7:0] | **IN_TRF_CNT** | **In-transfer data count.**<br>There is no mode selection for the control endpoint (but it operates like manual mode).The local-CPU has to fill the control-endpoint buffer with the data to be sent for an in-token and to write the count of bytes in this register. When zero is written into this field, a zero length packet is sent to the host. When the count written in the register is more than the MPS, the data sent will be of only MPS. |

### Out-transfer data count (OUT_TRF_CNT)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| OUT_TRF_CNT | 0xB000_603C | R | Out-transfer data count | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| OUT_TRF_CNT | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUT_TRF_CNT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **OUT_TRF_CNT** | **Out-Transfer Data Count.**<br>The DEVICE CONTROLLER maintains the count of the data received in case of an out transfer, during the control transfer. |

### Control- Endpoint data count (CEP_CNT)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| CEP_CNT | 0xB000_6040 | R | Control-ep data count | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CEP_CNT | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CEP_CNT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **CEP_CNT** | **Control-ep Data Count.**<br>The DEVICE CONTROLLER maintains the count of the data of control-ep. |

### Setup1 & Setup0 bytes (SETUP1_0)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| SETUP1_0 | 0xB000_6044 | R | Setup1 & Setup0 bytes | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SETUP1 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SETUP0 | | | | | | | |

| Bits | Descriptions |
|---|---|

| | | |
|---|---|---|
| [15:8] | **SETUP1** | **Setup Byte 1[15:8].**<br>This register provides byte 1 of the last setup packet received. For a Standard Device Request, the following bRequest Code information is returned. |

| Code | Descriptions |
|---|---|
| 0x00 | Get Status |
| 0x01 | Clear Feature |
| 0x02 | Reserved |
| 0x03 | Set Feature |
| 0x04 | Reserved |
| 0x05 | Set Address |
| 0x06 | Get Descriptor |
| 0x07 | Set Descriptor |
| 0x08 | Get Configuration |
| 0x09 | Set Configuration |
| 0x0A | Get Interface |
| 0x0B | Set Interface |
| 0x0C | Synch Frame |

<table>
<tr>
<td>[7:0]</td>
<td>SETUP0</td>
<td>Setup Byte 0[7:0].<br>This register provides byte 0 of the last setup packet received. For a Standard Device Request, the following bmRequestType information is returned.
<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[7]</td><td>Direction</td><td>0 = host to device;<br>1 = device to host</td></tr>
<tr><td>[6:5]</td><td>Type</td><td>0 = Standard,<br>1 = Class,<br>2 = Vendor,<br>3 = Reserved</td></tr>
<tr><td>[4:0]</td><td>Recipient</td><td>0 = Device,<br>1 = Interface,<br>2 = Endpoint,<br>3 = Other,<br>4-31 Reserved</td></tr>
</table>
</td>
</tr>
</table>

### Setup3 & Setup2 bytes (SETUP3_2)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| SETUP3_2 | 0xB000_6048 | R | Setup3 & Setup2 bytes | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SETUP3 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SETUP2 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:8] | **SETUP3** | **Setup Byte 3 [15:8].**<br>This register provides byte 3 of the last setup packet received. For a Standard Device Request, the most significant byte of the wValue field is returned. |
| [7:0] | **SETUP2** | **Setup Byte 2 [7:0].**<br>This register provides byte 2 of the last setup packet received. For a Standard Device Request, the least significant byte of the wValue field is returned. |

### Setup5 & Setup4 bytes (SETUP5_4)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| SETUP5_4 | 0xB000_604C | R | Setup5 & Setup4 bytes | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SETUP5 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SETUP4 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:8] | **SETUP5** | **Setup Byte 5[15:8].**<br>This register provides byte 5 of the last setup packet received. For a Standard Device Request, the most significant byte of the wIndex field is returned. |
| [7:0] | **SETUP4** | **Setup Byte 4[7:0].**<br>This register provides byte 4 of the last setup packet received. For a Standard Device Request, the least significant byte of the wIndex is returned. |

### Setup7 & Setup6 bytes (SETUP7_6)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| SETUP7_6 | 0xB000_6050 | R | Setup7 & Setup6 bytes | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SETUP7 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SETUP6 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:8] | **SETUP7** | **Setup Byte 7[15:8].**<br>This register provides byte 7 of the last setup packet received. For a Standard Device Request, the most significant byte of the wLength field is returned. |
| [7:0] | **SETUP6** | **Setup Byte 6[7:0].**<br>This register provides byte 6 of the last setup packet received. For a Standard Device Request, the least significant byte of the wLength field is returned. |

### Control Endpoint RAM Start Address Register (CEP_START_ADDR)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| CEP_START_ADDR | 0xB000_6054 | R/W | Control EP RAM Start Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | CEP_START_ADDR | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CEP_START_ADDR | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [10:0] | **CEP_START_ADDR** | This is the start-address of the RAM space allocated for the control-endpoint |

## Control Endpoint RAM End Address Register (CEP_END_ADDR)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| CEP_END_ADDR | 0xB000_6058 | R/W | Control EP RAM End Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | CEP_END_ADDR | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CEP_END_ADDR | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [10:0] | **CEP_END_ADDR** | This is the end-address of the RAM space allocated for the control-endpoint |

## DMA Control Status Register (DMA_CTRL_STS)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| DMA_CTRL_STS | 0xB000_605C | R/W | DMA Control Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RST_DMA | SCAT_GA_EN | DMA_EN | DMA_RD | DMA_ADDR | | | |

| Bits | Descriptions | |
|---|---|---|
| [7] | **RST_DMA** | **Reset DMA state machine.** |
| [6] | **SCAT_GA_EN** | **Scatter gather function enable** |
| [5] | **DMA_EN** | **DMA Enable Bit** |
| [4] | **DMA_RD** | **DMA Operation Bit.**<br>If '1', the operation is a DMA read and if '0' the operation is a DMA write. |
| [3:0] | **DMA_ADDR** | **DMA ep_addr Bits** |

When enable scatter gather DMA function, SCAT_GA_EN needs to be set high and DMA_CNT set to 8 bytes. Then DMA will enable to fetch the descriptor which describes the real memory address and length. The descriptor will be a

8-byte format, like the following:

| [31] | [30] | [29:0] | |
|---|---|---|---|
| MEM_ADDR[31:0] | | | |
| EOT | RD | reserved | count[19:0] |

**MEM_ADDR**: It specifies the memory address (AHB address).

**EOT**: end of transfer. When this bit sets to high, it means this is the last descriptor.

**RD**: "1" means read from memory into buffer. "0" means read from buffer into memory.

## DMA Count Register (DMA_CNT)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| DMA_CNT | 0xB000_6060 | R/W | DMA Count Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | DMA_CNT | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DMA_CNT | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMA_CNT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [19:0] | **DMA_CNT** | The transfer count of the DMA operation to be performed is written to this register. |

## Endpoint A~F Data Register (EPA_DATA_BUF~ EPF_DATA_BUF)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| EPA_DATA_BUF | 0xB000_6064 | R/W | Endpoint A Data Register | 0x0000_0000 |
| EPB_DATA_BUF | 0xB000_608C | R/W | Endpoint B Data Register | 0x0000_0000 |
| EPC_DATA_BUF | 0xB000_60B4 | R/W | Endpoint C Data Register | 0x0000_0000 |
| EPD_DATA_BUF | 0xB000_60DC | R/W | Endpoint D Data Register | 0x0000_0000 |
| EPE_DATA_BUF | 0xB000_6104 | R/W | Endpoint E Data Register | 0x0000_0000 |
| EPF_DATA_BUF | 0xB000_612C | R/W | Endpoint F Data Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| EP_DATA_BUF | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EP_DATA_BUF | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **EP_DATA_BUF** | **Endpoint A~F Data Register.**<br>Bits [15:8] of this register provide the high order byte and bits [7:0] of this register provide the lower order byte for the buffer transaction (read or write). |

## Endpoint A~F Interrupt Status Register (EPA_IRQ_STAT~ EPF_IRQ_STAT)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| EPA_IRQ_STAT | 0xB000_6068 | R/W | Endpoint A Interrupt Status Register | 0x0000_0002 |
| EPB_IRQ_STAT | 0xB000_6090 | R/W | Endpoint B Interrupt Status Register | 0x0000_0002 |
| EPC_IRQ_STAT | 0xB000_60B8 | R/W | Endpoint C Interrupt Status Register | 0x0000_0002 |
| EPD_IRQ_STAT | 0xB000_60E0 | R/W | Endpoint D Interrupt Status Register | 0x0000_0002 |
| EPE_IRQ_STAT | 0xB000_6104 | R/W | Endpoint E Interrupt Status Register | 0x0000_0002 |
| EPF_IRQ_STAT | 0xB000_6130 | R/W | Endpoint F Interrupt Status Register | 0x0000_0002 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | O_SHORT_PKT_IS | ERR_IS | NYET_IS | STALL_IS | NAK_IS |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PING_IS | IN_TK_IS | OUT_TK_IS | DATA_RxED_IS | DATA_TxED_IS | SHORT_PKT_IS | EMPTY_IS | FULL_IS |

| Bits | Descriptions | |
|---|---|---|
| [12] | O_SHORT_PKT_IS | **Bulk Out Short Packet Received** (Writing a '1' clears this bit.)<br>Received bulk out short packet (including zero length packet ) |
| [11] | ERR_IS | **ERR Sent.** (Writing a '1' clears this bit.)<br>This bit is set when there occurs any error in the transaction. |
| [10] | NYET_IS | **NYET Sent.** (Writing a '1' clears this bit.)<br>This bit is set when the space available in the RAM is not sufficient to accommodate the next on coming data packet. |

| Bits | Descriptions | |
|---|---|---|
| [9] | **STALL_IS** | **USB STALL Sent.** (Writing a '1' clears this bit.)<br>The last USB packet could not be accepted or provided because the endpoint was stalled, and was acknowledged with a STALL. |
| [8] | **NAK_IS** | **USB NAK Sent.** (Writing a '1' clears this bit.)<br>The last USB IN packet could not be provided, and was acknowledged with a NAK. |
| [7] | **PING_IS** | **PING Token Interrupt.** (Writing a '1' clears this bit.)<br>This bit is set when a Data IN token has been received from the host. |
| [6] | **IN_TK_IS** | **Data IN Token Interrupt.** (Writing a '1' clears this bit.)<br>This bit is set when a Data IN token has been received from the host. |
| [5] | **OUT_TK_IS** | **Data OUT Token Interrupt.** (Writing a '1' clears this bit.)<br>This bit is set when a Data OUT token has been received from the host. This bit also set by PING tokens (in high-speed only). |
| [4] | **DATA_RxED_IS** | **Data Packet Received Interrupt.** (Writing a '1' clears this bit.)<br>This bit is set when a data packet is received from the host by the endpoint. |
| [3] | **DATA_TxED_IS** | **Data Packet Transmitted Interrupt.** (Writing a '1' clears this bit.)<br>This bit is set when a data packet is transmitted from the endpoint to the host. |
| [2] | **SHORT_PKT_IS** | **Short Packet Transferred Interrupt.** (Writing a '1' clears this bit.)<br>This bit is set when the length of the last packet was less than the Maximum Packet Size (EP_MPS). |
| [1] | **EMPTY_IS** | **Buffer Empty.** (READ ONLY)<br>For an IN endpoint, a buffer is available to the local side for writing up to FIFO full of bytes. This bit is set when the endpoint buffer is empty. For an OUT endpoint, the currently selected buffer has a count of 0, or no buffer is available on the local side (nothing to read). |

| Bits | Descriptions | |
|---|---|---|
| [0] | **FULL_IS** | **Buffer Full.** (READ ONLY)<br>This bit is set when the endpoint packet buffer is full. For an IN endpoint, the currently selected buffer is full, or no buffer is available to the local side for writing (no space to write). For an OUT endpoint, there is a buffer available on the local side, and there are FIFO full of bytes available to be read (entire packet is available for reading). |

Endpoint A~F Interrupt Enable Register (EPA_IRQ_ENB~ EPF_IRQ_ENB)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| EPA_IRQ_ENB | 0xB000_606C | R/W | Endpoint A Interrupt Enable Register | 0x0000_0000 |
| EPB_IRQ_ENB | 0xB000_6094 | R/W | Endpoint B Interrupt Enable Register | 0x0000_0000 |
| EPC_IRQ_ENB | 0xB000_60BC | R/W | Endpoint C Interrupt Enable Register | 0x0000_0000 |
| EPD_IRQ_ENB | 0xB000_60E4 | R/W | Endpoint D Interrupt Enable Register | 0x0000_0000 |
| EPE_IRQ_ENB | 0xB000_610C | R/W | Endpoint E Interrupt Enable Register | 0x0000_0000 |
| EPF_IRQ_ENB | 0xB000_6134 | R/W | Endpoint F Interrupt Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | O_SHORT_PKT_IE | ERR_IE | NYET_IE | STALL_IE | NAK_IE |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PING_IE | IN_TK_IE | OUT_TK_IE | DATA_RxED_IE | DATA_TxED_IE | SHORT_PKT_IE | EMPTY_IE | FULL_IE |

| Bits | Descriptions | |
|---|---|---|
| [12] | **O_SHORT_PKT_IE** | **Bulk Out Short Packet Interrupt Enable**<br>When set, this bit enables a local interrupt to be set whenever bulk out short packet occurs on the bus for this endpoint. |

| Bits | Descriptions | |
|---|---|---|
| [11] | **ERR_IE** | **ERR interrupt Enable.**<br>When set, this bit enables a local interrupt to be set whenever ERR condition occurs on the bus for this endpoint. |
| [10] | **NYET_IE** | **NYET Interrupt Enable.**<br>When set, this bit enables a local interrupt to be set whenever NYET condition occurs on the bus for this endpoint. |
| [9] | **STALL_IE** | **USB STALL Sent Interrupt Enable.**<br>When set, this bit enables a local interrupt to be set when a stall token is sent to the host. |
| [8] | **NAK_IE** | **USB NAK Sent Interrupt Enable.**<br>When set, this bit enables a local interrupt to be set when a nak token is sent to the host. |
| [7] | **PING_IE** | **PING Token Interrupt Enable.**<br>When set, this bit enables a local interrupt to be set when a ping token has been received from the host. |
| [6] | **IN_TK_IE** | **Data IN Token Interrupt Enable.**<br>When set, this bit enables a local interrupt to be set when a Data IN token has been received from the host. |
| [5] | **OUT_TK_IE** | **Data OUT Token Interrupt Enable.**<br>When set, this bit enables a local interrupt to be set when a Data OUT token has been received from the host. |
| [4] | **DATA_RxED_IE** | **Data Packet Received Interrupt Enable**.<br>When set, this bit enables a local interrupt to be set when a data packet has been transmitted to the host. |
| [3] | **DATA_TxED_IE** | **Data Packet Transmitted Interrupt Enable.**<br>When set, this bit enables a local interrupt to be set when a data packet has been received from the host. |
| [2] | **SHORT_PKT_IE** | **Short Packet Transferred Interrupt Enable.**<br>When set, this bit enables a local interrupt to be set when a short data packet has been transferred to/from the host. |

| Bits | Descriptions | |
|---|---|---|
| [1] | **EMPTY_IE** | **Buffer Empty Interrupt.**<br>When set, this bit enables a local interrupt to be set when a buffer empty condition is detected on the bus. |
| [0] | **FULL_IE** | **Buffer Full Interrupt.**<br>When set, this bit enables a local interrupt to be set when a buffer full condition is detected on the bus. |

### Endpoint A~F Data Available count register (EPA_DATA_CNT~ EPF_DATA_CNT)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| EPA_DATA_CNT | 0xB000_6070 | R | Endpoint A Data Available count register | 0x0000_0000 |
| EPB_DATA_CNT | 0xB000_6098 | R | Endpoint B Data Available count register | 0x0000_0000 |
| EPC_DATA_CNT | 0xB000_60C0 | R | Endpoint C Data Available count register | 0x0000_0000 |
| EPD_DATA_CNT | 0xB000_60E8 | R | Endpoint D Data Available count register | 0x0000_0000 |
| EPE_DATA_CNT | 0xB000_6110 | R | Endpoint E Data Available count register | 0x0000_0000 |
| EPF_DATA_CNT | 0xB000_6133 | R | Endpoint F Data Available count register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | DMA_LOOP | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_LOOP | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DATA_CNT | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA_CNT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [30:16] | DMA_LOOP | This register is the remaining DMA loop to complete. Each loop means 32-byte transfer. |
| [15:0] | DATA_CNT | For an OUT / IN endpoint, this register returns the number of valid bytes in the endpoint packet buffer. |

## Endpoint A~F Response Set/Clear Register (EPA_RSP_SC~ EPF_RSP_SC)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| EPA_RSP_SC | 0xB000_6074 | R/W | Endpoint A Response Set/Clear Register | 0x0000_0000 |
| EPB_RSP_SC | 0xB000_609C | R/W | Endpoint B Response Set/Clear Register | 0x0000_0000 |
| EPC_RSP_SC | 0xB000_60C4 | R/W | Endpoint C Response Set/Clear Register | 0x0000_0000 |
| EPD_RSP_SC | 0xB000_60EC | R/W | Endpoint D Response Set/Clear Register | 0x0000_0000 |
| EPE_RSP_SC | 0xB000_6114 | R/W | Endpoint E Response Set/Clear Register | 0x0000_0000 |
| EPF_RSP_SC | 0xB000_613C | R/W | Endpoint F Response Set/Clear Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DIS_BUF | PK_END | ZEROLEN | HALT | TOGGLE | MODE | | BUF_FLUSH |

| Bits | Descriptions | |
|---|---|---|
| [7] | DIS_BUF | **Disable Buffer**<br>**This bit is used to disable buffer (set buffer size to 1) when received a bulk out short packet.** |
| [6] | PK_END | **Packet End.**<br>This bit is applicable only in case of Auto-Validate Method. This bit is set to validate any remaining data in the buffer which is not equal to the MPS of the endpoint, and happens to be the last transfer. |

| Bits | Descriptions | |
|---|---|---|
| [5] | **ZEROLEN** | **Zerolen In.**<br>This bit is used to send a zero-length packet n response to an in-token. When this bit is set, a zero packet is sent to the host on reception of an in-token. |
| [4] | **HALT** | **Endpoint Halt.**<br>This bit is used to send a stall handshake as response to the token from the host. When an Endpoint Set Feature (ep_halt) is detected by the local CPU, it must write a '1' to this bit. |
| [3] | **TOGGLE** | **Endpoint Toggle.**<br>This bit is used to clear the endpoint data toggle bit. Reading this bit returns the current state of the endpoint data toggle bit.<br>The local CPU may use this bit, to initialize the end-point's toggle incase of reception of a Set Interface request or a Clear Feature (ep_halt) request from the host. Only when toggle bit is "1", this bit can be written into the inversed write data bit [3]. |
| [2:1] | **MODE** | **Mode**.<br>These two bits decide the mode of operation of the in-endpoint.<br>MODE[2:1] / Mode Description<br>2'b00 / Auto-Validate Mode<br>2'b01 / Manual-Validate Mode<br>2'b10 / Fly Mode<br>2'b11 / Reserved.<br>These bits are not valid for an out-endpoint. The auto validate mode will be activated when the reserved mode is selected.<br>(These modes are explained detailed in later sections) |
| [0] | **BUF_FLUSH** | **Buffer Flush.**<br>Writing a 1 to this bit causes the packet buffer to be flushed and the corresponding EP_AVAIL register to be cleared. This bit is self-clearing. This bit should always be written after a configuration event. |

## Endpoint A~F Maximum Packet Size Register (EPA_MPS~ EPF_MPS)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| EPA_MPS | 0xB000_6078 | R/W | Endpoint A Maximum Packet Size Register | 0x0000_0000 |
| EPB_MPS | 0xB000_60A0 | R/W | Endpoint B Maximum Packet Size Register | 0x0000_0000 |
| EPC_MPS | 0xB000_60C8 | R/W | Endpoint C Maximum Packet Size Register | 0x0000_0000 |
| EPD_MPS | 0xB000_60F0 | R/W | Endpoint D Maximum Packet Size Register | 0x0000_0000 |
| EPE_MPS | 0xB000_6118 | R/W | Endpoint E Maximum Packet Size Register | 0x0000_0000 |
| EPF_MPS | 0xB000_6140 | R/W | Endpoint F Maximum Packet Size Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | EP_MPS | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EP_MPS | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [10:0] | **EP_MPS** | **Endpoint Maximum Packet Size.**<br>This field determines the Endpoint Maximum Packet Size. |

## Endpoint A~F Transfer Count Register (EPA_TRF_CNT~ EPF_TRF_CNT)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| EPA_TRF_CNT | 0xB000_607C | R/W | Endpoint A Transfer Count Register | 0x0000_0000 |
| EPB_TRF_CNT | 0xB000_60A4 | R/W | Endpoint B Transfer Count Register | 0x0000_0000 |
| EPC_TRF_CNT | 0xB000_60CC | R/W | Endpoint C Transfer Count Register | 0x0000_0000 |
| EPD_TRF_CNT | 0xB000_60F4 | R/W | Endpoint D Transfer Count Register | 0x0000_0000 |
| EPE_TRF_CNT | 0xB000_611C | R/W | Endpoint E Transfer Count Register | 0x0000_0000 |
| EPF_TRF_CNT | 0xB000_6144 | R/W | Endpoint F Transfer Count Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | | | | EP_TRF_CNT | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EP_TRF_CNT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [10:0] | **EP_TRF_CNT** | For IN endpoints, this field determines the total number of bytes to be sent to the host in case of manual validation method.<br>For OUT endpoints, this field has no effect |

## Endpoint A~F Configuration Register (EPA_CFG~ EPF_CFG)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| EPA_CFG | 0xB000_6080 | R/W | Endpoint A Configuration Register | 0x0000_0012 |
| EPB_CFG | 0xB000_60A8 | R/W | Endpoint B Configuration Register | 0x0000_0022 |
| EPC_CFG | 0xB000_60D0 | R/W | Endpoint C Configuration Register | 0x0000_0032 |
| EPD_CFG | 0xB000_60F8 | R/W | Endpoint D Configuration Register | 0x0000_0042 |
| EPE_CFG | 0xB000_6120 | R/W | Endpoint E Configuration Register | 0x0000_0052 |
| EPF_CFG | 0xB000_6148 | R/W | Endpoint F Configuration Register | 0x0000_0062 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | EP_MULT | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EP_NUM | | | | EP_DIR | EP_TYPE | | EP_VALID |

| Bits | Descriptions | |
|---|---|---|
| [9:8] | **EP_MULT** | **MULT Field.**<br>This field indicates number of transactions to be carried out in one single micro frame. |

| [9:8] | Description |
|---|---|
| 0x00 | One transaction |
| 0x01 | Reserved |
| 0x10 | Reserved |
| 0x11 | Invalid |

| Bits | Descriptions | |
|---|---|---|
| [7:4] | **EP_NUM** | **Endpoint Number**.<br>This field selects the number of the endpoint. Valid numbers 1 to 15. |
| [3] | **EP_DIR** | **Endpoint Direction**.<br>EP_DIR = 0 - OUT EP (Host OUT to Device) EP_DIR = 1- IN EP (Host IN to Device) Note that a maximum of one OUT and IN endpoint is allowed for each endpoint number. |
| [2:1] | **EP_TYPE** | **Endpoint Type**.<br>This field selects the type of this endpoint. Endpoint 0 is forced to a Control type. |
| [0] | **EP_VALID** | **Endpoint Valid**.<br>When set, this bit enables this endpoint. This bit has no effect on Endpoint 0, which is always enabled. |

| [2:1] | Description |
|---|---|
| 0x00 | Reserved |
| 0x01 | Bulk |
| 0x10 | Interrupt |
| 0x11 | Isochronous |

## Endpoint A~F RAM Start Address Register (EPA_START_ADDR~ EPF_START_ADDR)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| EPA_START_ADDR | 0xB000_6084 | R/W | Endpoint A RAM Start Address Register | 0x0000_0000 |
| EPB_START_ADDR | 0xB000_60AC | R/W | Endpoint B RAM Start Address Register | 0x0000_0000 |
| EPC_START_ADDR | 0xB000_60D4 | R/W | Endpoint C RAM Start Address Register | 0x0000_0000 |
| EPD_START_ADDR | 0xB000_60FC | R/W | Endpoint D RAM Start Address Register | 0x0000_0000 |
| EPE_START_ADDR | 0xB000_6124 | R/W | Endpoint E RAM Start Address Register | 0x0000_0000 |
| EPF_START_ADDR | 0xB000_614C | R/W | Endpoint F RAM Start Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | | | | EP_START_ADDR | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EP_START_ADDR | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [10:0] | EP_START_ADDR | This is the start-address of the RAM space allocated for the endpoint A~F. |

### Endpoint A~F RAM End Address Register (EPA_END_ADDR~ EPF_END_ADDR)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| EPA_END_ADDR | 0xB000_6088 | R/W | Endpoint A RAM End Address Register | 0x0000_0000 |
| EPB_END_ADDR | 0xB000_60B0 | R/W | Endpoint B RAM End Address Register | 0x0000_0000 |
| EPC_END_ADDR | 0xB000_60D8 | R/W | Endpoint C RAM End Address Register | 0x0000_0000 |
| EPD_END_ADDR | 0xB000_6100 | R/W | Endpoint D RAM End Address Register | 0x0000_0000 |
| EPE_END_ADDR | 0xB000_6128 | R/W | Endpoint E RAM End Address Register | 0x0000_0000 |
| EPF_END_ADDR | 0xB000_6150 | R/W | Endpoint F RAM End Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | | | | EP_END_ADDR | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EP_END_ADDR | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [10:0] | **EP_END_ADDR** | This is the end-address of the RAM space allocated for the endpoint A~F. |

### USB Address Register (USB_DMA_ADDR)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| USB_DMA_ADDR | 0xB000_6700 | R/W | USB DMA address register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| USB_DMA_ADDR | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| USB_DMA_ADDR | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| USB_DMA_ADDR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| USB_DMA_ADDR | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **USB_DMA_ADDR** | It specifies the address from which the DMA has to read / write. The address must WORD (32-bit) aligned. |

### USB PHY Control (USB_PHY_CTL)

| Register | Address | R/W | Description | Default Value |
|---|---|---|---|---|
| USB_PHY_CTL | 0xB000_6704 | R/W | USB PHY control register | 0x0000_0260 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | Phy_suspend | Reserved |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [9] | Phy_suspend | Set this bit low will cause USB PHY suspended. |

### 7.8.4 DMA Controller (DMAC)

The DMA Controller provides a DMA (Direct Memory Access) function for FMI to exchange data between system memory (ex. SDRAM) and shared buffer (one 2048 bytes). Software just simply fills in the starting address and enables DMAC, and then you can let DMAC to handle the data transfer automatically.

There is one 2048 bytes shared buffer inside DMAC, separate into four 512 bytes ping-pong FIFO. It can provide multi-block transfers using ping-pong mechanism for FMI. Software can access these shared buffers directly when FMI is not in busy.

### 7.8.5 DMA Controller Registers Map

**R**: read only, **W**: write only, **R/W**: both read and write

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **Shared Buffer (DMAC_BA = 0xB000_C000)** | | | | |
| **FB_0**<br>**……**<br>**FB_511** | 0XB000_C000<br>……<br>0xB000_C7FC | R/W | Shared Buffer (FIFO) | N/A |
| **DMAC Registers (DMAC_BA = 0xB000_C000)** | | | | |
| **DMACCSR** | 0xB000_C800 | R/W | DMAC Control and Status Register | 0x0000_0000 |
| **DMACSAR2** | 0xB000_C808 | R/W | DMAC Transfer Starting Address Register 2 | 0x0000_0000 |
| **DMACBCR** | 0xB000_C80C | R | DMAC Transfer Byte Count Register | 0x0000_0000 |
| **DMACIER** | 0xB000_C810 | R/W | DMAC Interrupt Enable Register | 0x0000_0001 |
| **DMACISR** | 0xB000_C814 | R/W | DMAC Interrupt Status Register | 0x0000_0000 |

### 7.8.6 DMAC Registers

## DMAC Control and Status Register (DMACCSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| DMACCSR | 0xB000_C800 | R/W | DMAC Control and Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | FMI_BUSY | Reserved |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | SG_EN2 | Reserved | SW_RST | DMACEN |

| Bits | Descriptions | |
|---|---|---|
| [9] | FMI_BUSY | **FMI DMA Transfer is in progress**<br>This bit indicates if FMI is granted and doing DMA transfer or not.<br>0 = FMI DMA transfer is not in progress.<br>1 = FMI DMA transfer is in progress. |
| [3] | SG_EN2 | **Enable Scatter-Getter Function for FMI**<br>Enable DMA scatter-getter function or not.<br>0 = Normal operation. DMAC will treat the starting address in DMACSAR2 as starting pointer of a single block memory.<br>1 = Enable scatter-getter operation. DMAC will treat the starting address in DMACSAR2 as a starting address of Physical Address Descriptor (PAD) table. The format of these PADs will be described later. |
| [1] | SW_RST | **Software Engine Reset**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit will reset the internal state machine and pointers. The contents of control register will not be cleared. This bit will auto clear after few clock cycles. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **DMACEN** | **DMAC Engine Enable**<br>Setting this bit to 1 enables DMAC's operation. If this bit is cleared, DMAC will ignore all DMA request from FMI and force Bus Master into IDLE state.<br>0 = Disable DMAC.<br>1 = Enable DMAC.<br>NOTE: If target abort is occurred, DMACEN will be cleared. |

## DMAC Transfer Starting Address Register 2 (DMACSAR2)

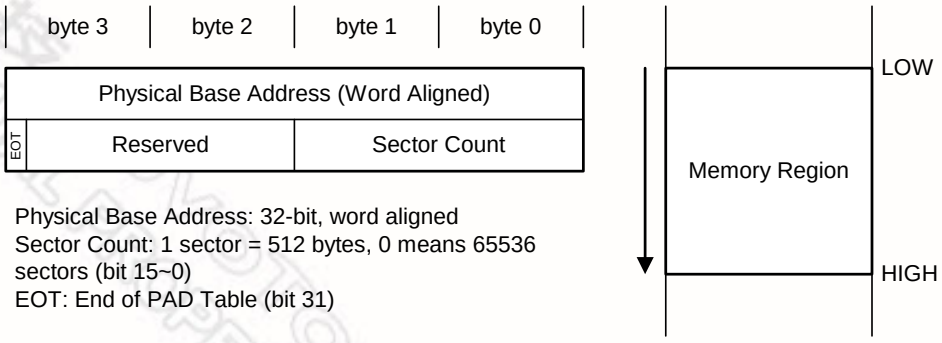
| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| DMACSAR2 | 0xB000_C808 | R/W | DMAC Transfer Starting Address Register 2 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DMACSA[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMACSA[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DMACSA[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMACSA[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | DMACSA | **DMA Transfer Starting Address for FMI**<br>This field indicates a 32-bit starting address of system memory (SRAM/SDRAM) for DMAC to retrieve or fill in data (for FMI engine).<br>If DMAC is not in normal mode, this field will be interpreted as a starting address of Physical Address Descriptor (PAD) table. |

**NOTE:** Starting address should be word alignment, for example, 0x0000_0000, 0x0000_0004…

The format of PAD table must like below. Note that the total sector count of all PADs must be equal to or greater than the sector count filled in FMI engine. EOT should be set to 1 in the last descriptor.



### DMAC Transfer Byte Count Register (DMACBCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **DMACBCR** | 0xB000_C80C | R | DMAC Transfer Byte Count Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | BCNT[25:24] | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BCNT[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| BCNT[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BCNT[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [25:0] | **BCNT** | **DMA Transfer Byte Count (Read Only)**<br>This field indicates the remained byte count of DMAC transfer. The value of this field is valid only when FMI is busy; otherwise, it is zero. |

### DMAC Interrupt Enable Register (DMACIER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| DMACIER | 0xB000_C810 | R/W | DMAC Interrupt Enable Register | 0x0000_0001 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | WEOT_IE | TABORT_IE |

| Bits | Descriptions | |
|---|---|---|
| [1] | WEOT_IE | **Wrong EOT Encountered Interrupt Enable**<br>• 0 = Disable interrupt generation when wrong EOT is encountered.<br>• 1 = Enable interrupt generation when wrong EOT is encountered. |
| [0] | TABORT_IE | **DMA Read/Write Target Abort Interrupt Enable**<br>0 = Disable target abort interrupt generation during DMA transfer.<br>1 = Enable target abort interrupt generation during DMA transfer. |

## DMAC Interrupt Status Register (DMACISR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| DMACISR | 0xB000_C814 | R/W | DMAC Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | WEOT_IF | TABORT_IF |

| Bits | Descriptions | |
|---|---|---|
| [1] | **WEOT_IF** | **Wrong EOT Encountered Interrupt Flag**<br>When DMA Scatter-Getter function is enabled, and EOT of the descriptor is encountered before DMA transfer finished (that means the total sector count of all PAD is less than the sector count of FMI), this bit will be set.<br>0 = No EOT encountered before DMA transfer finished.<br>1 = EOT encountered before DMA transfer finished.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [0] | **TABORT_IF** | **DMA Read/Write Target Abort Interrupt Flag**<br>0 = No bus ERROR response received.<br>1 = Bus ERROR response received.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

**NOTE:** When DMAC's bus master received ERROR response, it means that target abort is happened. DMAC will stop transfer and respond this event to software, FMI; and then go to IDLE state. When target abort occurred or WEOT_IF is set, suggest software reset DMAC and IP, and then transfer those data again.

## 7.9 Flash Memory Interface Controller (FMI)

The Flash Memory Interface (FMI) supports Secure Digital (SD and SDIO), Memory Stick (Memory stick PRO) and NAND-type flash. FMI is cooperated with DMAC to provide a fast data transfer between system memory and cards. There is one single 2048-byte buffer embedded in DMAC for temporary data storage. Due to DMAC only has single channel, that means only one interface can be active at the same time.

### 7.9.1 FMI Controller Registers Map

**R**: read only, **W**: write only, **R/W**: both read and write

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI Global Registers (FMI_BA = 0xB000_D000)** | | | | |
| FMICSR | 0xB000_D000 | R/W | Global Control and Status Register | 0x0000_0000 |
| FMIIER | 0xB000_D004 | R/W | Global Interrupt Control Register | 0x0000_0001 |
| FMIISR | 0xB000_D008 | R/W | Global Interrupt Status Register | 0x0000_0000 |
| **Secure Digital Registers** | | | | |
| SDCSR | 0xB000_D020 | R/W | SD Control and Status Register | 0x0101_0000 |
| SDARG | 0xB000_D024 | R/W | SD Command Argument Register | 0x0000_0000 |
| SDIER | 0xB000_D028 | R/W | SD Interrupt Control Register | 0x0000_0000 |
| SDISR | 0xB000_D02C | R/W | SD Interrupt Status Register | 0x000X_008C |
| SDRSP0 | 0xB000_D030 | R | SD Receiving Response Token Register 0 | 0x0000_0000 |
| SDRSP1 | 0xB000_D034 | R | SD Receiving Response Token Register 1 | 0x0000_0000 |
| SDBLEN | 0xB000_D038 | R/W | SD Block Length Register | 0x0000_01FF |
| **Memory Stick Registers** | | | | |
| MSCSR | 0xB000_D060 | R/W | Memory Stick Control and Status Register | 0x0000_0008 |
| MSIER | 0xB000_D064 | R/W | Memory Stick Interrupt Control Register | 0x0000_0000 |
| MSISR | 0xB000_D068 | R/W | Memory Stick Interrupt Status Register | 0x0000_0000 |
| MSBUF1 | 0xB000_D06C | R/W | Memory Stick Register Buffer 1 | 0x0000_0000 |
| MSBUF2 | 0xB000_D070 | R/W | Memory Stick Register Buffer 2 | 0x0000_0000 |
| **NAND-type Flash Registers** | | | | |
| SMCSR | 0xB000_D0A0 | R/W | NAND Flash Control and Status Register | 0x0600_0080 |
| SMTCR | 0xB000_D0A4 | R/W | NAND Flash Timing Control Register | 0x0001_0105 |
| SMIER | 0xB000_D0A8 | R/W | NAND Flash Interrupt Control Register | 0x0000_0000 |

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMISR | 0xB000_D0AC | R/W | NAND Flash Interrupt Status Register | 0x000X_0000 |
| SMCMD | 0xB000_D0B0 | W | NAND Flash Command Port Register | N/A |
| SMADDR | 0xB000_D0B4 | W | NAND Flash Address Port Register | N/A |
| SMDATA | 0xB000_D0B8 | R/W | NAND Flash Data Port Register | N/A |
| SMECC0 | 0xB000_D0BC | R | NAND Flash Error Correction Code 0 Register | 0x0000_0000 |
| SMECC1 | 0xB000_D0C0 | R | NAND Flash Error Correction Code 1 Register | 0x0000_0000 |
| SMECC2 | 0xB000_D0C4 | R | NAND Flash Error Correction Code 2 Register | 0x0000_0000 |
| SMECC3 | 0xB000_D0C8 | R | NAND Flash a Error Correction Code 3 Register | 0x0000_0000 |
| SMRA_0<br>…<br>SMRA_15 | 0xB000_D0CC<br>…<br>0xB000_D108 | R/W | NAND Flash Redundant Area Register | 0xFFFF_FFFF |
| SMECCAD0 | 0xB000_D10C | R | NAND Flash ECC Correction Address 0 | 0x0000_0000 |
| SMECCAD1 | 0xB000_D110 | R | NAND Flash ECC Correction Address 1 | 0x0000_0000 |
| ECC4ST | 0xB000_D114 | R | ECC4 Correction Status | 0x0000_0000 |
| ECC4F1A1 | 0xB000_D118 | R | ECC4 Field 1 Error Address 1 | 0x0000_0000 |
| ECC4F1A2 | 0xB000_D11C | R | ECC4 Field 1 Error Address 2 | 0x0000_0000 |
| ECC4F1D | 0xB000_D120 | R | ECC4 Field 1 Error Data | 0x0000_0000 |
| ECC4F2A1 | 0xB000_D124 | R | ECC4 Field 2 Error Address 1 | 0x0000_0000 |
| ECC4F2A2 | 0xB000_D128 | R | ECC4 Field 2 Error Address 2 | 0x0000_0000 |
| ECC4F2D | 0xB000_D12C | R | ECC4 Field 2 Error Data | 0x0000_0000 |
| ECC4F3A1 | 0xB000_D130 | R | ECC4 Field 3 Error Address 1 | 0x0000_0000 |
| ECC4F3A2 | 0xB000_D134 | R | ECC4 Field 3 Error Address 2 | 0x0000_0000 |
| ECC4F3D | 0xB000_D138 | R | ECC4 Field 3 Error Data | 0x0000_0000 |
| ECC4F4A1 | 0xB000_D13C | R | ECC4 Field 4 Error Address 1 | 0x0000_0000 |
| ECC4F4A2 | 0xB000_D140 | R | ECC4 Field 4 Error Address 2 | 0x0000_0000 |
| ECC4F4D | 0xB000_D144 | R | ECC4 Field 4 Error Data | 0x0000_0000 |

### 7.9.2 Register Details

## Global Control and Status Register (FMICSR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMICSR | 0xB000_D000 | R/W | Global Control and Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | CF_EN | SM_EN | MS_EN | SD_EN | SW_RST |

| Bits | Descriptions | |
|---|---|---|
| [3] | SM_EN | **NAND-type Flash Functionality Enable**<br>0 = Disable SM functionality of FMI.<br>1 = Enable SM functionality of FMI. |
| [2] | MS_EN | **Memory Stick Functionality Enable**<br>0 = Disable MS functionality of FMI.<br>1 = Enable MS functionality of FMI. |
| [1] | SD_EN | **Secure Digital Functionality Enable**<br>0 = Disable SD functionality of FMI.<br>1 = Enable SD functionality of FMI. |
| [0] | SW_RST | **Software Engine Reset**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit will reset all FMI engines. The contents of control register will not be cleared. This bit will auto clear after few clock cycles. |

**NOTE:** Software can enable only one engine at one time, or FMI will work abnormal.

## Global Interrupt Control Register (FMIIER)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMIIER | 0xB000_D004 | R/W | Global Interrupt Control Register | 0x0000_0001 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | DTA_IE |

| Bits | Descriptions | |
|---|---|---|
| [0] | DTA_IE | **DMAC READ/WRITE Target Abort Interrupt Enable**<br>0 = Disable DMAC READ/WRITE target abort interrupt generation.<br>1 = Enable DMAC READ/WRITE target abort interrupt generation. |

## Global Interrupt Status Register (FMIISR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMIISR | 0xB000_D008 | R/W | Global Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | DTA_IF |

| Bits | Descriptions | |
|---|---|---|
| [0] | DTA_IF | **DMAC READ/WRITE Target Abort Interrupt Flag (Read Only)**<br>This bit indicates DMAC received an ERROR response from internal AHB bus during DMA read/write operation. When Target Abort is occurred, please reset all engine.<br>0 = No bus ERROR response received.<br>1 = Bus ERROR response received.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

**NOTE:** No matter interrupt enable is turn on or not, the interrupt flag will be set when target condition is occurred.

## SD Control and Status Register (SDCSR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDCSR | 0xB000_D020 | R/W | SD Control and Status Register | 0x0101_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | SDPORT | | Reserved | SDNWR | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BLK_CNT | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DBW | SW_RST | CMD_CODE | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLK_KEEP0 | CLK8_OE | CLK74_OE | R2_EN | DO_EN | DI_EN | RI_EN | CO_EN |

| Bits | Descriptions | |
|---|---|---|
| [30:29] | **SDPORT** | **SD Port Selection**<br>This field should be set to 00 |
| [27:24] | **SDNWR** | **$N_{WR}$ Parameter for Block Write Operation**<br>This value indicates the $N_{WR}$ parameter for data block write operation in clock counts. The actual clock cycle will be SDNWR+1. |
| [23:16] | **BLK_CNT** | **Block Counts to Be Transferred or Received**<br>This field contains the block counts for data-in and data-out transfer. For **READ_MULTIPLE_BLOCK** and **WRITE_MULTIPLE_BLOCK** command, software can use this function to accelerate data transfer and improve performance. Note that only when SDBLEN=0x1FF, this field is valid. Otherwise, block counts will be set to 1 inside SD host engine.<br>**NOTE:** Value 0x0 in this field means 256. |
| [15] | **DBW** | **SD Data Bus Width**<br>0 = Data bus width is 1-bit.<br>1 = Data bus width is 4-bit. |

| Bits | Descriptions | |
|---|---|---|
| [14] | **SW_RST** | **Software Engine Reset**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit will reset the internal state machine and counters. The contents of control register will not be cleared (but RI_EN, DI_EN, DO_EN and R2_EN will be cleared). This bit will be auto cleared after few clock cycles. |
| [13:8] | **CMD_CODE** | **SD Command Code**<br>This register contains the SD command code (0x00 – 0x3F). |
| [7] | **CLK_KEEP0** | **SD Clock Enable for Port 0**<br>0 = Disable SD clock generation.<br>1 = SD clock always keeps free running. |
| [6] | **CLK8_OE** | **Generating 8 Clock Cycles Output Enable**<br>0 = No effect.<br>1 = Enable, SD host will output 8 clock cycles.<br>**NOTE:** When this operation is finished, this bit will be cleared automatically. |
| [5] | **CLK74_OE** | **Initial 74 Clock Cycles Output Enable**<br>0 = No effect.<br>1 = Enable, SD host will output 74 clock cycles to SD card.<br>**NOTE:** When this operation is finished, this bit will be cleared automatically. |
| [4] | **R2_EN** | **Response R2 Input Enable**<br>0 = No effect. (Please use SDCSR [SW_RST] to clear this bit.)<br>1 = Enable, SD host will wait to receive a response R2 from SD card and store the response data into DMAC's flash buffer (exclude CRC-7).<br>**NOTE:** When the R2 response is finished, this bit is cleared automatically. |
| [3] | **DO_EN** | **Data Output Enable**<br>0 = No effect. (Please use SDCSR [SW_RST] to clear this bit.)<br>1 = Enable, SD host will transfer block data and the CRC-16 value to SD card.<br>**NOTE:** When the output operation is finished, this bit is cleared automatically. |

| Bits | Descriptions | |
|---|---|---|
| [2] | **DI_EN** | **Data Input Enable**<br>0 = No effect. (Please use SDCSR [SW_RST] to clear this bit.)<br>1 = Enable, SD host will wait to receive block data and the CRC-16 value from SD card.<br>**NOTE:** When the input operation is finished, this bit will be cleared automatically. |
| [1] | **RI_EN** | **Response Input Enable**<br>0 = No effect. (Please use SDCSR [SW_RST] to clear this bit.)<br>1 = Enable, SD host will wait to receive a response from SD card.<br>**NOTE:** When the response operation is finished, this bit is cleared automatically. |
| [0] | **CO_EN** | **Command Output Enable**<br>0 = No effect.<br>1 = Enable, SD host will output a command to SD card.<br>**NOTE:** When the command operation is finished, this bit is cleared automatically. |

## SD Command Argument Register (SDARG)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDARG | 0xB000_D024 | R/W | SD Command Argument Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| SD_CMD_ARG | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SD_CMD_ARG | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SD_CMD_ARG | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SD_CMD_ARG | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | SD_CMD_ARG | **SD Command Argument**<br>This register contains a 32-bit value specifies the argument of SD command from host controller to SD card. |

## SD Interrupt Control Register (SDIER)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDIER | 0xB000_D028 | R/W | SD Interrupt Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | CD0SRC | Reserved | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | WKUP_EN | DITO_IE | RITO_IE | Reserved | SDIO0_IE | Reserved | CD0_IE |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | CRC_IE | BLKD_IE |

| Bits | Descriptions | |
|---|---|---|
| [30] | **CD0SRC** | **SD0 Card Detect Source Selection**<br>• 0 = From SD0 card's DAT3 pin.<br>• 1 = From GPIO pin. |
| [14] | **WKUP_EN** | **Wake-Up Signal Generating Enable**<br>Enable/Disable wake-up signal generating of SD host when SDIO card (current using) issues an interrupt (wake-up) via DAT [1] to host.<br>0 = Disable.<br>1 = Enable. |
| [13] | **DITO_IE** | **Data Input Time-out Interrupt Enable**<br>Enable/Disable interrupt generation of SD controller when data input time-out. Time-out value is specified at **SDTMOUT**.<br>0 = Disable.<br>1 = Enable. |

| Bits | Descriptions | |
|---|---|---|
| [12] | **RITO_IE** | **Response Time-out Interrupt Enable**<br>Enable/Disable interrupt generation of SD controller when receiving response or R2 time-out. Time-out value is specified at **SDTMOUT**.<br>0 = Disable.<br>1 = Enable. |
| [10] | **SDIO0_IE** | **SDIO Interrupt Enable for Port 0**<br>Enable/Disable interrupt generation of SD host when SDIO card 0 issues an interrupt via DAT [1] to host.<br>0 = Disable.<br>1 = Enable. |
| [8] | **CD0_IE** | **SD0 Card Detection Interrupt Enable**<br>Enable/Disable interrupt generation of SD controller when card 0 is inserted or removed.<br>0 = Disable.<br>1 = Enable. |
| [1] | **CRC_IE** | **CRC-7, CRC-16 and CRC Status Error Interrupt Enable**<br>0 = SD host will not generate interrupt when CRC-7, CRC-16 and CRC status is error.<br>1 = SD host will generate interrupt when CRC-7, CRC-16 and CRC status is error. |
| [0] | **BLKD_IE** | **Block Transfer Done Interrupt Enable**<br>0 = SD host will not generate interrupt when data-in (out) transfer done.<br>1 = SD host will generate interrupt when data-in (out) transfer done. |

## SD Interrupt Status Register (SDISR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDISR | 0xB000_D02C | R/W | SD Interrupt Status Register | 0x000X_008C |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | Reserved | SD0DAT1 | Reserved | CDPS0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | DITO_IF | RITO_IF | Reserved | SDIO0_IF | Reserved | CD0_IF |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SDDAT0 | CRCSTAT | | | CRC-16 | CRC-7 | CRC_IF | BLKD_IF |

| Bits | Descriptions | |
|---|---|---|
| [18] | SD0DAT1 | **DAT1 Pin Status of SD0 (Read Only)**<br>This bit is the DAT1 pin status of SD0. |
| [16] | CDPS0 | **Card Detect Pin Status of SD0 (Read Only)**<br>This bit is the DAT3 pin status of SD0, and it is using for card detection. When there is a card inserted in or removed from SD0, software should check this bit to confirm if there is really a card insertion or remove. |
| [13] | DITO_IF | **Data Input Time-out Interrupt Flag (Read Only)**<br>This bit indicates that SD host counts to time-out value when receiving data (waiting start bit).<br>0 = Not time-out.<br>1 = Data input time-out.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

| Bits | Descriptions | |
|---|---|---|
| [12] | **RITO_IF** | **Response Time-out Interrupt Flag (Read Only)**<br>This bit indicates that SD host counts to time-out value when receiving response or R2 (waiting start bit).<br>0 = Not time-out.<br>1 = Response time-out.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [10] | **SDIO0_IF** | **SDIO 0 Interrupt Flag (Read Only)**<br>This bit indicates that SDIO card 0 issues an interrupt to host.<br>0 = No interrupt is issued by SDIO card 0.<br>1 = An interrupt is issued by SDIO card 0.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [8] | **CD0_IF** | **SD0 Card Detection Interrupt Flag (Read Only)**<br>This bit indicates that SD card 0 is inserted or removed. Only if SDIER [CD0_IE] is set to 1, this bit is active.<br>0 = No card is inserted or removed.<br>1 = There is a card inserted in or removed from SD0.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [7] | **SDDAT0** | **DAT0 Pin Status of Current Selected SD (Read Only)**<br>This bit is the DAT0 pin status of current selected SD port. |
| [6:4] | **CRCSTAT** | **CRC Status Value of Data-out Transfer (Read Only)**<br>SD host will record CRC status of data-out transfer. Software could use this value to identify what type of error is during data-out transfer.<br>010 = Positive CRC status.<br>101 = Negative CRC status<br>111 = SD card programming error occurs. |

| Bits | Descriptions | |
|---|---|---|
| [3] | **CRC-16** | **CRC-16 Check Status of Data-in Transfer (Read Only)**<br>SD host will check CRC-16 correctness after data-in transfer.<br>0 = Fault.<br>1 = OK. |
| [2] | **CRC-7** | **CRC-7 Check Status (Read Only)**<br>SD host will check CRC-7 correctness during each response in. If that response does not contain CRC-7 information (R3), then software should turn off SDIER [CRC_IE] and ignore this bit.<br>0 = Fault.<br>1 = OK. |
| [1] | **CRC_IF** | **CRC-7, CRC-16 and CRC Status Error Interrupt Flag (Read Only)**<br>This bit indicates that SD host has occurred CRC error during response in, data-in or data-out (CRC status error) transfer. When CRC error is occurred, software should reset SD engine. Some response (ex. R3) doesn't have CRC-7 information with it; SD host will still calculate CRC-7, get CRC error and set this flag. In this condition, software should ignore CRC error and clears this bit manually.<br>0 = No CRC error is occurred.<br>1 = CRC error is occurred.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [0] | **BLKD_IF** | **Block Transfer Done Interrupt Flag (Read Only)**<br>This bit indicates that SD host has finished data-in or data-out block transfer. If there is a CRC-16 error or incorrect CRC status during multiple block data transfer, the transfer will be broken and this bit will be set.<br>0 = Not finished yet.<br>1 = Done.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

### SD Receiving Response Token Register 0 (SDRSP0)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDRSP0 | 0xB000_D030 | R | SD Receiving Response Token Register 0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| SD_RSP_TK0 | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SD_RSP_TK0 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SD_RSP_TK0 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SD_RSP_TK0 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | SD_RSP_TK0 | **SD Receiving Response Token 0**<br>SD host controller will receive a response token for getting a reply from SD card when SDCSR [RI_EN] is set. This field contains response bit 47-16 of the response token. |

## SD Receiving Response Token Register 1 (SDRSP1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDRSP1 | 0xB000_D034 | R | SD Receiving Response Token Register 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SD_RSP_TK1 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [7:0] | SD_RSP_TK1 | **SD Receiving Response Token 1**<br>SD host controller will receive a response token for getting a reply from SD card when SDCSR [RI_EN] is set. This register contains the bit 15-8 of the response token. |

## SD Block Length Register (SDBLEN)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDBLEN | 0xB000_D038 | R/W | SD Block Length Register | 0x0000_01FF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | SDBLEN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SDBLEN | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [8:0] | SDBLEN | **SD BLOCK LENGTH in Byte Unit**<br>A 9-bit value specifies the SD transfer byte count. The actual byte count is equal to SDBLEN+1. |

## SD Response/Data-in Time-out Register (SDTMOUT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDTMOUT | 0xB000_D03C | R/W | SD Response/Data-in Time-out Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SDTMOUT | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SDTMOUT | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SDTMOUT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | SDTMOUT | **SD Response/Data-in Time-out Value**<br>A 24-bit value specifies the time-out counts of response and data input. SD host controller will wait start bit of response or data-in until this value reached. The time period is depended on SD engine clock frequency. Do not write a small number into this field, or you may never get response or data due to time-out.<br>**NOTE:** Fill 0x0 into this field will disable hardware time-out function. |

## Memory Stick Control and Status Register (MSCSR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MSCSR | 0xB000_D060 | R/W | Memory Stick Control and Status Register | 0x0000_0008 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | MSPORT | DSIZE | | DCNT | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | TPC | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | SERIAL | MSPRO | MS_GO | SW_RST |

| Bits | Descriptions | |
|---|---|---|
| [21] | **MSPORT** | **Memory Stick Port Selection**<br>This bit should be set to 0 |
| [20:19] | **DSIZE** | **Data Size for Transfer (for Memory Stick PRO Only)**<br>This field defines how many bytes should be transferred of following TPC codes. Data will be obtained from (stored in) DMAC's FIFO.<br>READ_SHORT_DATA and WRITE_SHORT_DATA.<br>00 = 32 Bytes.<br>01 = 64 Bytes.<br>10 = 128 Bytes.<br>11 = 256 Bytes.<br>**NOTE:** This field is invalid when other TPC codes are executed. |

| Bits | Descriptions | |
|---|---|---|
| [18:16] | **DCNT** | **Data Count Number (in Byte Unit)**<br>This field defines how many bytes should be transferred of following TPC codes. Data will be obtained from (stored in) MSBUF1 and MSBUF2.<br>READ_REG, GET_INT, WRITE_REG, SET_R/W_REG_ADRS, SET_CMD and EX_SET_CMD.<br>For example, when software wants to use SET_R/W_REG_ADRS, you should write 0x4 into this field; when you want to use SET_CMD, you should write 0x1 into this field, etc.<br>**NOTE:** Value 0x0 means 8 bytes should be transferred, and it is the largest length this core can provide. |
| [11:8] | **TPC** | **TPC Code of the Packet**<br>This field defines the TPC code of the packet which software wants to transfer. This core supports all TPC code of Memory Stick and Memory Stick PRO specification. The lower 4 bits of TPC (TPC Check Code) will be generated by hardware automatically. |
| [3] | **SERIAL** | **Serial or Parallel Mode**<br>0 = MS host is working at parallel mode.<br>1 = MS host is working at serial mode (Default). |
| [2] | **MSPRO** | **Memory Stick or Memory Stick PRO**<br>0 = Type of the card is Memory Stick.<br>1 = Type of the card is Memory Stick PRO. |
| [1] | **MS_GO** | **Trigger Memory Stick Core to Transfer Packet**<br>0 = Writing 0 to this bit has no effect.<br>1 = Trigger Memory Stick core to transfer packet. When TPC code is READ_REG, GET_INT, WRITE_REG, SET_R/W_REG_ADRS, SET_CMD or EX_SET_CMD, data will be obtained from (stored in) MSBUF1 and MSBUF2. When TPC code is READ_LONG_DATA (READ_PAGE_DATA), READ_SHORT_DATA, WRITE_LONG_DATA (WRITE_PAGE_DATA) or WRITE_SHORT_DATA, data will be obtained from (stored in) DMAC's FIFO. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **SW_RST** | **Software Engine Reset**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit will reset the internal state machine and counters. The contents of control register will not be cleared. This bit will be auto cleared after few clock cycles. |

## Memory Stick Interrupt Control Register (MSIER)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MSIER | 0xB000_D064 | R/W | Memory Stick Interrupt Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | Reserved | CD0_IE |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | CRC_IE | BSYTO_IE | INTTO_IE | MSINT_IE | PKT_IE |

| Bits | Descriptions | |
|---|---|---|
| [16] | CD0_IE | **MS Card Detection 0 Interrupt Enable**<br>Enable/Disable Interrupt generation of MS controller when card 0 is inserted or removed.<br>0 = Disable.<br>1 = Enable. |
| [4] | CRC_IE | **CRC-16 Error Interrupt Enable**<br>0 = the core will not generate interrupt when CRC-16 is error.<br>1 = the core will generate interrupt when CRC-16 is error. |
| [3] | BSYTO_IE | **Busy to Ready Check Timeout Interrupt Enable**<br>0 = Disable Busy to Ready check timeout interrupt.<br>1 = Enable Busy to Ready check timeout interrupt. |
| [2] | INTTO_IE | **INT Response Timeout Interrupt Enable**<br>0 = Disable INT response timeout interrupt generation.<br>1 = Enable INT response timeout interrupt generation. |

| Bits | Descriptions | |
|---|---|---|
| [1] | **MSINT_IE** | **Memory Stick Card's Interrupt Enable**<br>0 = the core will not generate interrupt when MS card generates INT.<br>1 = the core will generate interrupt when MS card generates INT.<br>**NOTE:** Software should set MSIER[INTTO_IE] to '1' to enable INT detection function of the core, and set this bit to '1' if you want to get INT from MS card. |
| [0] | **PKT_IE** | **Packet Transfer Done Interrupt Enable**<br>0 = the core will not generate interrupt when packet transfer is done.<br>1 = the core will generate interrupt when packet transfer is done. |

## Memory Stick Interrupt Status Register (MSISR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MSISR | 0xB000_D068 | R/W | Memory Stick Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | Reserved | CD0_ |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | Reserved | CD0_IF |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | CMDNK | BREQ | ERR | CED |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | CRC_IF | BSYTO_IF | INTTO_IF | MSINT_IF | PKT_IF |

| Bits | Descriptions | |
|---|---|---|
| [24] | CD0_ | **Pin Status of MS Card Detection 0 (Read Only)**<br>This is the pin status of MS card detection 0. When there is a card insertion or removal, software should check this bit to confirm if it is really a card insertion or removal.<br>NOTE: Software should perform de-bounce for card detection function. |
| [16] | CD0_IF | **MS Card Detection 0 Interrupt Flag (Read Only)**<br>This bit indicates that MS card 0 is inserted or removed. Only if MSIER [CD0_IE] is set, this bit is active; otherwise, this bit is invalid.<br>0 = No card is inserted or removed.<br>1 = There is a card inserted in or removed from MS0.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [11:8] | CMDNK<br>BREQ<br>ERR<br>CED | **INT Status of Memory Stick PRO (Read Only)**<br>These 4 bits indicates the INT status of Memory Stick PRO card (only for parallel mode). When MSIER [INTTO_IE] is set, the core will wait for INT signal from card. If the card is working at parallel mode; after INT is occurred (MSISR [MSINT_IF] is set), the contents of INT register can be informed by these bits.<br>**NOTE:** These bits are valid in parallel mode only. |

| Bits | Descriptions | |
|---|---|---|
| [4] | **CRC_IF** | **CRC-16 Error Interrupt Flag (Read Only)**<br>When the packet transfer is done, the core will compare the value of CRC-16 which it calculated and received. If CRC-16 value is not the same, this flag will be set. The comparison executes only for READ packet.<br>0 = CRC-16 ok.<br>1 = CRC-16 failed.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [3] | **BSYTO_IF** | **Busy to Ready Check Timeout Interrupt Flag (Read Only)**<br>This bit indicates that the core cannot detect RDY signal on DATA [0] pin during Handshake State. It means some errors are occurred during packet transfer. The maximum timeout duration for RDY signal is 16 SCLKs.<br>0 = No RDY timeout occurred.<br>1 = RDY timeout occurred.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [2] | **INTTO_IF** | **INT Response Timeout Interrupt Flag (Read Only)**<br>This bit indicates that the core cannot detect INT signal of MS card after a period of time. In Memory Stick, the maximum period is 100ms. In Memory Stick PRO, the maximum period is 3500ms. If INT timeout is occurred, it means the card maybe malfunction.<br>0 = INT detection is not timeout.<br>1 = INT detection is timeout, no INT signal occurred.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [1] | **MSINT_IF** | **Memory Stick Card's Interrupt Flag (Read Only)**<br>Memory Stick will generate INT signal after some TPC codes are executed, ex. SET_CMD. This bit indicates that Memory Stick has generated INT signal after TPC code execution. This core will check INT for software only when MSIER [INTTO_IE] is set to '1', or this bit is invalid.<br>0 = No INT signal is detected.<br>1 = INT signal is detected.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **PKT_IF** | **Packet Transfer Done Interrupt Flag (Read Only)**<br>This bit indicates that the whole packet transfer is done. The four states of Memory Stick are BS1, BS2, BS3 and BS0.<br>0 = Packet transfer is not done yet.<br>1 = Packet transfer is done.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

**NOTE:** No matter interrupt is enabled or not, the interrupt flag is set when target condition is occurred.

### Memory Stick Register Buffer 1 (MSBUF1)

### Memory Stick Register Buffer 2 (MSBUF2)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **MSBUF1** | 0xB000_D06C | R/W | Memory Stick Register Buffer 1 | 0x0000_0x0000 |
| **MSBUF2** | 0xB000_D070 | | Memory Stick Register Buffer 2 | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DATA[31:24] | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| DATA[23:16] | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| DATA[15:8] | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| DATA[7:0] | | | | | | | |

| Bits | Descriptions |
|---|---|

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **DATA** | **Data Content of Packet Transfer**<br>This field contains the data of READ/WRITE TPC codes. When software uses following TPC codes, data will be obtained from (stored in) this field.<br>READ_REG, GET_INT, WRITE_REG, SET_R/W_REG_ADRS, SET_CMD and EX_SET_CMD.<br>This core will always send (store) data from MSB of MSBUF2. For example, if software wants to WRITE a packet with 1 byte data, you should put the data at MSBUF2 [31:24] and write 0x1 into MSCSR [DCNT] then trigger the core. The order of transfer will be MSBUF2 [31], MSBUF2 [30] …, MSBUF2 [24]. If you want to WRITE a packet with 6 bytes data, you should put the data at MSBUF2 [31:0] and MSBUF1 [31:16] and write 0x6 into MSCSR [DCNT] then trigger the core. The order of transfer will be MSBUF2 [31:24] … MSBUF2 [7:0], MSBUF1 [31:24], MSBUF1 [23:16]. The same order will be applied to READ packet. |

| MSBUF1 | MSBUF2 |
|---|---|
| BYTE 5 | BYTE 1 |
| BYTE 6 | BYTE 2 |
| BYTE 7 | BYTE 3 |
| BYTE 8 | BYTE 4 |

## NAND Flash Control and Status Register (SMCSR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMCSR | 0xB000_D0A0 | R/W | NAND Flash Control and Status Register | 0x0600_0080 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | SM_CS | | WP_ |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | MECC4 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ECC4CHK | Reserved | ECC4_EN | DBW | PSIZE | DWR_EN | DRD_EN | SW_RST |

| Bits | Descriptions | |
|---|---|---|
| [26:25] | **SM_CS** | **NAND Flash Select**<br>00 = Select card 0. (–CE0 will be active)<br>01 = Reserved<br>11 = No card will be selected. |
| [24] | **WP_** | **Write Protect Pin Control**<br>0 = Force –WP pin to **LOW** (0) level.<br>1 = Force –WP pin to **HIGH** (1) level. |
| [11:8] | **MECC4** | **Mask ECC4 During Write Page Data**<br>These 4 bits indicate NAND controller to write out ECC4 checksum or just 10 bytes 0xFF for each field.<br>0 = Do not mask the ECC4 checksum for each field.<br>1 = Mask ECC4 checksum and write out 10 bytes 0xFF to NAND. |

| Bits | Descriptions | |
|---|---|---|
| [7] | **ECC4CHK** | **None Used Field ECC4 Check After Read Page Data**<br>0 = Disable. NAND controller will always check ECC4 result for each field, no matter it is used or not.<br>1 = Enable. NAND controller will check 1's count for byte 2, 3 of redundant data in each field. If count value is greater than 8, NAND controller will treat this field as none used field; otherwise, it's used. If that field is none used field, NAND controller will ignore its ECC4 check result. |
| [5] | **ECC4_EN** | **ECC Algorithm Selection**<br>This bit is used to select the ECC algorithm for data protecting. There are two ECC algorithms inside this NAND controller, one is the standard used in SmartMedia's specification and the other is Reed-Solomon code.<br>0 = Using standard algorithm in SmartMedia specification.<br>1 = Using Reed-Solomon code encode/decode. |
| [4] | **DBW** | **SM Data Bus Width**<br>0 = Data bus width of NAND is 8-bit.<br>1 = Data bus width of NAND is reserved for 16-bit.<br>This bit should be set to 0. |
| [3] | **PSIZE** | **Page Size of NAND-type Flash**<br>This bit indicates the page size of NAND. Only two sizes are supported.<br>0 = Page size is 512 Bytes. (512+16B)<br>1 = Page size is 2048 Bytes. (2048+64B) |
| [2] | **DWR_EN** | **DMA Write Data Enable**<br>This bit enables the SM host to transfer data from DMAC's embedded frame buffer into SmartMedia card or NAND type flash.<br>0 = No effect.<br>1 = Enable DMA read data transfer.<br>**NOTE:** When DMA transfer completed, this bit will be cleared automatically. |

| Bits | Descriptions | |
|---|---|---|
| [1] | **DRD_EN** | **DMA Read Data Enable**<br>This bit enables the SM host to transfer data from SmartMedia card or NAND type flash into DMAC's embedded frame buffer.<br>0 = No effect.<br>1 = Enable DMA read data transfer.<br>**NOTE:** When DMA transfer completed, this bit will be cleared automatically. |
| [0] | **SW_RST** | **Software Engine Reset**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit will reset the internal state machine and counters (include SMCSR [DWR_EN] and SMCSR [DRD_EN]). The contents of control register will not be cleared. This bit will be auto cleared after few clock cycles. |

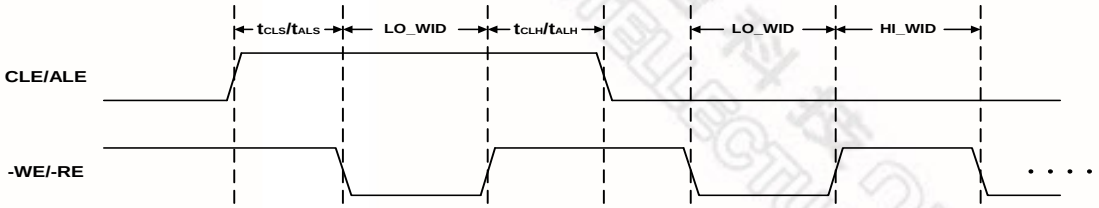
## NAND Flash Timing Control Register (SMTCR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMTCR | 0xB000_D0A4 | R/W | NAND Flash a Timing Control Register | 0x0001_0105 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | CALE_SH | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HI_WID | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LO_WID | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [22:16] | CALE_SH | **CLE/ALE Setup/Hold Time**<br>This field controls the CLE/ALE setup/hold time to –WE.<br>The setup/hold time can be calculated using following equation:<br>$t_{CLS}$ = (CALE_SH+1)*$T_{AHB}$<br>$t_{CLH}$ = ((CALE_SH*2)+2)*$T_{AHB}$<br>$t_{ALS}$ = (CALE_SH+1)*$T_{AHB}$<br>$t_{ALH}$ = ((CALE_SH*2)+2)*$T_{AHB}$ |
| [15:8] | HI_WID | **Read/Write Enable Signal High Pulse Width**<br>This field controls the high pulse width of signals –RE and –WE while H/W mode page access is enabled. The pulse width is a multiple of period of AHB bus clock. ( The actual width time will be [clock period*(HI_WID+1)] )<br>**NOTE:** Value of this field can not be 0x0. |
| [7:0] | LO_WID | **Read/Write Enable Signal Low Pulse Width**<br>This field controls the low pulse width of signals –RE and –WE while H/W mode page access is enabled. The pulse width is a multiple of period of AHB bus clock. ( The actual width time will be [clock period*(LO_WID+1)] )<br>**NOTE:** Value of this field can not be 0x0. |

**NOTE1:** The reset value is calculated base on 100MHz AHB Clock. Timing Effect of Above 3 Registers



## NAND Flash Interrupt Control Register (SMIER)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMIER | 0xB000_D0A8 | R/W | NAND Flash Interrupt Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | RB_IE | Reserved | CD0_IE |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | ECC_IE | DMA_IE |

| Bits | Descriptions | |
|---|---|---|
| [10] | RB_IE | **Ready/-Busy Rising Edge Detect Interrupt Enable**<br>0 = Disable R/-B rising edge detect interrupt generation.<br>1 = Enable R/-B rising edge detect interrupt generation. |
| [8] | CD0_IE | **NAND Flash # 0 Detection Interrupt Enable**<br>Enable/Disable interrupt generation of the controller when NAND Flash # 0 is inserted or removed.<br>0 = Disable.<br>1 = Enable. |
| [1] | ECC_IE | **ECC Check Error Interrupt Enable**<br>When reading data from SM card, SM host will check the ECC code inside redundant area with the ECC code which calculated by itself. Enable this bit to generate interrupt when there is an ECC code mismatch.<br>0 = Disable.<br>1 = Enable. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **DMA_IE** | **DMA Read / Write Data Complete Interrupt Enable**<br>0 = Disable DMA read/write data complete interrupt generation.<br>1 = Enable DMA read/write data complete interrupt generation. |

## NAND Flash Interrupt Status Register (SMISR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMISR | 0xB000_D0AC | R/W | NAND Flash Interrupt Status Register | 0x000X_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | RB_ | Reserved | CD0_ |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | RB_IF | Reserved | CD0_IF |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | ECC_IF | DMA_IF |

| Bits | Descriptions | |
|---|---|---|
| [18] | RB_ | **Ready/-Busy Pin Status (Read Only)**<br>This bit reflects the Ready/-Busy pin status of SmartMedia card. |
| [16] | CD0_ | **Card Detect Pin Status of SM0 (Read Only)**<br>This bit is the card detect pin status of SM0, and it is using for card detection. When there is a card inserted in or removed from SM0, software should check this bit to confirm if there is really a card insertion or remove. |
| [10] | RB_IF | **Ready/-Busy Rising Edge Detect Interrupt Flag (Read Only)**<br>0 = R/-B rising edge is not detected.<br>1 = R/-B rising edge is detected.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [8] | CD0_IF | **SM Card 0 Detection Interrupt Flag (Read Only)**<br>This bit indicates that SM card 0 is inserted or removed. Only if SMIER [CD0_IE] is set to 1, this bit is active.<br>0 = No card is inserted or removed.<br>1 = There is a card inserted in or removed from SM0.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

| Bits | Descriptions | |
|---|---|---|
| [1] | **ECC_IF** | **ECC Check Error Interrupt Flag (Read Only)**<br>0 = No ECC mismatch occurred.<br>1 = ECC mismatch occurred.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [0] | **DMA_IF** | **DMA Read / Write Data Complete Interrupt Flag (Read Only)**<br>0 = DMA read/write transfer is not finished yet.<br>1 = DMA read/write transfer is done.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

### NAND Flash Command Port Register (SMCMD)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMCMD | 0xB000_D0B0 | W | NAND Flash Command Port Register | N/A |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SMCMD | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [7:0] | SMCMD | **NAND Flash Command Port**<br>When CPU writes this port, SM H/W circuit will send a command to NAND Flash. |

## NAND Flash Address Port Register (SMADDR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMADDR | 0xB000_D0B4 | W | NAND Flash Address Port Register | N/A |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| EOA | Reserved | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SMADDR | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31] | **EOA** | **End of Address**<br>Writing to this bit to tell SM host if this address is the last one or not. When software first writes to address port with this bit cleared, SM host will set ALE pin to active (HIGH). After the last address is written (with this bit set), SM host will set ALE pin to inactive (LOW).<br>0 = Not the last address.<br>1 = The last one address. |
| [7:0] | **SMADDR** | NAND Flash **Address Port**<br>When CPU writes this port, SM H/W circuit will send an address to NAND Flash. |

## NAND Flash Data Port Register (SMDATA)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMDATA | 0xB000_D0B8 | R/W | NAND Flash Data Port Register | N/A |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SMDATA | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [7:0] | **SMDATA** | NAND Flash **Data Port**<br>CPU can access NAND Flash memory through this data port. |

## NAND Flash Error Correction Code 0 Register (SMECC0)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMECC0 | 0xB000_D0BC | R | NAND Flash Error Correction Code 0 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SMECC0 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SMECC0 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SMECC0 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | **SMECC0** | NAND Flash **ECC 0**<br>For 512+16 bytes/page models, this area contains a 3-byte ECC for page data from data byte 0 through byte 255.<br>[23:16]: CP5 ~ CP0, 0x3<br>[15:8]: LP15 ~ LP08<br>[7:0]: LP07 ~ LP00<br>For 2048+64 bytes/page models, this area contains a 3-byte ECC for page data from data byte 0 through byte 511.<br>[23:16]: CP5 ~ CP0, LP17 ~ LP16<br>[15:8]: LP15 ~ LP08<br>[7:0]: LP07 ~ LP00 |

## NAND Flash Error Correction Code 1 Register (SMECC1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMECC1 | 0xB000_D0C0 | R | NAND Flash Error Correction Code 1 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SMECC1 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SMECC1 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SMECC1 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | **SMECC1** | NAND Flash **ECC 1**<br>For 512+16 bytes/page models, this area contains a 3-byte ECC for page data from data byte 256 through byte 511.<br>[23:16]: CP5 ~ CP0, 0x3<br>[15:8]: LP15 ~ LP08<br>[7:0]: LP07 ~ LP00<br>For 2048+64 bytes/page models, this area contains a 3-byte ECC for page data from data byte 512 through byte 1023.<br>[23:16]: CP5 ~ CP0, LP17 ~ LP16<br>[15:8]: LP15 ~ LP08<br>[7:0]: LP07 ~ LP00 |

### NAND Flash Error Correction Code 2 Register (SMECC2)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMECC2 | 0xB000_D0C4 | R | NAND Flash Error Correction Code 2 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SMECC2 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SMECC2 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SMECC2 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | **SMECC2** | NAND Flash **ECC 2**<br>For 2048+64 bytes/page models only, this area contains a 3-byte ECC for page data from data byte 1024 through byte 1535.<br>[23:16]: CP5 ~ CP0, LP17 ~ LP16<br>[15:8]: LP15 ~ LP08<br>[7:0]: LP07 ~ LP00 |

## NAND Flash Error Correction Code 3 Register (SMECC3)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMECC3 | 0xB000_D0C8 | R | NAND Flash Error Correction Code 3 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SMECC3 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SMECC3 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SMECC3 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | **SMECC3** | NAND Flash **ECC 3**<br>For 2048+64 bytes/page models only, this area contains a 3-byte ECC for page data from data byte 1536 through byte 2047.<br>[23:16]: CP5 ~ CP0, LP17 ~ LP16<br>[15:8]: LP15 ~ LP08<br>[7:0]: LP07 ~ LP00 |

## NAND Flash Redundant Area Register (SMRA)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMRA_0<br>.....<br>SMRA_15 | 0xB000_D0CC<br>.....<br>0xB000_D108 | R/W | NAND Flash Redundant Area Register | 0xFFFF_FFFF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RArea | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RArea | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RArea | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RArea | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **RArea** | **Redundant Area**<br>This field keeps the 64 bytes data of redundant area for flash memory whose page size is 2K bytes.<br>For 512+16 byte/page models, only the 16 bytes redundant area is required. (I.e. Those 16 bytes redundant data must be programmed into SMRA_0, SMRA_1, SMRA_2 and SMRA_3.)<br>For 2048+64 byte/page models, the redundant data is programmed into SMRA_0 to SMRA_15. |

## NAND Flash ECC Correction Address 0 (SMECCAD0)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMECCAD0 | 0xB000_D10C | R | NAND Flash ECC Correction Address 0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| F2_STAT | | Reserved | | F2_ADDR | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F2_ADDR | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| F1_STAT | | Reserved | | F1_ADDR | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F1_ADDR | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:30] | **F2_STAT** | **ECC Status of ECC-Field 2**<br>This field contains the ECC correction status of ECC-field 2 (for page size 512+16B and 2048+64B).<br>● 00 = No error.<br>● 01 = Correctable error.<br>● 10 = Uncorrectable error.<br>● 11 = ECC Code error (for 1-bit code error only). |
| [27:16] | **F2_ADDR** | **Error Location of Received Data (Field 2)**<br>This field contains the error address result after ECC correct calculation. F2_ADDR [11:3] contains the byte address and F2_ADDR [2:0] contains the bit address of 256/512 bytes data. |
| [15:14] | **F1_STAT** | **ECC Status of ECC-Field 1**<br>This field contains the ECC correction status of ECC-field 1 (for page size 512+16B and 2048+64B).<br>● 00 = No error.<br>● 01 = Correctable error.<br>● 10 = Uncorrectable error.<br>● 11 = ECC Code error (for 1-bit code error only). |

| Bits | Descriptions | |
|---|---|---|
| [11:0] | **F1_ADDR** | **Error Location of Received Data (Field 1)**<br>This field contains the error address result after ECC correct calculation. F1_ADDR [11:3] contains the byte address and F1_ADDR [2:0] contains the bit address of 256/512 bytes data. For example, if F1_ADDR = 0x01E, the error bit will be located at byte 3, bit 6. Software can correct the data by inverting that bit. |

**NOTE:** NAND Flash host provide 1-bit error correction and 2-bit error detection. If there are more than two data error bits, the status could be correctable or uncorrectable. And also, if there are two or more code error bits, the status could be uncorrectable or ECC code error.

## NAND Flash ECC Correction Address 1 (SMECCAD1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SMECCAD1 | 0xB000_D110 | R | NAND Flash ECC Correction Address 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| F4_STAT | | Reserved | | F4_ADDR | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F4_ADDR | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| F3_STAT | | Reserved | | F3_ADDR | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F3_ADDR | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:30] | **F4_STAT** | **ECC Status of ECC-Field 4**<br>This field contains the ECC correction status of ECC-field 4 (for page size 2048+64B).<br>● 00 = No error.<br>● 01 = Correctable error.<br>● 10 = Uncorrectable error.<br>● 11 = ECC Code error (for 1-bit code error only). |
| [27:16] | **F4_ADDR** | **Error Location of Received Data (Field 4)**<br>This field contains the error address result after ECC correct calculation. F4_ADDR [11:3] contains the byte address and F4_ADDR [2:0] contains the bit address of 512 bytes data. |
| [15:14] | **F3_STAT** | **ECC Status of ECC-Field 3**<br>This field contains the ECC correction status of ECC-field 3 (for page size 2048+64B).<br>● 00 = No error.<br>● 01 = Correctable error.<br>● 10 = Uncorrectable error.<br>● 11 = ECC Code error (for 1-bit code error only). |

| Bits | Descriptions | |
|---|---|---|
| [11:0] | **F3_ADDR** | **Error Location of Received Data (Field 3)**<br>This field contains the error address result after ECC correct calculation. F3_ADDR [11:3] contains the byte address and F3_ADDR [2:0] contains the bit address of 512 bytes data. |

**NOTE:** NAND Flash host provide 1-bit error correction and 2-bit error detection. If there are more than two data error bits, the status could be correctable or uncorrectable. And also, if there are two or more code error bits, the status could be uncorrectable or ECC code error.

## ECC4 Correction Status (ECC4ST)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4ST | 0xB000_D114 | R | ECC4 Correction Status | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | F4_ECNT | | | F4_STAT | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | F3_ECNT | | | F3_STAT | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | F2_ECNT | | | F2_STAT | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | F1_ECNT | | | F1_STAT | |

| Bits | Descriptions | |
|---|---|---|
| [28:26] | **F4_ECNT** | **Error Count of ECC-Field 4**<br>This field contains the error counts after ECC4 correct calculation. For this ECC4 core, it can correct up to 4 errors in a single field. Only when **F4_STAT** equals to 0x01, the value in this field is meaningful. The value could be 0x1 ~ 0x4; it means one error to four errors. The error address will be put at **ECC4F4A1** and **ECC4F4A2**. |
| [25:24] | **F4_STAT** | **ECC4 Status of ECC-Field 4**<br>This field contains the ECC4 correction status of ECC-field 4 (for page size 2048+64B).<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error. |
| [20:18] | **F3_ECNT** | **Error Count of ECC-Field 3**<br>This field contains the error counts after ECC4 correct calculation. For this ECC4 core, it can correct up to 4 errors in a single field. Only when **F3_STAT** equals to 0x01, the value in this field is meaningful. The value could be 0x1 ~ 0x4; it means one error to four errors. The error address will be put at **ECC4F3A1** and **ECC4F3A2**. |

| Bits | Descriptions | |
|---|---|---|
| [17:16] | **F3_STAT** | **ECC4 Status of ECC-Field 3**<br>This field contains the ECC4 correction status of ECC-field 3 (for page size 2048+64B).<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error. |
| [12:10] | **F2_ECNT** | **Error Count of ECC-Field 2**<br>This field contains the error counts after ECC4 correct calculation. For this ECC4 core, it can correct up to 4 errors in a single field. Only when **F2_STAT** equals to 0x01, the value in this field is meaningful. The value could be 0x1 ~ 0x4; it means one error to four errors. The error address will be put at **ECC4F2A1** and **ECC4F2A2**. |
| [9:8] | **F2_STAT** | **ECC4 Status of ECC-Field 2**<br>This field contains the ECC4 correction status of ECC-field 2 (for page size 2048+64B).<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error. |
| [4:2] | **F1_ECNT** | **Error Count of ECC-Field 1**<br>This field contains the error counts after ECC4 correct calculation. For this ECC4 core, it can correct up to 4 errors in a single field. Only when **F1_STAT** equals to 0x01, the value in this field is meaningful. The value could be 0x1 ~ 0x4; it means one error to four errors. The error address will be put at **ECC4F1A1** and **ECC4F1A2**. |
| [1:0] | **F1_STAT** | **ECC4 Status of ECC-Field 1**<br>This field contains the ECC4 correction status of ECC-field 1 (for page size 512+16B and 2048+64B).<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error. |

## ECC4 Field 1 Error Address 1 (ECC4F1A1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4F1A1 | 0xB000_D118 | R | ECC4 Field 1 Error Address 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | F1_ADDR 2 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F1_ADDR2 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | F1_ADDR 1 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F1_ADDR1 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [24:16] | F1_ADDR2 | **ECC4 Error Address 2 of ECC-Field 1**<br>This field contains a 9-bit ECC4 error address 2 of ECC-field 1 (for page size 512+16B and 2048+64B). If it is a correctable error, software can read the error data at ECC4F1D [15:8] for correcting this error. |
| [8:0] | F1_ADDR1 | **ECC4 Error Address 1 of ECC-Field 1**<br>This field contains a 9-bit ECC4 error address 1 of ECC-field 1 (for page size 512+16B and 2048+64B). If it is a correctable error, software can read the error data at ECC4F1D [7:0] for correcting this error. |

## ECC4 Field 1 Error Address 2 (ECC4F1A2)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4F1A2 | 0xB000_D11C | R | ECC4 Field 1 Error Address 2 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | F1_ADDR4 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F1_ADDR4 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | F1_ADDR3 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F1_ADDR3 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [24:16] | F1_ADDR4 | **ECC4 Error Address 4 of ECC-Field 1**<br>This field contains a 9-bit ECC4 error address 4 of ECC-field 1 (for page size 512+16B and 2048+64B). If it is a correctable error, software can read the error data at ECC4F1D [31:24] for correcting this error. |
| [8:0] | F1_ADDR3 | **ECC4 Error Address 3 of ECC-Field 1**<br>This field contains a 9-bit ECC4 error address 3 of ECC-field 1 (for page size 512+16B and 2048+64B). If it is a correctable error, software can read the error data at ECC4F1D [23:16] for correcting this error. |

## ECC4 Field 1 Error Data (ECC4F1D)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4F1D | 0xB000_D120 | R | ECC4 Field 1 Error Data | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| F1_DATA4 | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F1_DATA3 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| F1_DATA2 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F1_DATA1 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:24] | **F1_DATA4** | **ECC4 Error Data 4 of ECC-Field 1**<br>This field contains an 8-bit ECC4 error data 4 of ECC-field 1 (for page size 512+16B and 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F1_ADDR4**; the result will be the correct data. |
| [23:16] | **F1_DATA3** | **ECC4 Error Data 3 of ECC-Field 1**<br>This field contains an 8-bit ECC4 error data 3 of ECC-field 1 (for page size 512+16B and 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F1_ADDR3**; the result will be the correct data. |
| [15:8] | **F1_DATA2** | **ECC4 Error Data 2 of ECC-Field 1**<br>This field contains an 8-bit ECC4 error data 2 of ECC-field 1 (for page size 512+16B and 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F1_ADDR2**; the result will be the correct data. |
| [7:0] | **F1_DATA1** | **ECC4 Error Data 1 of ECC-Field 1**<br>This field contains an 8-bit ECC4 error data 1 of ECC-field 1 (for page size 512+16B and 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F1_ADDR1**; the result will be the correct data. |

### ECC4 Field 2 Error Address 1 (ECC4F2A1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4F2A1 | 0xB000_D124 | R | ECC4 Field 2 Error Address 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | F2_ADDR 2 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F2_ADDR2 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | F2_ADDR 1 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F2_ADDR1 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [24:16] | F2_ADDR2 | **ECC4 Error Address 2 of ECC-Field 2**<br>This field contains a 9-bit ECC4 error address 2 of ECC-field 2 (for page size 2048+64B). If it is a correctable error, software can read the error data at ECC4F2D [15:8] for correcting this error. |
| [8:0] | F2_ADDR1 | **ECC4 Error Address 1 of ECC-Field 2**<br>This field contains a 9-bit ECC4 error address 1 of ECC-field 2 (for page size 2048+64B). If it is a correctable error, software can read the error data at ECC4F2D [7:0] for correcting this error. |

### ECC4 Field 2 Error Address 2 (ECC4F2A2)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4F2A2 | 0xB000_D128 | R | ECC4 Field 2 Error Address 2 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | F2_ADDR4 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F2_ADDR4 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | F2_ADDR3 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F2_ADDR3 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [24:16] | F2_ADDR4 | **ECC4 Error Address 4 of ECC-Field 2**<br>This field contains a 9-bit ECC4 error address 4 of ECC-field 2 (for page size 2048+64B). If it is a correctable error, software can read the error data at ECC4F2D [31:24] for correcting this error. |
| [8:0] | F2_ADDR3 | **ECC4 Error Address 3 of ECC-Field 2**<br>This field contains a 9-bit ECC4 error address 3 of ECC-field 2 (for page size 2048+64B). If it is a correctable error, software can read the error data at ECC4F2D [23:16] for correcting this error. |

### ECC4 Field 2 Error Data (ECC4F2D)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4F2D | 0xB000_D12C | R | ECC4 Field 2 Error Data | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| F2_DATA4 | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F2_DATA3 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| F2_DATA2 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F2_DATA1 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:24] | **F2_DATA4** | **ECC4 Error Data 4 of ECC-Field 2**<br>This field contains an 8-bit ECC4 error data 4 of ECC-field 2 (for page size 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F2_ADDR4**; the result will be the correct data. |
| [23:16] | **F2_DATA3** | **ECC4 Error Data 3 of ECC-Field 2**<br>This field contains an 8-bit ECC4 error data 3 of ECC-field 2 (for page size 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F2_ADDR3**; the result will be the correct data. |
| [15:8] | **F2_DATA2** | **ECC4 Error Data 2 of ECC-Field 2**<br>This field contains an 8-bit ECC4 error data 2 of ECC-field 2 (for page size 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F2_ADDR2**; the result will be the correct data. |
| [7:0] | **F2_DATA1** | **ECC4 Error Data 1 of ECC-Field 2**<br>This field contains an 8-bit ECC4 error data 1 of ECC-field 2 (for page size 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F2_ADDR1**; the result will be the correct data. |

## ECC4 Field 3 Error Address 1 (ECC4F3A1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4F3A1 | 0xB000_D130 | R | ECC4 Field 3 Error Address 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | F3_ADDR2 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F3_ADDR2 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | F3_ADDR1 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F3_ADDR1 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [24:16] | F3_ADDR2 | **ECC4 Error Address 2 of ECC-Field 3**<br>This field contains a 9-bit ECC4 error address 2 of ECC-field 3 (for page size 2048+64B). If it is a correctable error, software can read the error data at ECC4F3D [15:8] for correcting this error. |
| [8:0] | F3_ADDR1 | **ECC4 Error Address 1 of ECC-Field 3**<br>This field contains a 9-bit ECC4 error address 1 of ECC-field 3 (for page size 2048+64B). If it is a correctable error, software can read the error data at ECC4F3D [7:0] for correcting this error. |

## ECC4 Field 3 Error Address 2 (ECC4F3A2)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4F3A2 | 0xB000_D134 | R | ECC4 Field 3 Error Address 2 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | F3_ADDR4 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F3_ADDR4 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | F3_ADDR3 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F3_ADDR3 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [24:16] | F3_ADDR4 | **ECC4 Error Address 4 of ECC-Field 3**<br>This field contains a 9-bit ECC4 error address 4 of ECC-field 3 (for page size 2048+64B). If it is a correctable error, software can read the error data at ECC4F3D [31:24] for correcting this error. |
| [8:0] | F3_ADDR3 | **ECC4 Error Address 3 of ECC-Field 3**<br>This field contains a 9-bit ECC4 error address 3 of ECC-field 3 (for page size 2048+64B). If it is a correctable error, software can read the error data at ECC4F3D [23:16] for correcting this error. |

## ECC4 Field 3 Error Data (ECC4F3D)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4F3D | 0xB000_D138 | R | ECC4 Field 3 Error Data | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| F3_DATA4 | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F3_DATA3 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| F3_DATA2 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F3_DATA1 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:24] | **F3_DATA4** | **ECC4 Error Data 4 of ECC-Field 3**<br>This field contains an 8-bit ECC4 error data 4 of ECC-field 3 (for page size 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F3_ADDR4**; the result will be the correct data. |
| [23:16] | **F3_DATA3** | **ECC4 Error Data 3 of ECC-Field 3**<br>This field contains an 8-bit ECC4 error data 3 of ECC-field 3 (for page size 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F3_ADDR3**; the result will be the correct data. |
| [15:8] | **F3_DATA2** | **ECC4 Error Data 2 of ECC-Field 3**<br>This field contains an 8-bit ECC4 error data 2 of ECC-field 3 (for page size 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F3_ADDR2**; the result will be the correct data. |
| [7:0] | **F3_DATA1** | **ECC4 Error Data 1 of ECC-Field 3**<br>This field contains an 8-bit ECC4 error data 1 of ECC-field 3 (for page size 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F3_ADDR1**; the result will be the correct data. |

## ECC4 Field 4 Error Address 1 (ECC4F4A1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4F4A1 | 0xB000_D13C | R | ECC4 Field 4 Error Address 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | F4_ADDR 2 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F4_ADDR2 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | F4_ADDR 1 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F4_ADDR1 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [24:16] | F4_ADDR2 | **ECC4 Error Address 2 of ECC-Field 4**<br>This field contains a 9-bit ECC4 error address 2 of ECC-field 4 (for page size 2048+64B). If it is a correctable error, software can read the error data at ECC4F4D [15:8] for correcting this error. |
| [8:0] | F4_ADDR1 | **ECC4 Error Address 1 of ECC-Field 4**<br>This field contains a 9-bit ECC4 error address 1 of ECC-field 4 (for page size 2048+64B). If it is a correctable error, software can read the error data at ECC4F4D [7:0] for correcting this error. |

### ECC4 Field 4 Error Address 2 (ECC4F4A2)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4F4A2 | 0xB000_D140 | R | ECC4 Field 4 Error Address 2 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | F4_ADDR4 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F4_ADDR4 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | F4_ADDR3 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F4_ADDR3 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [24:16] | F4_ADDR4 | **ECC4 Error Address 4 of ECC-Field 4**<br>This field contains a 9-bit ECC4 error address 4 of ECC-field 4 (for page size 2048+64B). If it is a correctable error, software can read the error data at ECC4F4D [31:24] for correcting this error. |
| [8:0] | F4_ADDR3 | **ECC4 Error Address 3 of ECC-Field 4**<br>This field contains a 9-bit ECC4 error address 3 of ECC-field 4 (for page size 2048+64B). If it is a correctable error, software can read the error data at ECC4F4D [23:16] for correcting this error. |

### ECC4 Field 4 Error Data (ECC4F4D)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ECC4F4D | 0xB000_D144 | R | ECC4 Field 4 Error Data | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| F4_DATA4 | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| F4_DATA3 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| F4_DATA2 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F4_DATA1 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:24] | **F4_DATA4** | **ECC4 Error Data 4 of ECC-Field 4**<br>This field contains an 8-bit ECC4 error data 4 of ECC-field 4 (for page size 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F4_ADDR4**; the result will be the correct data. |
| [23:16] | **F4_DATA3** | **ECC4 Error Data 3 of ECC-Field 4**<br>This field contains an 8-bit ECC4 error data 3 of ECC-field 4 (for page size 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F4_ADDR3**; the result will be the correct data. |
| [15:8] | **F4_DATA2** | **ECC4 Error Data 2 of ECC-Field 4**<br>This field contains an 8-bit ECC4 error data 2 of ECC-field 4 (for page size 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F4_ADDR2**; the result will be the correct data. |
| [7:0] | **F4_DATA1** | **ECC4 Error Data 1 of ECC-Field 4**<br>This field contains an 8-bit ECC4 error data 1 of ECC-field 4 (for page size 2048+64B). If it is a correctable error, software can read out the error data in this field and doing bitwise XOR with received data locating at address **F4_ADDR1**; the result will be the correct data. |

## 7.10LCD Display Interface Controller (LCM)

The main purpose of Display Controller is used to display the image data to LCD device or connect with external TV-encoder. The input data format of the display controller can be packet YUV422, packet RGB444, packet RGB565, and packet RGB666. The OSD (On Screen Display) function supports packet YUV422 and 8/16 direct-color mode. The LCD controller supports both sync-type and MPU-type LCD Module. This LCD Controller is a bus master and can transfer display data from system memory (SDRAM) without CPU intervention.

### Features

- Input data format
  - YUV422, YUV444
  - RGB444, RGB565, RGB666
- Output format
  - YUV422, YUV444
  - RGB444, RGB565, RGB666
- Display size: Maximum size 1024x600
- Image resize
  - Horizontal up-scaling 1~8X in fractional steps
  - Vertical up-scaling 1~8X in fractional steps
- Convert RGB565, YUV422 display data to RGB444, RGB565, RGB666
- Convert full range YUV to CCIR601
- Windowing support for three OSD graphic or text overlay
- Support CCIR-656 (with header), CCIR-601(with hsync and vsync ) 8/16-bit YUV data output format to connect with external TV encoder
- Support both sync-type and MPU-type LCM (with v-sync or not)
- Support the 8/9/16/18-bit data output to connect with 80/68 series MPU type LCM module
- Convert YUV422, YUV444, RGB565, YUV444, display data to RGB444, RGB565, RGB666, YUV422, YUV444

The LCD Controller includes the following main functions:

- Image post-processing

- Display & overlay control
- Image output control
- Hardware cursor control

### 7.10.1 LCD Controller Function Description

#### 7.10.1.1 VPOST Processor

The Display Engine is used to scale-up the image for display. The image can be arbitrarily up-scaled to full-screen size in horizontal and vertical direction by programming the scaling-factor registers *VA_SCALE*. Similarly, the OSD function also supports up-scaling in horizontal and vertical direction. But it only supports 2X and 4X up-scaling.

#### 7.10.1.2 Display & Overlay Control

The Display Control unit includes timing controller and overlay controller. The timing controller generates the required horizontal and vertical timing for display device. The display timing is defined in the control registers, *CRTC_SIZE~CRTC_VR* and *OSD_WINS~OSD_WINE*. The following figure specifies the registers definition.

#### 7.10.1.3 Digital Display Output Control

Various digital image output modes are supported:

(1) 8-bit/16-bit YUV output for external TV-encoder;

(2) 8-bit RGB output for sync-based TFT-LCD device;

(3) 8-bit/16-bit/18-bit RGB output for high-color sync-based TFT-LCD device;

(4) 8-bit/9-bit/16-bit/18-bit RGB output for MPU-interfaced LCD device.

The display device is defined in register *DEVICE_CTRL* [DEVICE]. The data bus 8-bit/16-bit or 9-bit/18-bit is selected by *DEVICE_CTRL* [DBWORD]. For the MPU-interfaced LCD, 68-series and 80-series MPU interface are supported. The display color formats can be 4096 (RGB444), 65536 (RGB565), and 262144 (RGB666) colors both in 8-bit and 16-bit or 9-bit and 18-bit data bus modes. The related control signals for MPU-interfaced LCD are defined in register *DEVICE_CTRL*. In addition, the image source color format can be YUV or RGB by setting register *DCCS* [VA_SRC].

#### 7.10.1.4 Display Pin Assignment

| Pad name | VD [17:0] | HSYNC | VSYNC | VDEN | VICLK | VOCLK |
|---|---|---|---|---|---|---|
| **Sync mode** | LCD data bus(O) | HSYNC(O) | VSYNC(O) | Data enable(O) | Clock in (I) | Clock out (O) |
| **MPU80** | LCD data bus(I/O) | Write (WR) (O) | Read (RD) (O) | MPU-LCD (RS) (O) | Non used | Chip select(CS) (O) |
| **MPU80+VSync** | LCD data bus(I/O) | Write (WR) (O) | Read (RD) (O) | MPU-LCD (RS) (O) | Vsync (O) | Chip select(CS) (O) |

| | | | | | | |
|---|---|---|---|---|---|---|
| **MPU80+FMARK** | LCD data bus(I/O) | Write (WR) (O) | Read (RD) (O) | MPU-LCD (RS) (O) | FMARK (I) | Chip select(CS) (O) |
| **MPU68** | LCD data bus(I/O) | Read/Write (RW) (O) | Enable (EN) (O) | MPU-LCD (RS) (O) | Non used | Chip select(CS) (O) |
| **MPU68+VSync** | LCD data bus(I/O) | Read/Write (RW) (O) | Enable (EN) (O) | MPU-LCD (RS) (O) | Vsync (O) | Chip select(CS) (O) |
| **MPU68+FMARK** | LCD data bus(I/O) | Read/Write (RW) (O) | Enable (EN) (O) | MPU-LCD (RS) (O) | FMARK (I) | Chip select(CS) (O) |

### 7.10.2 LCD Controller Register Map

**Control and Status Registers**

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **(LCM_BA = 0xB000_8000 )** | | | | |
| **DCCS** | 0xB000_8000 | R/W | Display Controller Control/Status Register | 0x0000_0000 |
| **DEVICE_CTRL** | 0xB000_8004 | R/W | Display Output Device Control Register | 0x0000_00E0 |
| **MPULCD_CMD** | 0xB000_8008 | R/W | MPU-Interface LCD Write Command | 0x0000_0000 |
| **INT_CS** | 0xB000_800C | R/W | Interrupt Control/Status Register | 0x0000_0000 |
| **CRTC_SIZE** | 0xB000_8010 | R/W | CRTC Display Size Control Register | 0x0000_0000 |
| **CRTC_DEND** | 0xB000_8014 | R/W | CRTC Display Enable End | 0x0000_0000 |
| **CRTC_HR** | 0xB000_8018 | R/W | CRTC Internal Horizontal Retrace Control Register | 0x0000_0000 |
| **CRTC_HSYNC** | 0xB000_801C | R/W | CRTC Horizontal Sync Control Register | 0x0000_0000 |
| **CRTC_VR** | 0xB000_8020 | R/W | CRTC Internal Vertical Retrace Control Reg. | 0x0000_0000 |
| **VA_BADDR0** | 0xB000_8024 | R/W | Image Stream Frame Buffer-0 Starting Address. | 0x0000_0000 |
| **VA_BADDR1** | 0xB000_8028 | R/W | Image Stream Frame Buffer-1 Starting Address. | 0x0000_0000 |
| **VA_FBCTRL** | 0xB000_802C | R/W | Image Stream Frame Buffer Control Register | 0x0000_0000 |
| **VA_SCALE** | 0xB000_8030 | R/W | Image Stream Scaling Control Register | 0x0000_0000 |
| **VA_WIN** | 0xB000_8038 | R/W | Image Stream Active Window Coordinates | 0x0001_07FF |
| **VA_STUFF** | 0xB000_803C | R/W | Image Stream Stuff Pixel | 0x0000_0000 |
| **OSD_WINS** | 0xB000_8040 | R/W | OSD Window Starting Coordinates | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **OSD_WINE** | 0xB000_8044 | R/W | OSD Window Ending Coordinates | 0x0000_0000 |
| **OSD_BADDR** | 0xB000_8048 | R/W | OSD Stream Frame Buffer Starting Address | 0x0000_0000 |
| **OSD_FBCTRL** | 0xB000_804C | R/W | OSD Stream Frame Buffer Control Register | 0x0000_0000 |
| **OSD_OVERLAY** | 0xB000_8050 | R/W | OSD Overlay Control Register | 0x0000_0000 |
| **OSD_CKEY** | 0xB000_8054 | R/W | OSD Overlay Color-Key Pattern Register | 0x0000_0000 |
| **OSD_CMASK** | 0xB000_8058 | R/W | OSD Overlay Color-Key Mask Register | 0x0000_0000 |
| **OSD_SKIP1** | 0xB000_805C | R/W | OSD Window Skip1 Register | 0x0000_0000 |
| **OSD_SKIP2** | 0xB000_8060 | R/W | OSD Window Skip2 Register | 0x0000_0000 |
| **OSD_SCALE** | 0xB000_8064 | R/W | OSD horizontal up scaling control register | 0x0000_0000 |
| **MPU_VSYNC** | 0xB000_8068 | R/W | MPU Vsync control register | 0x0000_0000 |
| **HC_CTRL** | 0xB000_806C | R/W | Hardware cursor control Register | 0x0000_0000 |
| **HC_POS** | 0xB000_8070 | R/W | Hardware cursor tip point position on va picture | 0x0000_0000 |
| **HC_WBCTRL** | 0xB000_8074 | R/W | Hardware Cursor Window Buffer Control Register | 0x0000_0000 |
| **HC_BADDR** | 0xB000_8078 | R/W | Hardware cursor memory base address register | 0x0000_0000 |
| **HC_COLOR0** | 0xB000_807C | R/W | Hardware cursor color ram register mapped to bpp = 0 | 0x0000_0000 |
| **HC_COLOR1** | 0xB000_8080 | R/W | Hardware cursor color ram register mapped to bpp = 1 | 0x0000_0000 |
| **HC_COLOR2** | 0xB000_8084 | R/W | Hardware cursor color ram register mapped to bpp = 2 | 0x0000_0000 |
| **HC_COLOR3** | 0xB000_8088 | R/W | Hardware cursor color ram register mapped to bpp = 3 | 0x0000_0000 |

### 7.10.3 LCD Controller Register

## Display Controller Control / Status Register (DCCS)

The register includes Display-Output-Control, Stream-Control and Display-Image-Source-Format-Control registers.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| DCCS | 0xB000_8000 | R/W | Display Controller Control and Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | LACE_F | VSYNC | HACT | VACT | DISP_ON | Reserved |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | OSD_HUP | | OSD_VUP | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ITU_EN | OSD_SRC | | | Reserved | VA_SRC | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SINGLE | FIELD_INTR | CMD_ON | DISP_INT_EN | DISP_OUT_EN | OSD_EN | VA_EN | ENG_RST |

| Bits | Descriptions | |
|---|---|---|
| [29] | **LACE_F** | **Interlace Mode Display Field Status (Read-Only)**<br>0 = Current displayed field is even field<br>1 = Current displayed field is odd field |
| [28] | **VSYNC** | **Internal Vertical Sync Status (Read Only)**<br>When *DEVICE_CTRL*[V_POL] = 1 (high active)<br>0 = Display operation is not within vertical sync period<br>1 = Display operation is within vertical sync period<br>When *DEVICE_CTRL*[V_POL] = 0 (low active)<br>0 = Display operation is within vertical sync period<br>1 = Display operation is not within vertical sync period |

| Bits | Descriptions | |
|---|---|---|
| [27] | **HACT** | **Display Horizontal Line (Read Only)**<br>0 = Display Controller is not operating for horizontal line display<br>1 = Display Controller is operating for horizontal line display |
| [26] | **VACT** | **Display Image Frame (Read Only)**<br>0 = Display Controller is not operating for Image frame display<br>*1 = Display Controller is operating for Image frame display* |
| [25] | **DISP_ON** | **Display Controller Active(Read Only)**<br>0 = Display Controller is active<br>*1 = Display Controller is off* |
| [19:18] | **OSD_HUP** | **OSD Stream Horizontal Up-Scaling**<br>00 = Original<br>01 = 2X<br>10 = Reserved<br>11 = Reserved |
| [17:16] | **OSD_VUP** | **OSD Stream Vertical Up-scaling**<br>00 = 1X<br>01 = 2X<br>10 = 4X<br>11 = Reserved |
| [15] | **ITU_EN** | **ITU656 format header encode**: when DEVICE_CTRL[DEVICE] = 000 and DEVICE_CTRL[DBWORD] = 0<br>0 = Disable<br>1 = Enable |
| [14:12] | **OSD_SRC** | **OSD Stream Source Color Format**<br>000 = YUV422<br>001 = YCBCR422<br>010 = Reserved<br>011 = RGB666<br>100 = RGB565<br>101 = RGB444 Low: {4'h0,R,G,B}<br>111 = RGB444 High: {R,G,B,4'h0}<br>110 = RGB332 |

| Bits | Descriptions | |
|---|---|---|
| [10:8] | **VA_SRC** | **Image Stream Source Color Format**<br>000 = YUV422<br>001 = YCBCR422<br>010 = Reserved<br>011 = RGB666<br>100 = RGB565<br>101 = RGB444 Low: {4'h0,R,G,B}<br>111 = RGB444 High: {R,G,B,4'h0}<br>110 = Reserved |
| [7] | **SINGLE** | **Display Single Frame Mode**<br>0 = Display continuous Image frame<br>1 = Display single picture frame, the Display Controller will stop operating after finishing display one frame. |
| [6] | **FIELD_INTR** | **Interrupt Mode Control**<br>0 = Interrupt signal responses at each frame display complete<br>1 = Interrupt signal responses at each field display complete<br>Note: The setting for this is only meaningful when the display mode is operated at interlaced mode. |
| [5] | **CMD_ON** | **Command Mode**<br>0 = Normal Image display mode<br>1 = Turn-on command mode for sending LCD command or parameter data |
| [4] | **DISP_INT_EN** | **Display controller interrupt output enable**<br>0 = Disable<br>1 = Enable |
| [3] | **DISP_OUT_EN** | **Display-relative Output Pins Tri-state Mode**<br>0 = Output disabled, output pins in tri-state mode (default)<br>1 = Display output enable, normal mode |
| [2] | **OSD_EN** | **OSD Data Fetch Control**<br>0 = Disable<br>1 = Enable |

| Bits | Descriptions | |
|---|---|---|
| [1] | **VA_EN** | **Image**<br>0 = Disable<br>1 = Enable |
| [0] | **ENG_RST** | **Display Engine Reset (except Display Control Registers)**<br>0: Disable, normal operation<br>1: Reset the Display Engine, but the value of the display control registers keep no change |

## Display Device Control Register (DEVICE_CTRL)

The type of external output device is controlled by this register.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| DEVICE_CTRL | 0xB000_8004 | R/W | Display Controller Control and Status Register | 0x0000_00E0 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CMD_LOW | CM16t18 | CMD16 | DE_POL | MCU68 | DBWORD | RGB_SCALE | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LACE | VR_LACE | V_POL | H_POL | FAL_D | LCD_ODD | SEL_ODD | YUV2CCIR |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LCD_DDA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DEVICE | | | RGB_SHIFT | | SWAP_YCbCr | | Reserved |

| Bits | Descriptions | |
|---|---|---|
| [31] | **CMD_LOW** | **Command Low**<br>0 = Output pin RS = 1: command data, 0: display/parameter data<br>1 = Output pin RS = 0: command data, 1: display/parameter data |
| [30] | **CM16t18** | **Command Mapping From 16-bit to 18-bit or 8-bit to 9-bit data bus**<br>Used for 18-bit/9-bit RGB666 MPU-Interfaced LCD device mode.<br>0 = For 18-bit data bus mode,<br>Data[17:0] = {Command[15:8], 1'b0, Command[7:0], 1'b0};<br>For 9-bit data bus mode,<br>Data[8:0] = {Command[7:0], 1'b0};<br>1 = For 18-bit data bus mode,<br>Data[17:0] = {2'b00, Command[15:0]};<br>For 9-bit data bus mode,<br>Data[8:0] = {1'b0, Command[7:0]}; |

| Bits | Descriptions | |
|---|---|---|
| [29] | **CMD16** | **Command Data 16-bit Mode**<br>0 = The command data is 8-bit<br>1 = The command data is 16-bit<br>Note: The 16-bit command mode is only valid when DEVICE_CTRL [DBWORD] is active. |
| [28] | **DE_POL/ IM_262K (Share bit)** | **Active Polarity of Display Output Enable for Sync-Type LCM**<br>0 = Active high<br>1 = Active low<br>Interface Mode Selection for 262K MPU-Interface LCM<br>0 = 9/18 bit data bus<br>1 = 8/16 bit data bus |
| [27] | **MPU68** | **MPU Interface Selection**<br>0 = 80-series MPU interface<br>1 = 68-series MPU interface |

| Bits | Descriptions | |
|---|---|---|
| [26] | **DBWORD** | **Digital LCD Data Bus Width Selection**<br>(Data bus width is equal to WORD length):<br>0 = bus width is equal to half-word, two bus transactions are required for single pixel<br>1 = bus width is equal to word, one bus transaction is required for single pixel<br>For YUV422 output mode:<br>0 = Data bus is 8-bits<br>1 = Data bus is 16-bits<br>For 256/4096/65536 colors mode:<br>0 = Data bus is 8-bits<br>1 = Data bus is 16-bits<br>For 262144 colors mode:<br>0 = Data bus is 8/9-bits<br>1 = Data bus is 16/18-bits<br>For 1677721 colors mode:<br>This bit should be set to 0.<br>0 = Data bus is 8-bits – 3 cycles per pixel at DEVICE[SWAP_YCbCr[1] ] = 0<br>4 cycles per pixel at DEVICE[SWAP_YCbCr[1] ] = 1 |
| [25:24] | **RGB_SCALE** | **RGB Color Type**<br>00 = 4096 colors mode<br>01 = 65536 colors mode<br>10 = 262144 colors mode<br>11 = 16777216 colors mode |
| [23] | **LACE** | **Display Data Output Mode**<br>0 = Non-interlace<br>1 = Interlace |
| [22] | **VR_LACE** | **Sync (Horizontal and Vertical Sync) Interlace**<br>0 = Non-interlace<br>1 = Interlace |

| Bits | Descriptions | |
|---|---|---|
| [21] | **V_POL** | **V_POL (Vertical Polarity)**<br>0 = Low Active<br>1 = High Active |
| [20] | **H_POL** | **H_POL (Horizontal Polarity)**<br>0 = Low Active<br>1 = High Active |
| [19] | **FAL_D** | **FAL_D**<br>0 = Falling Latch Out<br>1 = Rising Latch Out |
| [18:17] | **[LCD_ODD : SEL_ODD]** | **Control LCD Line Data Out**<br>00 = First line data is RGB, second line data is GBR<br>01 = First line data is BGR, second line data is RBG<br>10 = First line data is GBR, second line data is RGB<br>11 = First line data is RBG, second line data is BGR |
| [16] | **YUV2CCIR** | 1 = Convert full range YUV to CCIR601<br>0 = No operation |
| [15:8] | **LCD_DDA** | **Generate LCD Clock Frequency for TFT-LCD Panel**<br>Note: Only for *DEBICE_CTRL*[DEVICE] "100" and "110"<br>Set LCD_DDA = 0 will disable DDA operation. |

| Bits | Descriptions | |
|---|---|---|
| [7:5] | **DEVICE** | **DEVICE Setting**<br>000 = Packed YUV422<br>001 = Packed YUV444<br>100 = Sync-based TFT-LCD (UNIPAC)<br>101 = Sync-based TFT-LCD (SEIKO EPSON)<br>110 = Sync-based High-color TFT-LCD (RGB565/RGB666)<br>111 = MPU-Interfaced LCD (RGB332/RGB444/RGB565/RGB666)(default)<br>Notes:<br>1. Device "000" is supported both in 8-bit and 16-bit data bus.<br>2. Device "110" is supported both in 16-bit and 18-bit. The 16-bit and 18-bit data bus is selected by *DEVICE_CTRL* [RGB_SCALE].<br>3. Device "111" is supported in 8-bit, 9-bit, 16-bit, and 18-bit data bus.<br>4. The 8-bit/16-bit or 9-bit/18-bit data bus is selected by the combination of *DEVICE_CTRL*[DBWORD], *DEVICE_CTRL*[RGB_SCALE] and *DEVICE_CTRL*[DE_POL] |

| Bits | Descriptions | |
|---|---|---|
| [4:3] | **RGB_SHIFT/ DM_262K** (Share bit) | **RGB Data Output Shift for Sync-type LCD Panel**<br>When *DEBICE_CTRL*[DEVICE] = 100, 101<br>00 = Not Shift<br>01 = Shift One Cycle<br>10 = Shift 2 Cycle<br>11 = Not Defined<br><br>RGB Data Output Arrangement for 262K MPU-Interface LCM<br><br>When *DEBICE_CTRL*[DEVICE] = 111, and 16-bit data bus mode (*denote don't care bit)<br>00 = RRRRRRGGGGGGBBBB, **********BB<br>01 = **********RR, RRRRGGGGGGBBBBBB<br>10 = RRRRRR**GGGGGG**, ********BBBBBB**<br>11 = RRRRRR**GGGGGG**, BBBBBB**********<br><br>When *DEBICE_CTRL*[DEVICE] = 111, and 8-bit data bus mode<br>*0 = RRRRRR**, GGGGGG**, BBBBBB**<br>*1 = RRRRRRGG, GGGGBBBB, ******BB<br><br>**ITU656 format select**<br>When DEVICE_CTRL[DEVICE] = 000, DCCS[ITU_EN]==1 and 8-bit data bus mode<br>01 = NTSC<br>10 = PAL |

| Bits | Descriptions | |
|---|---|---|
| [2-1] | **SWAP_YcbCr (share_bit** *DEVICE_CTRL*[DEVICE] = 000**)** | **YUV Data Output Swap** (for Packed YUV mode)<br>When DEVICE_CTRL[DEVICE] = 000<br>00 = UYVY<br>01 = YUYV<br>10 = VYUY<br>11 = YVYU |
| [2] | **SWAP_YcbCr[1](share_bit** DEVICE_CTRL[DEVICE] = 100**)** | **Delay control:** make the cycle of reading data from FIFO to be delay one cycles per two pixel so the output rate is 1.5 cycles per pixel .When DEVICE_CTRL[DEVICE] = 100, 8-bit data bus , **DEVICE_CTRL [SWAP_YcbCr[0]] = 0**, and **DEVICE [DBWORD ]= 1**(Pixel data read from FIFO is 1 cycle per pixel),<br>=> Unipac sub-sampling one component from each RGB pixel<br>0 = 1/3 sub-sampling, per pixel is sub-sampling a component<br>Seq: R0G1B2, R3G4B5… output 1 cycles per pixel<br>For 960x240 panel, (source are expand from 320x240 to 960x240)<br>1= 1/2 sub-sampling, even pixel is sub sampled two components and odd pixel is sub sampled one components<br>Seq: R0G0B1,R2G3B3,R4G4B5….. output 1.5 cycles per pixel<br>For 480x240 (source is 320x240) |
| [1] | **SWAP_YcbCr[0](share_bit** DEVICE_CTRL[DEVICE] = 100**)** | Read cycles per pixel control: When DEVICE_CTRL[DEVICE] = 100, 8-bit data bus, **DEVICE[DBWORD]= 0**, and **DEVICE_CTRL [SWAP_YcbCr[1]]=0**,<br>0 = Pixel data read from FIFO is 2 cycles per pixel,<br>Seq: R0G0B1,R1G2B2,R3G3B,4R4…. 2 cycles per pixel<br>For 640 x240 panel (source 320x240)<br>1= Pixel data read from FIFO is 3 cycles per pixel,<br>Seq: R0G0B0,R1G1B1,R2G2B2,R3G3B3... 3 cycles per pixel<br>For 960x240 panel, (source is 320x240) |

| Bits | Descriptions | |
|---|---|---|
| [2] | **SWAP_YCbCr[1] (share_bit** DEVICE_CTRL[DEVICE] = 110**)** | RGB_Dumy format (for Sync-based High-color TFT-LCD -interfaced LCD)<br>When DEVICE_CTRL[DEVICE] = 110, 8-bit data bus & **16M-color mode**, **DEVICE [DBWORD]= 0**, **SWAP_YCbCr[0] = x**<br>0 = RGB – output 8bit data in the sequence of " R0G0B0R1G1B1….", 3 cycles per pixel<br>1 = RGBX– output 8bit data in the sequence of " R0G0B0XR1G1B1X….", 4 cycles per pixel |
| [1] | **SWAP_YCbCr[0] (share_bit** DEVICE_CTRL[DEVICE] = 110**)** | RGB Data Output Swap (for 65536-color LCD & 262144-color LCD)<br>When **DEVICE_CTRL[DEVICE] = 110**, 8-bit data bus & 65536-color, 9-bit data bus & 256K-color mode and **SWAP_YCbCr[1] = 0**<br>0 = the high-byte of 16-bit RGB565 pixel data is output first, the low-byte data is output secondly; the msb 9-bit of 18-bit RGB666 pixel data is output first, the LSB 9-bit data is output secondly<br>1 = the low-byte of 16-bit RGB565 pixel data is output first, the low-byte data is output secondly;<br>the LSB 9-bit of 18-bit RGB666 pixel data is output first, the msb 9-bit data is output secondly |
| [2] | **SWAP_YCbCr[1] (share_bit** DEVICE_CTRL[DEVICE] = 111**)** | RGB Data Output Swap (for MPU-interfaced LCD)<br>When **DEVICE_CTRL[DEVICE] = 111**, 16-bit data bus & 4096-color mode<br>0 = Data format of DDATA[15:0] is {R,G,B,4'h0} in 4096-color mode<br>1 = Data format of DDATA[15:0] is {4'h0,R,G,B} in 4096-color mode |
| [1] | **SWAP_YCbCr[0] (share_bit** DEVICE_CTRL[DEVICE] = 111**)** | RGB Data Output Swap (for 65536-color LCD & 262144-color LCD)<br>When **DEVICE_CTRL[DEVICE] = 111**, 8-bit data bus & 65536-color, 9-bit data bus & 256K-color mode<br>0 = the high-byte of 16-bit RGB565 pixel data is output first, the low-byte data is output secondly; the msb 9-bit of 18-bit RGB666 pixel data is output first, the LSB 9-bit data is output secondly<br>1 = the low-byte of 16-bit RGB565 pixel data is output first, the low-byte data is output secondly;<br>the LSB 9-bit of 18-bit RGB666 pixel data is output first, the msb 9-bit data is output secondly |

**Data bus arrangement for different pixel and bus for MPU-Interface LCM**

| Gray Scale Selection (RGB_SCALE) | Bus Interface Mode(DBWORD, IM_262K) | SWAP_YCbCr | Data Mode for 262K panels (DM_262K) | Data bus arrangement ("*" denote don't care bit ) | Note |
|---|---|---|---|---|---|
| 12 bits/pixel (00) | 8 bits | ** | ** | **RRRRGGGG**<br>**BBBBRRRR**<br>**GGGGBBBB** | 3xfer/2pixels |
| | 16 bits | 0* | ** | **RRRRGGGGBBBB****** | 1xfer/1pixel |
| | 16 bits | 1* | ** | ******RRRRGGGGBBBB** | 1xfer/1pixel |
| 16 bits/pixel (01) | 8 bits | *0 | ** | **RRRRRGGG**<br>**GGGBBBBB** | 2xfer/1pixel |
| | 8 bits | *1 | ** | **GGGBBBBB**<br>**RRRRRGGG** | 2xfer/1pixel |
| | 16 bits | ** | ** | **RRRRRGGGGGGBBBBB** | 1xfer/1pixel |
| 18 bits/pixel (10) | 8 bits | ** | *0 | **RRRRRR****<br>**GGGGGG****<br>**BBBBBB**** | 3xfer/1pixel |
| | | ** | *1 | **RRRRRRGG**<br>**GGGGBBBB**<br>**********BB** | 3xfer/1pixel |
| | 9 bits | *0 | ** | **RRRRRRGGG**<br>**GGGBBBBBB** | 2xfer/1pixel |
| | 9 bits | *1 | ** | **GGGBBBBBB**<br>**RRRRRRGGG** | 2xfer/1pixel |
| | 16 bits | ** | 00 | **RRRRRRGGGGGGBBBB**<br>**************BB** | 2xfer/1pixel |
| | | ** | 01 | **************RR**<br>**RRRRGGGGGGBBBBBB** | 2xfer/1pixel |
| | | ** | 10 | **RRRRRR**GGGGGG****<br>********BBBBBB** | 2xfer/1pixel |
| | | ** | 11 | **RRRRRR**GGGGGG****<br>**BBBBBB********** | 2xfer/1pixel |
| | 18 bits | | ** | **RRRRRRGGGGGGBBBBBB** | 1xfer/1pixel |

**Data bus arrangement for different pixel for Unipac - Interface LCM at 16 M colors and 8bits data bus**

| Gray Scale Selection | Bus Interface Mode (DBWORD) | SWAP_YCbCr | Control LCD Line Data Out [LCD_ODD : SEL_ODD] | Data bus arrangement ("*" denote don't care bit ) First line(odd line) | Data bus arrangement ("*" denote don't care bit ) second line(even line) | Note |
|---|---|---|---|---|---|---|
| 24 bits/pixel | 1 | 00 | 00 | **RRRRRRRR0 (pixel 0)**<br>**GGGGGGGG1 (pixel 1)**<br>**BBBBBBBB2 (pixel 2)** | **GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB1 (pixel 1)**<br>**RRRRRRRR2 (pixel 2)** | 1xfer/ 1pixel |
| | 1 | 00 | 01 | **BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG1 (pixel 1)**<br>**RRRRRRRR2 (pixel 2)** | **RRRRRRRR0 (pixel 0)**<br>**BBBBBBBB1 (pixel 1)**<br>**GGGGGGGG2 (pixel 2)** | 1xfer/ 1pixel |
| | 1 | 00 | 10 | **GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB1 (pixel 1)**<br>**RRRRRRRR2 (pixel 2)** | **RRRRRRRR0 (pixel 0)**<br>**GGGGGGGG1 (pixel 1)**<br>**BBBBBBBB2 (pixel 2)** | 1xfer/ 1pixel |
| | 1 | 00 | 11 | **RRRRRRRR0 (pixel 0)**<br>**BBBBBBBB1 (pixel 1)**<br>**GGGGGGGG2 (pixel 2)** | **BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG1 (pixel 1)**<br>**RRRRRRRR2 (pixel 2)** | 1xfer/ 1pixel |
| | 1 | 10 | 00 | **RRRRRRRR0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB1 (pixel 2)** | **GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)**<br>**RRRRRRRR1 (pixel 1)** | 1.5xfer / 1pixel |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 10 | 01 | **BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**RRRRRRRR1 (pixel 1)**<br>**BBBBBBBB2 (pixel 2)**<br>**GGGGGGGG2 (pixel 2)**<br>**RRRRRRRR3 (pixel 3)** | **RRRRRRRR0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG1 (pixel 1) RRRRRRRR2 (pixel 2)**<br>**BBBBBBBB2 (pixel 2)**<br>**GGGGGGGG3 (pixel 3)** | 1.5xfer / 1pixel |
| | 1 | 10 | 10 | **GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)**<br>**RRRRRRRR1 (pixel 1)** | **RRRRRRRR0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB1 (pixel 1)** | 1.5xfer / 1pixel |
| | 1 | 10 | 11 | **RRRRRRRR0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG1 (pixel 1)** | **BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**RRRRRRRR1 (pixel 1)** | 1.5xfer / 1pixel |
| | 0 | 00 | 00 | **RRRRRRRR0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB1 (pixel 1)** | **GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)**<br>**RRRRRRRR1 (pixel 1)** | 2xfer/ 1pixel |
| | 0 | 00 | 01 | **BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**RRRRRRRR1 (pixel 1)** | **RRRRRRRR0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG1 (pixel 1)** | 2xfer/ 1pixel |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 0 | 00 | 10 | **GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)**<br>**RRRRRRRR1 (pixel 1)** | **RRRRRRRR0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB1 (pixel 1)** | 2xfer/ 1pixel |
| | 0 | 00 | 11 | **RRRRRRRR0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG1 (pixel 1)** | **BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**RRRRRRRR1 (pixel 1)** | 2xfer/ 1pixel |
| | 0 | 01 | 00 | **RRRRRRRR0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)** | **GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)**<br>**RRRRRRRR0 (pixel 0)** | 3xfer/ 1pixel |
| | 0 | 01 | 01 | **BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**RRRRRRRR0 (pixel 0)** | **RRRRRRRR0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)** | 3xfer/ 1pixel |
| | 0 | 01 | 10 | **GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)**<br>**RRRRRRRR0 (pixel 0)** | **RRRRRRRR0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)** | 3xfer/ 1pixel |
| | 0 | 01 | 11 | **RRRRRRRR0 (pixel 0)**<br>**BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)** | **BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**RRRRRRRR0 (pixel 0)** | 3xfer/ 1pixel |
| | | | | | | |

**Data bus arrangement for different pixel for TFT High colors device - Interface LCM at 16 M colors and 8bits data bus**

| Gray Scale Selection | Bus Interface Mode (DBWORD) | SWAP_YCbCr | Control LCD Line Data Out [LCD_ODD : SEL_ODD] | Data bus arrangement ("*" denote don't care bit ) | | Note |
|---|---|---|---|---|---|---|
| 24 bits data bus | 1 | 00 | ** | **RRRRRRRRGGGGGGGGBBBBBBBB (pixel 0)** | | 1xfer/ 1pixel |
| 8 bits data bus | 0 | 00 | ** | **BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**RRRRRRRR0 (pixel 0)**<br>**BBBBBBBB1 (pixel 1)**<br>**GGGGGGGG1 (pixel 1)**<br>**RRRRRRRR1 (pixel 1)** | | 3xfer/ 1pixel |
| | 0 | 10 | ** | **BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**RRRRRRRR0 (pixel 0)**<br>**********0 (pixel 0)**<br>**BBBBBBBB1 (pixel 1)**<br>**GGGGGGGG1 (pixel 1)**<br>**RRRRRRRR1 (pixel 1)**<br>**********1 (pixel 1)** | | 4xfer/ 1pixel |
| | | | | | | |

**Data bus arrangement for different pixel for TFT High colors device - Interface LCM at 16 M colors and 8bits data bus**

| Gray Scale Selection | Bus Interface Mode (DBWORD) | SWAP_YCbCr | Control LCD Line Data Out [LCD_ODD : SEL_ODD] | Data bus arrangement ("*" denote don't care bit ) | | Note |
|---|---|---|---|---|---|---|
| 8 bits data bus | 0 | 00 | ** | **BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**RRRRRRRR0 (pixel 0)**<br>**BBBBBBBB1 (pixel 1)**<br>**GGGGGGGG1 (pixel 1)**<br>**RRRRRRRR1 (pixel 1)** | | 3xfer/1pixel |
| | 0 | 10 | ** | **BBBBBBBB0 (pixel 0)**<br>**GGGGGGGG0 (pixel 0)**<br>**RRRRRRRR0 (pixel 0)**<br>**********0 (pixel 0)**<br>**BBBBBBBB1 (pixel 1)**<br>**GGGGGGGG1 (pixel 1)**<br>**RRRRRRRR1 (pixel 1)**<br>**********1 (pixel 1)** | | 4xfer/1pixel |
| | | | | | | |

## MPU-Interfaced LCD Write Command Register (MPULCD_CMD)

When DEVICE = 111, a 16-bit value represents MPU-interfaced LCD command/parameter data. For 8-bit data bus or 16-bit data bus with 8-bit command mode, the MPULCD_CMD [15:8] is discarded. When writing data to this register (MPULCD_CMD[7:0]), Display Controller will switch to command mode and write this data to LCM if *DCCS*[CMD_ON] is enabled and Display Controller is not outputting display data. You can read *DCCS* [HACT] and *DCCS* [VACT] to get display status.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MPULCD_CMD | 0xB000_8008 | R/W | MPU-Interface LCD Write Command | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CMD_BUSY | WR_RS | READ | Reserved | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | | | | MPULCD_CMD[17:16] | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| MPULCD_CMD[15:8] | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| MPULCD_CMD[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31] | **CMD_BUSY** | **Command Interface is Busy**<br>0 = Command interface is ready for next command<br>1 = Command interface is busy for writing/reading pending command |
| [30] | **WR_RS** | **Write/Read RS Setting**<br>0 = Output pin RS = 0 when sending command/parameter via MPULCD_CMD<br>1 = Output pin RS = 1 when sending command/parameter via MPULCD_CMD |
| [30] | **READ** | **Read Status or Data**<br>0 = Write command/parameter LCM<br>1 = Read status/data from LCM<br>Note: Data will be stored in MPULCD_CMD[17:0], when CMD_BUSY is inactive after read operation |
| [17:16] | **MPULCD_CMD** | **MPU-interfaced LCD read data (READ ONLY)** |

| | | |
|---|---|---|
| [15:0] | **MPULCD_CMD** | **MPU-interfaced LCD command/parameter data, read data** |

## Interrupt Control/Status Register (INT_CS)

Interrupts are the communication method for Display Controller-initiated communication with the Display Controller Driver. There are several events that may trigger an interrupt from the Display Controller. Each specific event sets a specific bit in the *INT_CS* register. The Display Controller requests an interrupt when all three of the following conditions are met:

- The **DISP_INT_EN** bit in *DCCS* is set to '1'.
- A status bit in *INT_CS* is set to '1'.
- The corresponding enable bit in *INT_CS* for the *Status* bit is set to '1'.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| INT_CS | 0xB000_800C | R/W | Interrupt Control/Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DISP_F_INT | DISP_F_STATUS | UNDERRUN_INT | BUS_ERROR_INT | Reserved | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | UNDERRUN_EN | DISP_F_EN |

| Bits | Descriptions | |
|---|---|---|
| [31] | **DISP_F_INT** | **Frame Display Complete Interrupt Status, (Write-Clear)**<br>When write "1" value on the bit, the interrupt will be cleared.<br>(This status bit can be internally written no matter DISP_F_EN is enable or not) |
| [30] | **DISP_F_STATUS** | **Frame Display Complete Internal Status (2), (Write-Clear)**<br>When write "1" value on the bit, it will be cleared.<br>(This status bit can be internally written only if DISP_F_EN is enable)<br>Note: The interrupt status can be programmed for indicating frame displayed complete or field displayed complete by *DCCS* [FIELD_INTR]. |

| Bits | Descriptions | |
|---|---|---|
| [29] | **UNDERRUN_INT** | **FIFO under-run Interrupt Status (Write-Clear)**<br>When write "1" value on the bit, the interrupt will be cleared. |
| [28] | **BUS_ERROR_INT** | **Bus Error Interrupt (Write-Clear)**<br>When DMA bus master receive an error response from slaves, this bit will be set.<br>When write "1" value on the bit, the interrupt will be cleared.<br>Note: This interrupt is always enabled |
| [1] | **UNDERRUN_EN** | **FIFO under-run Interrupt Enable**<br>0 = Disable<br>1 = Enable |
| [0] | **DISP_F_EN** | **Frame Display Complete Interrupt Enable**<br>0 = Disable<br>1 = Enable |

## CRTC Display Size Register (CRTC_SIZE)

This register controls the display size. It includes Horizontal-Total (HTT) and Vertical-Total (VTT) registers. The value of HTT specifies the total number of pixels in the CRTC horizontal scan line interval including retrace time. And the value of VTT specifies the total number of scan line for each field (frame), including the retrace time.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRTC_SIZE | 0xB000_8010 | R/W | CRTC Display Size | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | VTT[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VTT[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | HTT[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HTT[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **VTT[10:0]** | **CRTC Vertical Total Scan Lines**<br>An 11-bits value specifies the total number of scan line for each field, including the retrace time |
| [10:0] | **HTT[10:0]** | **CRTC Horizontal Total Pixels**<br>*An 11-bits value specifies the total number of pixels in the CRTC horizontal scan line interval including the retrace time* |

## CRTC Display Enable End Register (CRTC_DEND)

This register controls the actual display size of the output device. It includes HDEND and VDEND registers. The value of HDEND specifies the total number of displayed pixels for a scan line. And the value of VDEND specifies the total number of displayed scan line for each field (frame).

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRTC_DEND | 0xB000_8014 | R/W | CRTC Display Enable End | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | VDEND[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VDEND[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | HDEND[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HDEND[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **VDEND[10:0]** | **CRTC Vertical Display Enable End**<br>An 11-bits value specifies the total number of displayed scan line for each field. |
| [10:0] | **HDND[10:0]** | **CRTC Horizontal Display Enable End**<br>An 11-bits value specifies the total number of displayed pixels for scan line. |

## CRTC Internal Horizontal Retrace Timing Register (CRTC_HR)

The internal horizontal retrace timing can be controlled by properly setting the values of retrace starting (HRS) and ending (HRE) registers included in this register. The values are programmed in number of pixels.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRTC_HR | 0xB000_8018 | R/W | CRTC Internal Horizontal Retrace Timing | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | HRE[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HRE[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | HRS[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HRS[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **HRE[10:0]** | **CRTC Internal Horizontal Retrace End Low**<br>An 11-bits value programmed in pixels, at which the Internal Horizontal Retrace becomes inactive |
| [10:0] | **HRS[10:0]** | **CRTC Internal Horizontal Retrace Start Timing**<br>An 11-bits value programmed in pixels, at which the Internal Horizontal Retrace becomes active |

## CRTC Horizontal Sync Timing Register (CRTC_HSYNC)

The horizontal sync timing can be controlled by properly setting the values of starting (HSYNC_S) and ending (HSYNC_E) registers included in this register. The values are programmed in numbers of pixel.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRTC_HSYNC | 0xB000_801C | R/W | CRTC Horizontal Sync Timing | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| HSYNC_SHIFT [1:0] | | Reserved | | | HSYNC_E[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HSYNC_E[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | HSYNC_S[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HSYNC_S[7:0] | | | | | | | |

| Bits | Descriptions |
|---|---|

| Bits | Descriptions | |
|---|---|---|
| [31-30] | **HSYNC_SHIFT [1:0]** | **Hsync signal adjustment for multi-cycles per pixel mode of Sync-based Unipac-LCD**<br>When DEVICE_CTRL[DEVICE] = 100,8-bit data bus, DEVICE[DBWORD]= 0, and DEVICE_CTRL[SWAP_YcbCr[1]]=0,<br>If DEVICE_CTRL[SWAP_YcbCr[0]]= 0, it means that 2 cycles per pixel, so hsync's hrs would move three cycles if hrs added or subtracted 1.In order to adjust hsync signal in pclk unit, RGB_SHIFT shared register aids this function in -LCD with 8 bits data bus mode.<br>HSYNC_SHIFT = 0 : hsync will not move<br>HSYNC _SHIFT = 1 : hsync will left move 1 pclk cycle<br>If DEVICE_CTRL[SWAP_YcbCr[0]] = 1, it means that 3 cycles per pixel, so hsync's hrs would move three cycles if hrs added or subtracted 1.In order to adjust hsync signal in pclk unit, RGB_SHIFT shared register aids this function in High-color TFT-LCD with 8 bits data bus mode.<br>RGB_SHIFT = 0 : hsync will not move<br>HSYNC _SHIFT = 1 : hsync will left move 1 pclk cycle<br>HSYNC _SHIFT = 2 : hsync will left move 2 pclk cycle<br><br>**Hsync signal adjustment for multi-cycles per pixel mode of Sync-based High-color TFT-LCD**<br>When DEVICE_CTRL[DEVICE] = 110 , DEVICE_CTRL[RGB_SCALE]=3(16M-color mode) and DEVICE[DBWORD]= 0,<br>If DEVICE_CTRL[SWAP_YcbCr[1]] = 0, it means that 3 cycles per pixel, so hsync's hrs would move three cycles if hrs added or subtracted 1.In order to adjust hsync signal in pclk unit, RGB_SHIFT shared register aids this function in High-color TFT-LCD with 8 bits data bus mode.<br>HSYNC _SHIFT = 0 : hsync will not move<br>HSYNC _SHIFT = 1 : hsync will left move 1 pclk cycle<br>HSYNC _SHIFT = 2 : hsync will left move 2 pclk cycle<br>If DEVICE_CTRL[SWAP_YcbCr[1]] = 1, it means that 4 cycles per pixel, so hsync's hrs would move three cycles if hrs added or subtracted 1.In order to adjust hsync signal in pclk unit, RGB_SHIFT shared register aids this function in High-color TFT-LCD with 8 bits data bus mode.<br>HSYNC _SHIFT = 0 : hsync will not move<br>HSYNC _SHIFT = 1 : hsync will left move 1 pclk cycle<br>HSYNC _SHIFT = 2 : hsync will left move 2 pclk cycle<br>HSYNC _SHIFT = 3 : hsync will left move 3 pclk cycle |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **HSYNC_E[10:0]** | **CRTC Horizontal Sync End Timing**<br>An 11-bits value programmed in pixels, at which the Horizontal Sync Signal becomes inactive |
| [10:0] | **HSYNC_S[10:0]** | **CRTC Horizontal Sync Start Timing**<br>An 11-bits value programmed in pixels, at which the Horizontal Sync signal becomes active |

## CRTC Internal Vertical Retrace Timing Register (CRTC_VR)

The vertical retrace timing can be controlled by properly setting the values of starting (VRS) and ending (VRE) registers included in this register. The values are programmed in numbers of scan-line.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRTC_VR | 0xB000_8020 | R/W | CRTC Internal Vertical Retrace Timing | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | VRE[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VRE[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | VRS[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VRS[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **VRE[10:0]** | **CRTC Vertical Internal Retrace End Low**<br>An 11-bits value is programmed in number of scan line, at which the internal vertical retrace becomes inactive. |
| [10:0] | **VRS[10:0]** | **CRTC Vertical Internal Retrace Start Timing**<br>An 11-bits value is programmed in number of scan line, at which the internal vertical retrace becomes active |

## Image Stream Frame Buffer-0 Starting Address (VA_BADDR0)

The value of this register represents the starting memory address of the frame buffer-0 for Image data stream.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| VA_BADDR0 | 0xB000_8024 | R/W | Image Stream Frame Buffer-0 Starting Address | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| VA_BADDR0[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VA_BADDR0[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| VA_BADDR0[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VA_BADDR0[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **VA_BADDR0[31:0]** | Starting memory address of the frame buffer-0 for Image data stream<br>The value of this register represents the starting memory address of the frame buffer-0 for Image data stream. |

## Image Stream Frame Buffer-1 Starting Address (VA_BADDR1)

The value of this register represents the starting memory address of the frame buffer-1 for Image data stream.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| VA_BADDR1 | 0xB000_8028 | R/W | Image Stream Frame Buffer-1 Starting Address | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| VA_BADDR1[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VA_BADDR1[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| VA_BADDR1[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VA_BADDR1[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **VA_BADDR1[31:0]** | Starting memory address of the frame buffer-1 for Image data stream<br>The value of this register represents the starting memory address of the frame buffer-1 for Image data stream. |

## Image Stream Frame Buffer Control Register (VA_FBCTRL)

The information contained in this register is used to efficiently control the frame buffer operation. The VA_STRIDE shows the word offset of memory address between two vertically adjacent lines. The VA_FF specifies the number of WORD SDRAM access cycle for a horizontal scan line fetching. The data buffer FIFO is divided into two or four regions depending on the value of the IO_REGION_HALF. If IO_REGION_HALF is not asserted, there are four regions of 8 words each. If IO_REGION_HALF is asserted, there are two regions of 16 words each. The size of the region affects the AHB burst transfer size. There are two pointers into the FIFO: one for the display engine data and one for the AHB data. These pointers are maintained in the Data Buffer Control module. When the pointers are not in the same region, an AHB burst cycle is issued to read or write the data in the region pointed to by the AHB pointer.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| VA_FBCTRL | 0xB000_802C | R/W | Image Stream Frame Buffer Control | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DB_EN | START_BUF | FIELD_DUAL | IO_REGION_HALF | Reserved | VA_FF[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VA_FF[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | VA_STRIDE[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VA_STRIDE[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31] | **DB_EN** | **Dual Buffer Switch Enable**<br>0 = Dual buffer switch disable, Always fetch data from address which START_BUF indicated<br>1 = Dual buffer switch enable. Switch starting address between VA_BADDR0 and VA_BADDR1 at each frame/field starts (controlled by FIELD_DUAL). The first display frame/field address is controlled by START_BUF. |
| [30] | **START_BUF** | **Starting Buffer of Dual-buffer**<br>0 = Starting fetch data from VA_BADDR0<br>1 = Starting fetch data from VA_BADDR1 |

| Bits | Descriptions | |
|---|---|---|
| [29] | **FIELD_DUAL** | **Dual-buffer Switch Control**<br>0 = Switch dual-buffer before each frame starts<br>1 = Switch dual-buffer before each field starts |
| [28] | **IO_REGION_HALF** | **Data Buffer Region Size**<br>0 = 8 words/region<br>1 = 16 words/region<br>Note: Both VA and OSD FIFO are controlled by this bit. |
| [26:16] | **VA_FF[10:0]** | **Image Stream Fetch Finish**<br>An 11-bits value specifies the number of WORD SDRAM access cycle for a horizontal scan line fetching of Image data stream. |
| [10:0] | **VA_STRIDE[10:0]** | **Image Stream Frame Buffer Stride**<br>An 11-bits value specifies the word offset of memory address of vertically adjacent line for Image stream. |

## Image Stream Scaling Control Register (VA_SCALE)

This register control the Image up-scaling factors, both horizontal and vertical up-scaling ratios are ranging from 1.0 to 7.99999 in fractional steps. There are two modes of horizontal up-scaling, interpolation and duplication, which can be controlled by setting XCOPY.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| VA_SCALE | 0xB000_8030 | R/W | CRTC Display Size | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | VA_SCALE_V[12:8] | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VA_SCALE_V[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | Reserved | | VA_SCALE_H[12:8] | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VA_SCALE_H[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [28:16] | **VA_SCALE_V[12:0]** | **Image Vertical Scaling Control**<br>A 13-bits value specifies the vertical scaling factor of 1.0~7.99999. Bits 12-10 specify the integral part and bits 9-0 specifies the decimal part of the scaling factor. |
| [15] | **XCOPY** | **Image Stream Horizontal Up-scaling Mode**<br>0 = Interpolation<br>1 = Duplication |
| [12:0] | **VA_SCALE_H[12:0]** | **Image Horizontal Scaling Control**<br>A 13-bits value specifies the horizontal scaling factor of 1.0~7.99999. Bit 12-10 specifies the integral part and bits 9-0 specifies the decimal part of the scaling factor. |

## Image Stream Active Window Coordinates (VA_WIN)

This pair of registers (VA_WYS, VA_WYE) specifies the area which Image stream will occupy in the screen. It is called Active window for Vide Stream. The pixels outside the active window will be filled with the color specified by *VA_STUFF*. When the value of VA_WYE is greater than *CRTC_DEND* [VDEN*]*, the Active window will be actually ended at *CRTC_DEND* [VDEN],

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| VA_WIN | 0xB000_8038 | R/W | Image Stream Active Window Coordinates | 0x0001_07FF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | VA_WYS[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VA_WYS[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | VA_WYE [10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VA_WYE[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **VA_WYS[10:0]** | **VA Active Window Y-Start**<br>An 11-bit s value specifies the vertical starting scan line of the Active VA window |
| [10:0] | **VA_WYE[10:0]** | **VA Active Window Y-End**<br>An 11-bit value specifies the last vertical scan line of the Active VA window. |

## Image Stream Stuff Register (VA_STUFF)

A 24-bit value specifies stuff pattern for non-active window area in Image Stream.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| VA_STUFF | 0xB000_8054 | R/W | Image Stream Stuff Pixel for non-active area | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VA_STUFF[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| VA_STUFF [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VA_STUFF [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:16] | **VA_STUFF [23:16]** | The 8 higher-order bits are used for Y or R component according to the source color format |
| [15:8] | **VA_STUFF [15:8]** | The 8 middle-order bits are used for U or G component according to the source color format |
| [7:0] | **VA_STUFF [7:0]** | The 8 lower-order bits are used for V or B component according to the source color format |

## OSD Window Starting Coordinates Register (OSD_WINS)

The starting coordinates of the OSD window is specified in this register. Two values form the coordinates; they are the horizontal starting pixel (OSD_WXS) and the vertical starting scan line (OSD_WYS).

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| OSD_WINS | 0xB000_8040 | R/W | OSD Window Starting Coordinates Timing | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | OSD_WYS[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| OSD_WYS[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | OSD_WXS [10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSD_WXS[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **OSD_WYS** | **OSD Window Y-Start**<br>An 11-bit s value specifies the vertical starting scan line of the OSD window |
| [10:0] | **OSD_WXS** | **OSD Window X-Start**<br>An 11-bits value specifies the horizontal starting pixel position of the OSD window. |

## OSD Window Ending Coordinates Register (OSD_WINE)

The ending coordinates of the OSD window is specified in this register. Two values form the coordinates; they are the last horizontal pixel (OSD_WXE) and the last vertical scan-line (OSD_WYS).

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| OSD_WINE | 0xB000_8044 | R/W | OSD Window Ending Coordinates Timing | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | OSD_WYE[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| OSD_WYE[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | OSD_WXE [10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSD_WXE[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **OSD_WYE** | **OSD Window Y-End**<br>An 11-bit value specifies the last vertical scan line of the OSD window. |
| [10:0] | **OSD_WXE** | **OSD Window X-End**<br>An 11-bits value specifies the last horizontal pixel position of the OSD window. |

## OSD Stream Frame Buffer Starting Address (OSD_BADDR)

The value of this register represents the starting memory address of the frame buffer for OSD data stream.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| OSD_BADDR | 0xB000_8048 | R/W | OSD Stream Frame Buffer Starting Address | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| OSD_BADDR [31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| OSD _BADDR [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| OSD _BADDR [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSD _BADDR [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **OSD_BADDR** | Starting memory address of the frame buffer for OSD data stream<br>The value of this register represents the starting memory address of the frame buffer for OSD data stream. |

## OSD Stream Frame Buffer Control Register (OSD_FBCTRL)

The information contained in this register is used to efficiently control the frame buffer operation. The OSD_STRIDE shows the word offset of memory address between two vertically adjacent lines. The OSD_FF specifies the number of WORD SDRAM access cycle for a horizontal scan line fetching.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| OSD_FBCTRL | 0xB000_804C | R/W | OSD Stream Frame Buffer Control | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | OSD_FF[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| OSD_FF[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | OSD _STRIDE[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSD _STRIDE[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **OSD_FF** | **OSD Stream Fetch Finish**<br>An 11-bits value specifies the number of WORD SDRAM access cycle for a horizontal scan line fetching of OSD data stream. |
| [10:0] | **OSD_STRIDE** | **OSD Stream Frame Buffer Stride**<br>An 11-bits value specifies the word offset of memory address of vertically adjacent line for OSD stream. |

## OSD Overlay Control Register (OSD_OVERLAY)

Setting this register can control the display effect of the overlay area. It can be periodic blanking, VA and OSD data mixing, and VA or OSD data alone.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| OSD_OVERLAY | 0xB000_8050 | R/W | OSD Overlay Control | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BLINK_VCNT[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | BLI_ON | CKEY_ON |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | VA_SYNW[2:0] | | | OCR1[1:0] | | OCR0[1:0] | |

| Bits | Descriptions | |
|---|---|---|
| [23:16] | **BLINK_VCNT** | **OSD Blinking Cycle Time.**<br>An 8 bits value specifies the OSD blinking cycle time (unit: Vsync). |
| [9] | **BLI_ON** | **OSD Blinking Control**<br>0 = Blinking Disable<br>1 = Blinking Enable<br>Note: Blinking control mode share the same color-key pattern registers with color-key control mode |
| [8] | **CKEY_ON** | **OSD Color-Key Control**<br>0 = Color-Key Disable<br>1 = Color-Key Enable |
| [6:4] | **VA_SYNW** | **Synthesis Image Weighting**<br>000 = Synthesized Image = Image; otherwise,<br>Synthesized Image=((Image × VA_SYNW)+(OSD × (8-VA_SYNW))) / 8 |

| Bits | Descriptions | |
|---|---|---|
| [3:2] | **OCR1** | **Image/OSD Overlay Control 1**<br>When (DCCS[WIN_EN:OSD_EN]==2'b11), (OSD_OVERLAY[CKEY_ON]==1), Display region within OSD window, color-key condition match,<br>00 = Display Image data<br>01 = Display OSD data<br>10 = Display synthesized (Image + OSD) data<br>11 = Reserved |
| [1:0] | **OCR0** | **Image/OSD Overlay Control 0**<br>When (DCCS[WIN_EN:OSD_EN]==2'b11), (OSD_OVERLAY[CKEY_ON]==1), Display region within OSD window, color-key condition un-match,<br>00 = Display Image data<br>01 = Display OSD data<br>10 = Display synthesized (Image + OSD) data<br>11 = Reserved |

## OSD Overlay Color-Key Pattern Register (OSD_CKEY)

A 24-bit value specifies OSD color-key pattern. When 24-bit OSD data is equal to the specified pattern data, the color-key condition is matched.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| OSD_CKEY | 0xB000_8054 | R/W | OSD Overlay Color-Key Pattern | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| OSD_CKEY[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| OSD_CKEY[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSD_CKEY[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:16] | **OSD_CKEY** | The 8 higher-order bits are used for OSD data comparing of Y or R component according to the source color format |
| [15:8] | **OSD_CKEY** | The 8 middle-order bits are used for OSD data comparing of U or G component according to the source color format |
| [7:0] | **OSD_CKEY** | The 8 lower-order bits are used for OSD data comparing of V or B component according to the source color format |

## OSD Overlay Color-Key Mask Register (OSD_CMASK)

A 24-bit value serves as the mask of OSD color-key pattern comparing. The OSD data only compare with the color-key pattern where the mask bits are set as 1.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| OSD_CKEY | 0xB000_8058 | R/W | OSD Overlay Color-Key Mask | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| OSD_MASK[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| OSD_ MASK [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSD_ MASK [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:16] | **OSD_CMASK** | The 8 higher-order bits are used for pattern mask of Y or R component according to the source color format |
| [15:8] | **OSD_CMASK** | The 8 middle-order bits are used for pattern mask of U or G component according to the source color format |
| [7:0] | **OSD_CMASK** | The 8 lower-order bits are used for pattern mask of V or B component according to the source color format |

## OSD Window Skip1 Register (OSD_SKIP1)

This register is used to separate OSD into two sub-windows. The OSD-data fetching of lines enclosed by this skip region in OSD window is skipped. The value of starting-Y address (OSK_SKIP1_YS) can't be the same with OSD_WYS. The minimum value of it is OSD_WYS+1. In interlace mode, the minimum value of ending address (OSK_SKIP1_YE) is OSD_SKIP1_YS +1.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| OSD_SKIP1 | 0xB000_805C | R/W | OSD Window SKIP1 Y address | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | OSD_SKIP1_YS[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| OSD_SKIP1_YS[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | OSD_SKIP1_YE[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSD_SKIP1_YE[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **OSD_SKIP1_YS** | **OSD Window Skip1 Y-Start**<br>An 11-bit value specifies the first vertical scan line of the OSD skip1 window. |
| [10:0] | **OSD_SKIP1_YE** | **OSD Window X-End**<br>An 11-bits value specifies the last vertical scan line of the OSD skip1 window. |

## OSD Window SKIP2 Register (OSD_SKIP2)

This register is used to separate OSD into two sub-windows. The OSD-data fetching of lines enclosed by this skip region in OSD window is skipped. The value of starting-Y address (OSK_SKIP2_YS) can't be the same with OSD_WYS. The minimum value of it is OSD_WYS+1. In interlace mode, the minimum value of ending address (OSK_SKIP2_YE) is OSD_SKIP2_YS +1.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| OSD_SKIP2 | 0xB000_8060 | R/W | OSD Window SKIP2 Y address | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | OSD_SKIP2_YS[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| OSD_SKIP2_YS[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | OSD_SKIP2_YE[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSD_SKIP2_YE[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **OSD_SKIP2_YS** | **OSD Window SKIP2 Y-Start**<br>An 11-bit value specifies the first vertical scan line of the OSD SKIP2 window. |
| [10:0] | **OSD_SKIP2_YE** | **OSD Window X-End**<br>An 11-bits value specifies the last vertical scan line of the OSD SKIP2 window. |

## OSD Scaling Control Register (OSD_SCALE)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| OSD_SCALE | 0xB000_8064 | R/W | OSD Horizontal Up-Scaling Factor | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | Reserved | | OSD_SCALE_H[12:8] | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSD_SCALE_H[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [12:0] | **OSD_SCALE_H** | **OSD Horizontal Up-Scaling Factor**<br>This register control the OSD up-scaling factor, the horizontal up-scaling ratios are ranging from 1.0 to 7.99999 in fractional steps. There is only one mode of horizontal up-scaling by duplication. |

## MPU Vsync Control Register (MPU_VSYNC)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| MPU_VSYNC | 0xB000_8068 | R/W | MPU Vsync control register | 0x0000_0000 |

This register controls the MPU Vsync output pin.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | MPU_VSYNC_WIDTH | | | | MPU_VSYNC_POL | MPU_FMARK | MPU_V_EN |

| Bits | Descriptions | |
|---|---|---|
| [6:3] | **MPU_VSYNC_WIDTH** | **MPU Vsync pulse width:**<br>1 ~15 scanning line<br>1 – 15 : 1 (default) ~ 15 |
| [2] | **MPU_VSYNC_POL** | **MPU_Vsync polarity , when MPU_VSYNC [MPU_FMARK] = 0**<br>0 = Low Active (default)<br>1 = High Active<br>**FMARK_POL (FMARK Polarity), when MPU_VSYNC[MPU_FMARK] = 1**<br>0 = Low Active<br>1 = High Active |
| [1] | **MPU_FMARK** | **MPU FMARK mode:**<br>0: vsync output – output enable will be high, vsync output from VPOST. (default)<br>1: vsync input – output enable will be low, VPOST receive FMARK(vsync) from mpu device |
| [0] | **MPU_V_EN** | **MPU Vsync functional enable**<br>0: disable (default)<br>1: enable |

## Hardware Cursor Control Register (HC_CTRL)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_CTRL | 0xB000_806C | R/W | Hardware cursor control register | 0x0000_0000 |

This register is used to control the modes of hardware cursor. (HC_TIP_X, HC_TIP_Y) specifies which the cursor's tip is located at on hardware cursor block.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | HC_TIP_Y [5:0] | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | HC_TIP_X [5:0] | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | HC_MODE[2:0] | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **HC_TIP_Y** | **Y position of Hardware cursor picture's tip on hardware cursor bit map** |
| [10:0] | **HC_TIP_X** | **X position of Hardware cursor picture's tip on hardware cursor bit map.** |
| [2:0] | **HC_MODE** | **Hardware Cursor Mode setting:**<br>0: 32x32x2bpp – 4 color mode<br>1: 32x32x2bpp – 3 color mode and transparency mode<br>2: 64x64x2bpp – 4 color mode<br>3: 64x64x2bpp – 3 color mode and transparency mode<br>4: 128x128x1bpp – 2 color mode<br>5: 128x128x1bpp – 1 color mode and transparency mode |

## HC POSITION Register (HC_POS)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_POSITION | 0xB000_8070 | R/W | Hardware cursor tip position control register | 0x0000_0000 |

This register is used to control the position of hardware cursor coordinate on va picture. (HC_X, HC_Y) can be changed dynamically at by software setting.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | HC_Y[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HC_Y[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | HC_X[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HC_X [7:0] | | | | | | | |

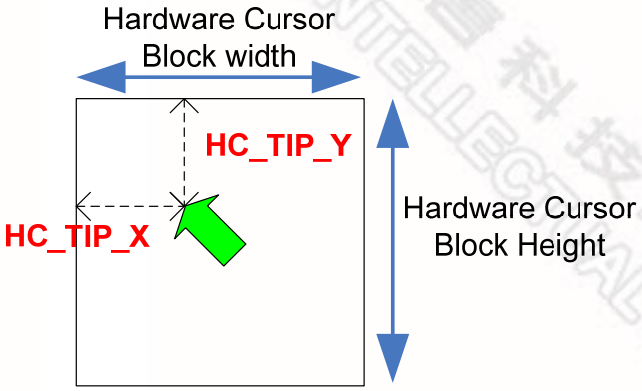
| Bits | Descriptions | |
|---|---|---|
| [31:27] | **Reserve** | |
| [26:16] | **HC_Y** | Y position of hardware cursor's tip on va picture |
| [15:11] | **Reserve** | |
| [10:0] | **HC_X** | X position of hardware cursor's tip on va picture |

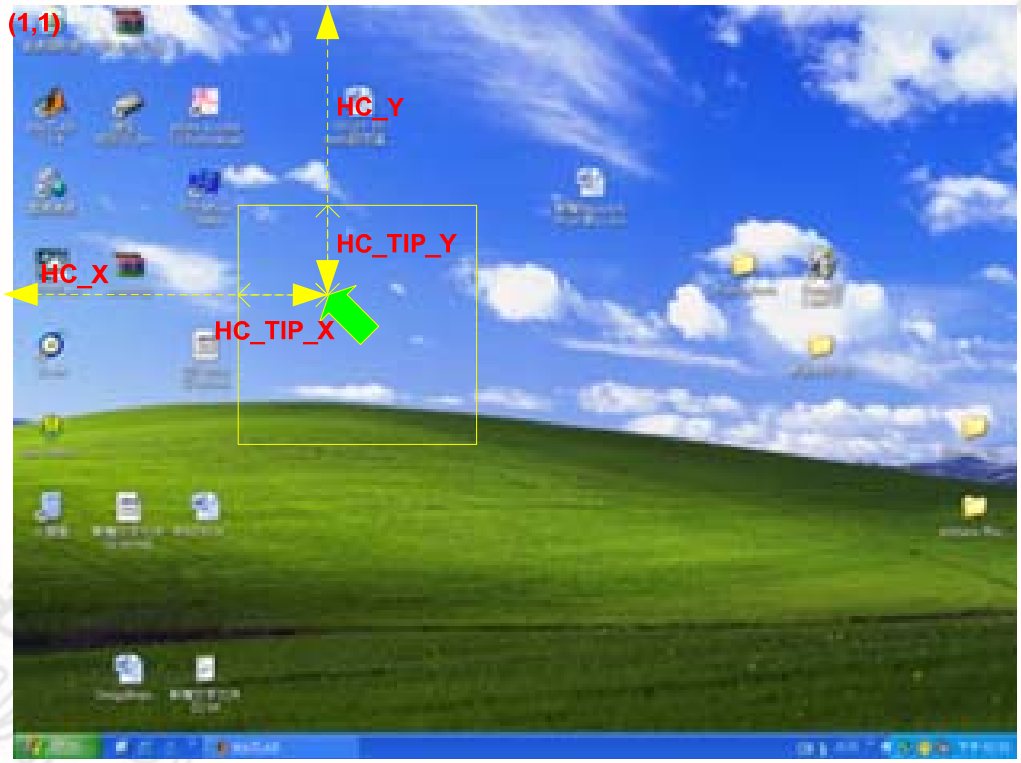
Hardware Cursor block width and height depend on HC_CTRL [HC_MODE] setting.

HC_CTRL [HC_MODE] = 0, 1 => hardware cursor block width = hardware cursor block height= 32

HC_CTRL [HC_MODE] = 2, 3 => hardware cursor block width = hardware cursor block height= 64

HC_CTRL [HC_MODE] = 4, 5 => hardware cursor block width = hardware cursor block height= 128


Hardware Cursor
Block width
HC_TIP_Y
HC_TIP_X
Hardware Cursor
Block Height


(1,1)
HC_Y
HC_TIP_Y
HC_X
HC_TIP_X

## Hardware Cursor Window Buffer Control Register (HC_WBCTRL)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_WBCTRL | 0xB000_8074 | R/W | Hardware Cursor Window Buffer Control | 0x0000_0000 |

The information contained in this register is used to efficiently control the hardware cursor window buffer operation. The HC_STRIDE shows the word offset of memory address between two vertically adjacent lines. The VA_FF specifies the number of WORD SDRAM access cycle for a horizontal scan line fetching.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | HC_FF[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HC_FF[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | HC_STRIDE[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HC_STRIDE[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **HC_FF** | **Hardware cursor Fetch Finish**<br>A 11-bits value specifies the number of WORD SDRAM access cycle for a horizontal scan line fetching of Hardware cursor window |
| [10:0] | **HC_STRIDE** | **Hardware cursor Window Buffer Stride**<br>A 11-bits value specifies the word offset of memory address of vertically adjacent line for Hardware cursor window |

## HC BADDR Register (HC_BADDR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_BADDR | 0xB000_8078 | R/W | Hardware cursor memory base address register | 0x0000_0000 |

This register is used to control the starting memory address of the frame buffer for Hardware cursor data stream.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| HC_BADDR[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HC_BADDR[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HC_BADDR[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HC_BADDR[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **HC_BADDR** | Starting memory address of the frame buffer for Hardware cursor data stream<br>The value of this register represents the starting memory address of the frame buffer |

## HC Color RAM 0 Register (HC_COLOR0)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_COLOR0 | 0xB000_807C | R/W | Hardware cursor color ram register mapped to bpp = 0 | 0x0000_0000 |

This register is used to control the color of hardware cursor according to bpp value 0.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HC_COLOR0_R | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HC_COLOR0_G | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HC_COLOR0_B | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:16] | **HC_COLOR0_R** | Hardware cursor color 0 R |
| [15:8] | **HC_COLOR0_G** | Hardware cursor color 0 G |
| [7:0] | **HC_COLOR0_B** | Hardware cursor color 0 B |

### HC Color RAM 1 Register (HC_COLOR1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_COLOR1 | 0xB000_8080 | R/W | Hardware cursor color ram register mapped to bpp = 1 | 0x0000_0000 |

This register is used to control the color of hardware cursor according to bpp value 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HC_COLOR1_R | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HC_COLOR1_G | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HC_COLOR1_B | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:16] | **HC_COLOR1_R** | Hardware cursor color 1 R |
| [15:8] | **HC_COLOR1_G** | Hardware cursor color 1 G |
| [7:0] | **HC_COLOR1_B** | Hardware cursor color 1 B |

### HC Color RAM 2 Register (HC_COLOR2)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_COLOR2 | 0xB000_8084 | R/W | Hardware cursor color ram register mapped to bpp = 2 | 0x0000_0000 |

This register is used to control the color of hardware cursor according to bpp value 2.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HC_COLOR2_R | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HC_COLOR2_G | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HC_COLOR2_B | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:16] | **HC_COLOR2_B** | Hardware cursor color 2 R |
| [15:8] | **HC_COLOR2_G** | Hardware cursor color 2 G |
| [7:0] | **HC_COLOR2_R** | Hardware cursor color 2 B |

## HC Color RAM 3 Register (HC_COLOR3)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_COLOR3 | 0xB000_8088 | R/W | Hardware cursor color ram register mapped to bpp = 3 | 0x0000_0000 |

This register is used to control the color of hardware cursor according to bpp value 3. When transparency is enabled, this color ram will be ignored.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HC_COLOR3_R | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HC_COLOR3_G | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HC_COLOR3_B | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:16] | **HC_COLOR3_R** | Hardware cursor color 3 R. When transparency is enabled, this color ram will be ignored. |
| [15:8] | **HC_COLOR3_G** | Hardware cursor color 3 G. When transparency is enabled, this color ram will be ignored. |
| [7:0] | **HC_COLOR3_B** | Hardware cursor color 3 B. When transparency is enabled, this color ram will be ignored. |

# 7.11 Audio Controller

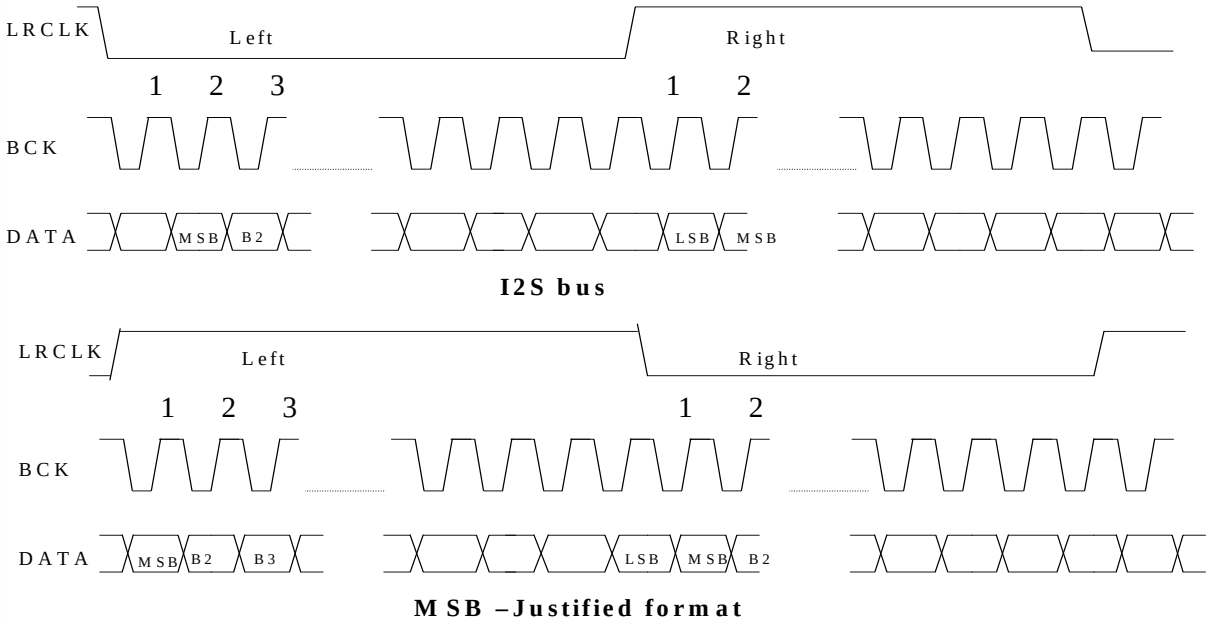
The audio controller consists of IIS/AC-link protocol to interface with external audio CODEC. One 8-level deep FIFO is for read path and write path, and each level has 32-bit width (16 bits for right channel and 16 bits for left channel). One DMA controller handles the data movement between FIFO and memory.

The following are the property of the DMA.

- Always 8-beat incrementing burst
- Always bus lock when 8-beat incrementing burst
- When reach middle and end address of destination address, a DMA_IRQ is requested to CPU automatically

## 7.11.1 IIS Interface

The 16 bits IIS and MSB-justified format are supported; the timing diagram is shown the following.

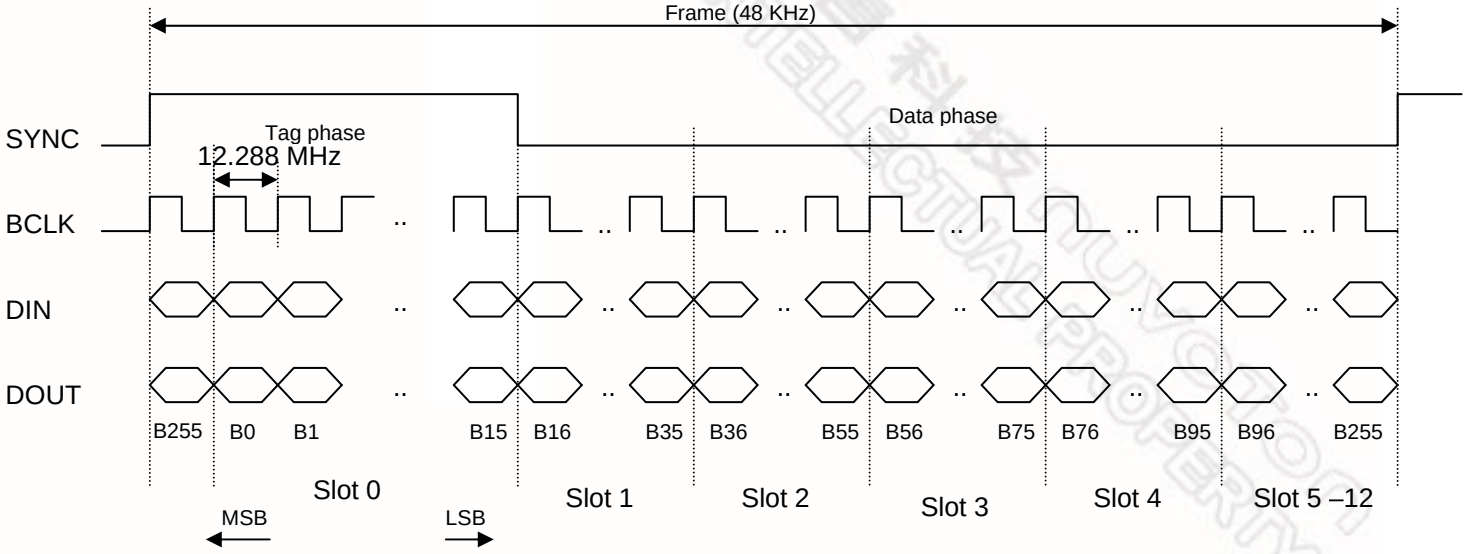



The sampling rate, bit shift clock frequency could be set by the control register ACTL_IISCON.

### 7.11.2 AC97 Interface

The AC97 interface, called AC-link is supported. For input and output direction, each frame contains a Tag slot and 12 data slots. However, in the 12 data slots, only 4 slots are used in this chip, other 8 slots are not supported, and the control data and audio data are transferred in the 4 valid slots. Each slot contains 20 bits data.

### AC97 Interface Signal Format




### 7.11.3 Audio Controller Register Map

**R**: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ACTL_BA = 0xB000_9000** | | | | |
| **ACTL_CON** | 0xB000_9000 | R/W | Audio control register | 0x0000_0000 |
| **ACTL_RESET** | 0xB000_9004 | R/W | Sub block reset control register | 0x0000_0000 |
| **ACTL_RDSTB** | 0xB000_9008 | R/W | DMA destination base address register for record | 0x0000_0000 |
| **ACTL_RDST_LENGTH** | 0xB000_900C | R/W | DMA destination length register for record | 0x0000_0000 |
| **ACTL_RDSTC** | 0xB000_9010 | R | DMA destination current address register for record | 0x0000_0000 |
| **ACTL_RSR** | 0xB000_9014 | R/W | Record status register | 0x0000_0000 |
| **ACTL_PDSTB** | 0xB000_9018 | R/W | DMA destination base address register for play | 0x0000_0000 |
| **ACTL_PDST_LENGTH** | 0xB000_901C | R/W | DMA destination length register for play | 0x0000_0000 |

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ACTL_PDSTC** | 0xB000_9020 | R | DMA destination current address register for play | 0x0000_0000 |
| **ACTL_PSR** | 0xB000_9024 | R/W | Play status register | 0x0000_0004 |
| **ACTL_IISCON** | 0xB000_9028 | R/W | IIS control register | 0x0000_0000 |
| **ACTL_ACCON** | 0xB000_902C | R/W | AC-link control register | 0x0000_0000 |
| **ACTL_ACOS0** | 0xB000_9030 | R/W | AC-link out slot 0 | 0x0000_0000 |
| **ACTL_ACOS1** | 0xB000_9034 | R/W | AC-link out slot 1 | 0x0000_0080 |
| **ACTL_ACOS2** | 0xB000_9038 | R/W | AC-link out slot 2 | 0x0000_0000 |
| **ACTL_ACIS0** | 0xB000_903C | R | AC-link in slot 0 | 0x0000_0000 |
| **ACTL_ACIS1** | 0xB000_9040 | R | AC-link in slot 1 | 0x0000_0000 |
| **ACTL_ACIS2** | 0xB000_9044 | R | AC-link in slot 2 | 0x0000_0000 |
| **ACTL_COUNTER** | 0xB000_9048 | R/W | DMA counter down values | 0xFFFF_FFFF |

## Audio Control Register (ACTL_CON)

The ACTL_CON register control the basic operation of audio controller.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_CON | 0xB000_9000 | R/W | Audio Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | R_DMA_IRQ | T_DMA_IRQ | Reserved | | IIS_AC_PIN_SEL |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FIFO_TH | Reserved | | IRQ_DMA_counter_EN | IRQ_DMA_DATA_zero_EN | BLOCK_EN[1:0] | | Reserved |

| Bits | Descriptions | |
|---|---|---|
| [12] | **R_DMA_IRQ** | **Recording DMA Interrupt Request Bit.**<br>When recording, when the DMA destination current address reach the DMA destination end address or middle address, the R_DMA_IRQ bit will be set to 1 automatically, and this bit could be cleared to 0 by CPU. The bit is hardwired to ARM as interrupt request signal with an inverter.<br>The R_DMA_IRQ bit is read/write (**write 1 to clear**) |
| [11] | **T_DMA_IRQ** | **Transmit DMA Interrupt Request Bit.**<br>When DMA current address reach the middle address (((ACTL_DESE – ACTL_DESB)-1)/2 + ACTL_DESB) or reach the end address ACTL_DESB, the bit T_DMA_IRQ will be set to 1, and this bit could be clear to 0 by write "1" by CPU. And the bit is hardwired to ARM as interrupt request signal with an inverter.<br>The T_DMA_IRQ bit is read/write (**write 1 to clear**). |

| Bits | Descriptions | |
|---|---|---|
| [8] | **IIS_AC_PIN_SEL** | **IIS or AC-link Pin Selection**<br>If IIS_AC_PIN_SEL = 0, the pins select IIS<br>If IIS_AC_PIN_SEL = 1, the pins select AC-link<br>The IIS_AC_PIN_SEL bit is read/write |
| [7] | **FIFO_TH** | **FIFO Threshold Control Bit**<br>If FIFO_TH=0, the FIFO threshold is 8 level<br>If FIFO_TH=1, the FIFO threshold is 4 level<br>The FIFO_TH bit is read/write |
| [4] | **IRQ_DMA_counter_EN** | **IRQ_DMA counter function enable Bit**<br>If IRQ_DMA_counter_EN=0, not allowed to generation T_DMA_IRQ<br>If IRQ_DMA_counter_EN =1, allowed to generation T_DMA_IRQ<br>The IRQ_DMA_counter_EN bit is read/write |
| [3] | **IRQ_DMA_DATA_zero_EN** | **IRQ_DMA_DATA zero and sign detect enable bit**<br>If IRQ_DMA_DATA_zero_EN =0, not allowed to generation T_DMA_IRQ<br>If IRQ_DMA_DATA_zero_EN =1, allowed to generation T_DMA_IRQ<br>The IRQ_DMA_DATA_zero_EN bit is read/write |
| [2:1] | **BLOCK_EN[1:0]** | **Audio Interface Type Selection**<br>If BLOCK_EN[0]=0/1, IIS interface is disable/enable<br>If BLOCK_EN[1]=0/1, AC-link interface is disable/enable<br>The BLOCK_EN[1:0] bits are read/write |

## Sub-block Reset Control Register (ACTL_RESET)

The value of ACTL_RESET register controls the reset operation in each sub block.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_RESET | 0xB000_9004 | R/W | Sub block reset control | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | ACTL_RESET |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RECORD_SINGLE | | PLAY_SINGLE | | Reserved | | | AC_RECORD |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AC_PLAY | IIS_RECORD | IIS_PLAY | DMA_counter_EN | DMA_DATA_zero_EN | Reserved | AC_RESET | IIS_RESET |

| Bits | Descriptions | |
|---|---|---|
| [16] | **ACTL_RESET** | **Audio Controller Reset Control Bit**<br>If ACTL_RESET = 1, the whole audio controller is reset<br>If ACTL_RESET = 0, the audio controller is normal operation<br>The ACTL_RESET bit is read/write |
| [15:14] | **RECORD_SINGLE** | **Record Single / Dual Channel Select Bits**<br>If RECORD_SINGLE[1:0]=11, the record is dual channel<br>If RECORD_SINGLE[1:0]=01, the record only select left channel<br>If RECORD_SINGLE[1:0]=10, the record only select right channel<br>RECORD_SINGLE[1:0]=00 is reserved<br>Note that, when ADC is selected as record path, it only supported left channel record.<br>The PLAY_SINGLE[1:0] bits are read/write |

| Bits | Descriptions | |
|---|---|---|
| [13:12] | **PLAY_SINGLE** | **Playback Single / Dual Channel Select Bits**<br>If PLAY_SINGLE[1:0]=11, the playback is in **stereo** mode<br>If PLAY_SINGLE[1:0]=10, the playback is in **mono** mode<br>PLAY_SINGLE[1:0]= 00 & 01 is reserved<br>The PLAY_SINGLE[1:0] bits are read/write |
| [8] | **AC_RECORD** | **AC link Record Control Bit**<br>If AC_RECORD=0, the record path of AC link is disable<br>If AC_RECORD=1, the record path of AC link is enable<br>The AC_RECORD bit is read/write |
| [7] | **AC_PLAY** | **AC link Playback Control Bit**<br>If AC_PLAY=0, the playback path of AC link is disable<br>If AC_PLAY=1, the playback path of AC link is enable<br>The AC_PLAY bit is read/write |
| [6] | **IIS_RECORD** | **IIS Record Control Bit**<br>If IIS_RECORD=0, the record path of IIS is disable<br>If IIS_RECORD=1, the record path of IIS is enable<br>The IIS_RECORD bit is read/write |
| [5] | **IIS_PLAY** | **IIS Playback Control Bit**<br>If IIS_PLAY=0, the playback path of IIS is disable<br>If IIS_PLAY=1, the playback path of IIS is enable<br>The IIS_PLAY bit is read/write |
| [4] | **DMA_counter_EN** | **DMA counter function enable Bit**<br>If DMA_counter_EN=0, not enable DMA counter function<br>If DMA_counter_EN =1, enable DMA counter function<br>The DMA_counter_EN bit is read/write |

| Bits | Descriptions | |
|---|---|---|
| [3] | **DMA_DATA_zero_EN** | **DMA_DATA zero and sign detect enable bit**<br>If DMA_DATA_zero_EN =0, not enable DMA_DATA zero and sign detect function<br>If DMA_DATA_zero_EN =1, enable DMA_DATA zero and sign detect function<br>The DMA_DATA_zero_EN bit is read/write |
| [1] | **AC_RESET** | **AC link Sub Block RESET Control Bit**<br>If AC_RESET=0, release the AC link function block from reset mode<br>If AC_RESET=1, force the AC link function block to reset mode<br>The AC_RESET bit is read/write |
| [0] | **IIS_RESET** | **IIS Sub Block RESET Control Bit**<br>If IIS_RESET=0, release the IIS function block from reset mode<br>If IIS_RESET=1, force the IIS function block to reset mode<br>The IIS_RESET bit is read/write |

## DMA Record Destination Base Address (ACTL_RDSTB)

The value in ACTL_RDSTB register is the record destination base address of DMA, and only could be changed by CPU.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_RDSTB | 0xB000_9008 | R/W | DMA record destination base address | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| AUDIO_RDSTB[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| AUDIO_RDSTB[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| AUDIO_RDSTB[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AUDIO_RDSTB[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | AUDIO_RDSTB | **32-bit Record Destination Base Address**<br>The AUDIO_RDSTB [31:0] bits are read/write. |

## DMA Destination End Address (ACTL_RDST_LENGTH)

The value in ACTL_RDST_LENGTH register is the record destination address length of DMA, and the register could only be changed by CPU.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_RDST_LENGTH | 0xB000_900C | R/W | DMA record destination address length | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| AUDIO_RDST_L[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| AUDIO_RDST_L[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| AUDIO_RDST_L[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AUDIO_RDST_L[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | AUDIO_RDST_L | **32-bit Record Destination Address Length**<br>The AUDIO_RDST_L [31:0] bits are read/write. |

## DMA Destination Current Address (ACTL_RDSTC)

The value in ACTL_RDSTC is the DMA record destination current address; this register could only be read by CPU.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ACTL_RDSTC** | 0xB000_9010 | R | DMA record destination current address | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **AUDIO_RDSTC[31:24]** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **AUDIO_RDSTC[23:16]** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **AUDIO_RDSTC[15:8]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **AUDIO_RDSTC[7:0]** | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **AUDIO_RDSTC** | **32-bit Record Destination Current Address**<br>The AUDIO_RDSTC [31:0] bits are read only. |

## Audio Controller Record Status Register (ACTL_RSR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_RSR | 0xB000_9014 | R/W | Audio controller FIFO and DMA status register for record | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | R_FIFO_FULL | R_DMA_END_IRQ | R_DMA_MIDDLE_IRQ |

| Bits | Descriptions | |
|---|---|---|
| [2] | R_FIFO_FULL | **Record FIFO Full Indicator Bit**<br>If R_FIFO_FULL=0, the record FIFO not full<br>If R_FIFO_FULL=1, the record FIFO is full<br>The R_FIFO_READY bit is read only |
| [1] | R_DMA_END_IRQ | **DMA End Address Interrupt Request Bit for Record**<br>If R_DMA_END_IRQ=0, means record DMA address does not reach the end address<br>If R_DMA_END_IRQ=1, means record DMA address reach the end address<br>The R_DMA_END_IRQ bit is readable, and only can be clear by write "1" to this bit |

| Bits | Descriptions | |
|---|---|---|
| [0] | **R_DMA_MIDDLE_IRQ** | **DMA Address Interrupt Request Bit for Record**<br>If R_DMA_MIDDLE_IRQ=0, means record DMA address does not reach the middle address<br>If R_DMA_MIDDLE_IRQ=1, means record DMA address reach the middle address<br>The R_DMA_MIDDLE_IRQ bit is readable, and only can be clear by write "1" to this bit |

## DMA Play Destination Base Address (ACTL_PDSTB)

The value in ACTL_PDSTB register is the play destination base address of DMA, and only could be changed by CPU.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_PDSTB | 0xB000_9018 | R/W | DMA play destination base address | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| AUDIO_PDSTB[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| AUDIO_PDSTB[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| AUDIO_PDSTB[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AUDIO_PDSTB[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | AUDIO_PDSTB | **32-bit Play Destination Base Address**<br>The AUDIO_PDSTB [31:0] bits are read/write. |

## DMA Destination End Address (ACTL_PDST_LENGTH)

The value in ACTL_PDST_LENGTH register is the play destination address length of DMA, and the register could only be changed by CPU.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_PDST_LENGTH | 0xB000_901C | R/W | DMA play destination address length | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| AUDIO_PDST_L[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| AUDIO_PDST_L[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| AUDIO_PDST_L[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AUDIO_PDST_L[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | AUDIO_PDST_L | **32-bit Play Destination Address Length**<br>The AUDIO_PDST_L [31:0] bits are read/write. |

## DMA Destination Current Address (ACTL_PDSTC)

The value in ACTL_PDSTC is the DMA play destination current address; this register could only be read by CPU.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_PDSTC | 0xB000_9020 | R | DMA play destination current address | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| AUDIO_PDSTC[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| AUDIO_PDSTC[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| AUDIO_PDSTC[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AUDIO_PDSTC[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | AUDIO_PDSTC | **32-bit Play Destination Current Address**<br>The AUDIO_PDSTC [31:0] bits are read/write. |

## Audio Controller Playback Status Register (ACTL_PSR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_PSR | 0xB000_9024 | R/W | Audio controller FIFO and DMA status register for playback | 0x0000_0004 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | DMA_counter_IRQ | DMA_DATA_zero_IRQ | P_FIFO_EMPTY | P_DMA_END_IRQ | P_DMA_MIDDLE_IRQ |

| Bits | Descriptions | |
|---|---|---|
| [4] | **DMA_counter_IRQ** | **DMA counter IRQ**<br>If DMA_counter_IRQ=0, not found DMA_counter to zero<br>If DMA_counter_IRQ =1, DMA_COUNTER counter down to zero<br>The DMA_counter_IRQ bit is readable ,<br>and only can be clear by **write "1" to clear this bit** |
| [3] | **DMA_DATA_zero_IRQ** | **DMA_DATA zero IRQ**<br>If DMA_DATA_zero_IRQ =0, not found DMA DATA is zero or sign change(two channel)<br>If DMA_DATA_zero_IRQ =1, found DMA DATA is zero or sign change (two channel)<br>The DMA_DATA_zero_IRQ bit is readable ,<br>and only can be clear by **write "1" to clear this bit** |
| [2] | **P_FIFO_EMPTY** | **Playback FIFO Empty Indicator Bit**<br>If P_FIFO_EMPTY=0, the playback FIFO is not empty<br>If P_FIFO_EMPTY=1, the playback FIFO is empty<br>The P_FIFO_EMPTY bit is read only |

| Bits | Descriptions | |
|---|---|---|
| [1] | **P_DMA_END_IRQ** | **DMA End Address Interrupt Request Bit for Playback**<br>If P_DMA_END_IRQ=0, means playback DMA address does not reach the end address<br>If P_DMA_END_IRQ=1, means playback DMA address reach the end address<br>The P_DMA_END_IRQ bit is readable, and only can be clear by write "1" to this bit |
| [0] | **P_DMA_MIDDLE_IRQ** | **DMA Address Interrupt Request Bit for Playback**<br>If P_DMA_MIDDLE_IRQ=0, means playback DMA address does not reach the middle address<br>If P_DMA_MIDDLE_IRQ=1, means playback DMA address reach the middle address<br>The P_DMA_MIDDLE_IRQ bit is readable, and only can be clear by write "1" to this bit |

| **Play** (0xB000_9004;bit7,5 ) | **DMA_DATA_zero_EN** (0xB000_9004;bit 3) | **DMA_DATA_zero_IRQ** (0xB000_9024; bit 3) | |
|---|---|---|---|
| 1 | 0 | 0 | **play** |
| 1 | 0 | 0 | **Play** |
| 1 | 1 | 0 | **Play** |
| 1 | 1 | 1 | **Play**<br>**(output 0,DMA not stop)** |
| 0 | 0 | 0 | **stop** |
| 0 | 0 | 0 | **Stop** |
| 0 | 1 | 0 | **Play** |
| 0 | 1 | 1 | **Stop**<br>**(DMA stop and output 0 after output data is zero)** |

## IIS Control Register (ACTL_IISCON)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_IISCON | 0xB000_9028 | R/W | IIS control register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | PRS[3:0] | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BCLK_SEL[1:0] | | FS_SEL | MCLK_SEL | FORMAT | RESERVED | | |

| Bits | Descriptions |
|---|---|

| Bits | Descriptions | |
|---|---|---|
| [19:16] | **PRS** | **IIS Frequency Pre-scalar Selection Bits.**<br>(FPLL is the Audio Frequency from PLL, IIS_SYSCLK is the output main clock)<br>If PSR[3:0]=0000, IIS_SYSCLK=FPLL/1<br>If PSR[3:0]=0001, IIS_SYSCLK=FPLL/2<br>If PSR[3:0]=0010, IIS_SYSCLK=FPLL/3<br>If PSR[3:0]=0011, IIS_SYSCLK=FPLL/4<br>If PSR[3:0]=0100, IIS_SYSCLK=FPLL/5<br>If PSR[3:0]=0101, IIS_SYSCLK=FPLL/6<br>If PSR[3:0]=0110, IIS_SYSCLK=FPLL/7<br>If PSR[3:0]=0111, IIS_SYSCLK=FPLL/8<br>If PSR[3:0]=1000, reserved<br>If PSR[3:0]=1001, IIS_SYSCLK=FPLL/10<br>If PSR[3:0]=1010, reserved<br>If PSR[3:0]=1011, IIS_SYSCLK=FPLL/12<br>If PSR[3:0]=1100, reserved<br>If PSR[3:0]=1101, IIS_SYSCLK=FPLL/14<br>If PSR[3:0]=1110, reserved<br>If PSR[3:0]=1111, IIS_SYSCLK=FPLL/16<br>(when the division factor is 3/5/7, the duty cycle of MCLK is not 50%, the high duration is 0.5*FPLL)<br>The PSR[3:0] bits are read/write |
| [7:6] | **BCLK_SEL** | **IIS Serial Data Clock Frequency Selection Bit**<br>If BCLK_SEL[1:0]=00, the frequency of bit clock (BCLK) is IIS_SYSCLK/8,<br>If BCLK_SEL [1:0] =01, the frequency of bit clock (BCLK) is IIS_SYSCLKK/12.<br>The BCLK_SEL[1:0] bits are read/write |

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[5]</td><td>FS_SEL</td><td>
IIS Sampling Frequency Selection Bit<br>
If BCLK_SEL[1:0]=00, and FS_SEL=0, 32fs is selected, the sampling frequency (LRCLK) = IIS_SYSCLK/(8*32) = IIS_SYSCLK/(256)<br>
If BCLK_SEL[1:0]=00, and FS_SEL=1, 48fs is selected, the sampling frequency (LRCLK) = IIS_SYSCLK/(8*48) = IIS_SYSCLK/(384)<br>
If BCLK_SEL[1:0]=01, this bit is ignored, 32fs is selected, the sampling frequency (LRCLK) = IIS_SYSCLK/(12*32) = IIS_SYSCLK/(384)<br>
(fs is sampling rate)<br>
The FS_SEL bit is read/write<br>
Example:
<table>
<tr><th>IIS_SYSCLK</th><th>Sample Rate</th><th>Sample Freq.</th><th>BCLK_SEL</th><th>FS_SEL</th></tr>
<tr><td>12.288MHz</td><td>32fs</td><td>48.0KHz</td><td>00</td><td>0</td></tr>
<tr><td>16.934MHz</td><td>32fs</td><td>44.1KHz</td><td>01</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>[4]</td><td>MCLK_SEL</td><td>
IIS_SYSCLK Output Selection Bit<br>
If MCLK_SEL=0, IIS_SYSCLK output will follow the PRS [3:0] setting.<br>
If MCLK_SEL=1, IIS_SYSCLK output will be the same with FPLL.<br>
The MCLK_SEL bit is read/write
</td></tr>
<tr><td>[3]</td><td>FORMAT</td><td>
IIS Format Selection Bits<br>
If FORMAT=0, IIS compatible format is selected<br>
If FORMAT=1, MSB-justified format is selected<br>
The FORMAT bit is read/write
</td></tr>
</table>

## AC-link Control Register (ACTL_ACCON)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_ACCON | 0xB000_902C | R/W | AC-link control register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | AC_BCLK_PU_EN | AC_R_FINISH | AC_W_FINISH | AC_W_RES | AC_C_RES | RESERVED |

| Bits | Descriptions | |
|---|---|---|
| [5] | AC_BCLK_PU_EN | **AC_BCLK Pin Pull-high Resister Enable**<br>If AC_BCLK_PU_EN=0, the AC_BCLK pin pull-high resister will be disabled<br>If AC_BCLK_PU_EN=1, the AC_BCLK pin pull-high resister will be enabled<br>The AC_BCLK_PU_EN bit is read/write. |
| [4] | AC_R_FINISH | **AC-link Read Data Ready Bit.**<br>When read data indexed by previous frame is shifted into ACTL_ACIS2, the AC_R_FINISH bit will be set to 1 automatically. After CPU read out the read data, AC_R_FINISH bit will be cleared to 0.<br>If AC_R_FINISH=0, read data buffer has been read by CPU<br>If AC_R_FINISH=1, read data buffer is ready for CPU read<br>The AC_R_FINISH bit is read only |

| Bits | Descriptions | |
|---|---|---|
| [3] | **AC_W_FINISH** | **AC-link Write Frame Finish Bit.**<br>When writing data to register ACTL_ACOS0, the AC_W_FINISH bit will be set to 1 automatically. After AC-link interface shift out the register ACTL_ACOS0, the AC_W_FINISH bit will be cleared to 0.<br>If AC_W_FINISH=0, AC-link control data out buffer has been shifted out to codec by CPU and data out buffer is empty.<br>If AC_W_FINISH=1, AC-link control data out buffer is ready to be shifted out(After users have wrote data into register ACTL_ACOS0)<br>The AC_W_FINISH bit is read only |
| [2] | **AC_W_RES** | **AC-link Warm Reset Control Bit**<br>When this bit is set to 1, (AC-link begin warn reset procedure, after warn reset procedure finished, this bit will be cleared automatically) the interface signal AC_SYNC is high, when this bit is set to 0, the interface signal AC_SYNC is controlled by AC_BCLK input when this bit is set to 1. Note the AC-link spec. shows it need at least 10 us high duration of AC_SYNC to warn reset AC97.<br>If AC_W_RES=0, AC_SYNC pin is controlled by AC_BCLK input pin<br>If AC_W_RES=1, AC_SYNC pin is forced to high<br>The AC_W_RES bit is read/write |
| [1] | **AC_C_RES** | **AC-link Cold Reset Control Bit**<br>When this bit is set to 1, the interface signal AC_RESETB is low, when this bit is set to 0, the signal AC_RESETB is high. Note the AC-link spec. shows it need at least 10 us low duration of AC_RESETB to cold reset AC97.<br>If AC_C_RES=0, AC_RESETB pin is set to 1<br>If AC_C_RES=1, AC_RESETB pin is set to 0<br>The AC_C_RES bit is read/write |

## AC-link output slot 0 (ACTL_ACOS0)

The ACTL_ACOS0 register store the slot 0 value to be shift out by AC-link. Note that write data to ACTL_ACOS0 register when AC_W_FINISH bit (ACTL_ACCON [3]) set is invalid. Therefore, check AC_W_FINISH bit status before write data into ACTL_ACOS0 register.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ACTL_ACOS0** | 0xB000_9030 | R/W | AC-link out slot 0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **RESERVED** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **RESERVED** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **RESERVED** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **RESERVED** | | | **VALID_ FRAME** | **SLOT_VALID[3:0]** | | | |

| Bits | Descriptions | |
|---|---|---|
| [4] | **VALID_FRAME** | **Frame Valid Indicated Bits**<br>VALID_FRAME=1, any one of slot is valid<br>VALID_FRAME=0, no any slot is valid<br>The VALID_FRAME bits are read/write |
| [3:0] | **SLOT_VALID[3:0]** | **Slot Valid Indicated Bits**<br>SLOT_VALID[0]= 1/0, indicate Slot 1 valid/invalid<br>SLOT_VALID[1]= 1/0, indicate Slot 2 valid/invalid<br>SLOT_VALID[2]= 1/0, indicate Slot 3 valid/invalid<br>SLOT_VALID[3]= 1/0, indicate Slot 4 valid/invalid<br>The SLOT_VALID[3:0] bits are read/write |

## The AC-link output slot 1 (ACTL_ACOS1)

The ACTL_ACOS1 register store the slot 1 value to be shift out by AC-link.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_ACOS1 | 0xB000_9034 | R/W | AC-link out slot 1 | 0x0000_0080 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R_WB | R_INDEx[6:0] | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [7] | R_WB | **Read/Write Select Bit**<br>If R_WB=1, a read specified by R_INDEx[6:0] will occur, and the data will appear in next frame<br>If R_WB=0, a write specified by R_INDEx[6:0] will occur, and the write data is put at out slot 2<br>The R_WB bit is read/write |
| [6:0] | R_INDEx[6:0] | **External AC97 CODEC Control Register Index (address) Bits**<br>The R_INDEx[6:0] bits are read/write |

## AC-link output slot 2 (ACTL_ACOS2)

The ACTL_ACOS2 register store the slot 2 value to be shift out by AC-link.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_ACOS2 | 0xB000_9038 | R/W | AC-link out slot 2 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WD[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WD[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **WD[15:0]** | **AC-link Write Data**<br>The WD[15:0] bits are read/write |

## AC-link input slot 0 (ACTL_ACIS0)

The ACTL_ACIS0 store the shift in slot 0 data of AC-link.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_ACIS0 | 0xB000_903C | R | AC-link in slot 0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | CODEC_READY | SLOT_VALID[3:0] | | | |

| Bits | Descriptions | |
|---|---|---|
| [4] | **CODEC_READY** | **External AC97 Audio CODEC Ready Bit**<br>If CODEC_READY=0, indicate external AC97 audio CODEC is not ready<br>If CODEC_READY=1, indicate external AC97 audio CODEC is ready<br>The CODEC_READY bit is read only |
| [3:0] | **SLOT_VALID[3:0]** | **Slot Valid Indicated Bits**<br>SLOT_VALID[0]= 1/0, indicate Slot 1 valid/invalid<br>SLOT_VALID[1]= 1/0, indicate Slot 2 valid/invalid<br>SLOT_VALID[2]= 1/0, indicate Slot 3 valid/invalid<br>SLOT_VALID[3]= 1/0, indicate Slot 4 valid/invalid<br>The SLOT_VALID[3:0] bits are read |

## AC-link input slot 1 (ACTL_ACIS1)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_ACIS1 | 0xB000_9040 | R | AC-link in slot 1 | 0x0000_0000 |

The ACTL_ACIS1 stores the shift in slot 1 data of AC-link.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | R_INDEx[6] |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R_INDEx[5:0] | | | | | | SLOT_REQ[1:0] | |

| Bits | Descriptions | |
|---|---|---|
| [8:2] | **R_INDEx[6:0]** | **Register Index.**<br>The R_INDEx [6:0] echo the register index (address) when a register read has been requested in the previous frame.<br>The R_INDEx[6:0] bits are read only |
| [1:0] | **SLOT_REQ[1:0]** | **Slot Request.**<br>The bits indicate if the external codec need new PCM data that will transfer in next frame.<br>Any bit in SLOT_REQ[1:0] is set to 1, indicate external codec does not need a new sample in the corresponding slot[3:4] of the next frame<br>Any SLOT_REQ[1:0] is clear to 0, indicate external codec need a new sample in the corresponding slot[3:4] of the next frame<br>The SLOT_REQ[1:0] bits are read only |

## AC-link input slot 2 (ACTL_ACIS2)

The ACTL_ACIS2 stores the shift in slot 2 data of AC-link.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_ACIS2 | 0xB000_9044 | R | AC-link in slot 2 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RD[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RD[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **RD[15:0]** | **AC-link Read Data.**<br>The RD[15:0] bits are read only |

### DOWN_COUNTER Control Register (ACTL_counter)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ACTL_COUNTER | 0xB000_9048 | R/W | DMA down counter register | 0xFFFF_FFFF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| ACTL_COUNTER[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ACTL_COUNTER[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ACTL_COUNTER[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ACTL_COUNTER[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **ACTL_COUNTER** | **ACTL_COUNTER is Read and Write Data.**<br>The ACTL_COUNTER [31:0] bits are read and write, When the register is Zero that set DMA_counter_IRQ bit =1. |

## 7.12 2-D Graphic Engine

A 32-bit 2D Graphics Engine (2D GE) is specially designed to improve the performance of graphic processing. It can accelerate the operation of individual GUI functions such as BitBLTs and Bresenham Line Draw to operate at all pixel depths including 8/16/32 bit-per-pixel. A pixel is the smallest addressable screen element as defined in Microsoft Windows, and lines and pictures are made up by a variety of pixels. 2D GE is used to speed up graphic performance in pixel data moving and line drawing, as well as to accelerate almost all computer graphic Boolean operations by eliminating the CPU overhead. Meanwhile, the functions of rotation and scaling down are implemented for some special applications. In image scaling down function, both programmable horizontal and vertical N/M scaling down factors are provided for resizing the image. For the 2D rotation, it can rotate left or right 45, 90 or 180 degrees, and also supports the flip/flop, mirror or up-side-down pictures.

### 7.12.1 2-D Graphic Engine Control Registers Map

**R**: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE_BA = 0xB000_B000** | | | | |
| **2D_GETG** | 0xB000_B000 | R/W | Graphic Engine Trigger Control Register | 0x0000_0000 |
| **2D_GEXYSORG** | 0xB000_B004 | R/W | Graphic Engine XY Mode Source Origin Starting Register | 0x0000_0000 |
| **2D_TileXY_VHSF** | 0xB000_B008 | R/W | Graphic Engine Tile Width/Height or V/H Scale Factor N/M | 0x0000_0000 |
| **2D_GERRXY** | 0xB000_B00C | R/W | Graphic Engine Rotate Reference Point XY Address | 0x0000_0000 |
| **2D_GEINTS** | 0xB000_B010 | R/W | Graphic Engine Interrupt Status Register | 0x0000_0000 |
| **2D_GEPLS** | 0xB000_B014 | R/W | Graphic Engine Pattern Location Starting Address Register | 0x0000_0000 |
| **2D_GEBER** | 0xB000_B018 | R/W | GE Bresenham Error Term Stepping Constant Register | 0x0000_0000 |
| **2D_GEBIR** | 0xB000_B01C | R/W | GE Bresenham Initial Error, Pixel Count Major M Register | 0x0000_0000 |
| **2D_GEC** | 0xB000_B020 | R/W | Graphic Engine Control Register | 0x0000_0000 |
| **2D_GEBC** | 0xB000_B024 | R/W | Graphic Engine Background Color Register | 0x0000_0000 |
| **2D_GEFC** | 0xB000_B028 | R/W | Graphic Engine Foreground Color Register | 0x0000_0000 |
| **2D_GETC** | 0xB000_B02C | R/W | Graphic Engine Transparency Color Register | 0x0000_0000 |
| **2D_GETCM** | 0xB000_B030 | R/W | Graphic Engine Transparency Color Mask Register | 0x0000_0000 |
| **2D_GEXYDORG** | 0xB000_B034 | R/W | Graphic Engine XY Mode Display Origin Starting Register | 0x0000_0000 |
| **2D_GESDP** | 0xB000_B038 | R/W | Graphic Engine Source/Destination Pitch Register | 0x0000_0000 |
| **2D_GESSXYL** | 0xB000_B03C | R/W | Graphic Engine Source Start XY/Linear Address Register | 0x0000_0000 |
| **2D_GEDSXYL** | 0xB000_B040 | R/W | Graphic Engine Destination Start XY/Linear Register | 0x0000_0000 |
| **2D_GEDIXYL** | 0xB000_B044 | R/W | Graphic Engine Dimension XY/Linear Register | 0x0000_0000 |
| **2D_GECBTL** | 0xB000_B048 | R/W | Graphic Engine Clipping Boundary Top/Left Register | 0x0000_0000 |
| **2D_GECBBR** | 0xB000_B04C | R/W | Graphic Engine Clipping Boundary Bottom/Right Register | 0x0000_0000 |
| **2D_GEPTNA** | 0xB000_B050 | R/W | Graphic Engine Pattern A Register | 0x0000_0000 |
| **2D_GEPTNB** | 0xB000_B054 | R/W | Graphic Engine Pattern B Register | 0x0000_0000 |
| **2D_GEWPM** | 0xB000_B058 | R/W | Graphic Engine Write Plane Mask Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **2D_GEMC** | 0xB000_B05C | R/W | Graphic Engine Miscellaneous Control Register | 0x0000_0000 |

### 7.12.2 2-D Graphic Engine Control Registers

## Graphic Engine Trigger Control Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **2D_GETG** | 0xB000_B000 | R/W | Graphic Engine Trigger Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | | | | **GO** |

| Bits | Descriptions | |
|---|---|---|
| [0] | **GO** | Trigger Graphics Engine Acceleration<br>1 = Start GE acceleration, it will automatically be cleared when job completed.<br>0 = No acceleration or the acceleration is finished. |

## Graphic Engine XY Mode Source Memory Origin Starting Address Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEXYSORG | 0xB000_B004 | R/W | X/Y Addressing Mode Source Origin Starting Address | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | X/Y Source Origin starting address | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| X/Y Source Origin starting address | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| X/Y Source Origin starting address | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| X/Y Source Origin starting address | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [27:0] | X/Y Origin Starting Address | **28-bit X/Y Mode Origin Starting Address (byte unit)**<br>This 28-bit byte address specifies the starting address of an object or a picture in the display memory when addressed by X/Y mode. This value is specified by bytes and should be 16K word (64K bytes) boundary. That is, the bits 15-0 are ignored. |

### Graphic Engine Tile Width/Height Numbers and DDA V/H Scale Up/Down Factors

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **2D_TileXY** | 0xB000_B008 | R/W | 2D Tile Width X and Tile Height Y Register | 0x0000_0000 |
| **2D_VHSF** | 0xB000_B008 | R/W | DDA Vertical and Horizontal Scaling Down Factor N/M | 0x0000_0000 |

**(A)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Tile Height Y [7:0]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Tile Width X [7:0]** | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:8] | **Tile Height Y** | **8-bit tile height Y value**<br>This divider provides the tile height Y value. |
| [7:0] | **Tile Width X** | **8-bit tile width X value**<br>This divider provides the tile width X value. |

**(B)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **VSF_N [7:0]** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **VSF_M [7:0]** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **HSF_N [7:0]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **HSF_M [7:0]** | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:24] | **VSF_N** | **8-bit Vertical N Scaling Factor**<br>An 8-bit value specifies the numerator part (N) of the vertical scaling factor in graphic engine. The output image height will be equal to the input image height x N / M. ***The value of N must be equal or less than M.*** |
| [23:16] | **VSF_M** | **8-bit Vertical M Scaling Factor**<br>An 8-bit value specifies the denominator part (M) of the vertical scaling factor in graphic engine. The output image height will be equal to the input image height x N / M. ***The value of N must be equal or less than M.*** |
| [15:8] | **HSF_N** | **8-bit Horizontal N Scaling Factor**<br>An 8-bit value specifies the numerator part (N) of the horizontal scaling factor in graphic engine. The output image width will be equal to the input image width x N / M. ***The value of N must be equal or less than M.*** |
| [7:0] | **HSF_M** | **8-bit Horizontal M Scaling Factor**<br>An 8-bit value specifies the denominator part (M) of the horizontal scaling factor in graphic engine. The output image width will be equal to the input image width x N / M. ***The value of N must be equal or less than M.*** |

## Graphic Engine Rotate Reference Point XY Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GERRXY | 0xB000_B00C | R/W | Graphic Engine Rotate Reference Point in X/Y (pixel) | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | Rotate Reference Y [10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Rotate Reference Y [7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | Rotate Reference X [10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Rotate Reference X [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **Rotate Reference Y** | **11-bit Rotate Reference Y**<br>For Rotation in X/Y addressing, this register specifies the reference point Y in pixels. |
| [10:0] | **Rotate Reference X** | **11-bit Rotate Reference X**<br>For Rotation in X/Y addressing, this register specifies the reference point X in pixels. |

## Graphic Engine Interrupt Status Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEINTS | 0xB000_B010 | R/W | Graphic Engine Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | INTS |

| Bits | Descriptions | |
|---|---|---|
| [0] | INTS | **GE interrupt status**<br>0 = No interrupt occur<br>1 = Interrupt occur, host writes one to clear INTS. |

## Graphic Engine Pattern Location Starting Address Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEPLS | 0xB000_B014 | R/W | Pattern Location Starting Address | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | Pattern Location [27:24] | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Pattern Location [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Pattern Location [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Pattern Location [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [27:0] | Pattern Location | **28-bit Pattern Location (byte unit)**<br>The byte address of 28-bit pattern specifies the beginning location of an 8×8 pixel pattern stored in the off-screen memory when in BitBLT operation. This value must be programmed on an M-byte boundary. M=8*8*BPP/8 bytes, where BPP=8/16/32. |

## Graphic Engine Bresenham Error Term Stepping Constant Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEBER | 0xB000_B018 | R/W | Bresenham Error Term Stepping Constant Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Diagonal Error Increment [13:8] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Diagonal Error Increment [7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Axial Error Increment [13:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Axial Error Increment [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [29:16] | **Diagonal Error Increment** | **14-bit Diagonal Error Increment**<br>For Bresenham line draw, this register specifies the constant to be added to the Error Term for diagonal stepping (Error > 0). The initial value is (2 * (delta Y - delta X)) after normalization to first octant. |
| [13:0] | **Axial Error Increment** | **14-bit Axial Error Increment**<br>For Bresenham line draw, this register specifies the constant to be added to the Error Term for axial stepping (Error < 0). The initial value is (2 * delta Y) after normalization to first octant. |

## Graphic Engine Bresenham Initial Error, Pixel Count Major –1 Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEBIR | 0xB000_B01C | R/W | Bresenham Initial Error, Pixel Count Major –1 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Initial Error Term [13:8] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Initial Error Term [7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Line Pixel Count Major -1 [10:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Line Pixel Count Major -1 [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [29:16] | **Initial Error Term** | **14-bit Initial Error Term**<br>For Bresenham line draw, this register specifies the initial Error Term. The initial value is (2 * (delta Y) - delta X) after normalization to first octant. |
| [10:0] | **Line Pixel Count Major -1** | **11-bit Line Pixel Count Major -1**<br>For Bresenham line draw, this register specifies the pixel count of major axis |

## Graphic Engine Control Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEC | 0xB000_B020 | R/W | Graphic Engine Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| ROP (Raster Operation Code) | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| COMMAND | | ALPHA_BLND | LINE_STYLE | M/D | LSTP | INT_EN | ADDR_MD |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TRANSPARENCY | | MTS | CTS | CTP | AU | CLIP_EN | CLPC |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SDT | SRCS | | PDT | XY Octant | | | DDTO |

| Bits | Descriptions | |
|---|---|---|
| [31:24] | ROP | **ROP Code**<br>It supports all Microsoft 256 Raster Operation Codes. Each raster operation code is an 8-bit value that represents the result of the Boolean operation on pre-defined pattern, source, and destination. |
| [23:22] | COMMAND | **Graphics Engine Command**<br>00 = No operation<br>01 = BitBLT acceleration<br>10 = Bresenham Line Draw acceleration<br>11 = Rectangle Border drawing |
| [21] | ALPHA BLENDING | **Alpha Blending Control**<br>0 = Disable alpha blending<br>1 = Enable alpha blending |
| [20] | LINE STYLE | **Line Style Control**<br>0 = Disable line style<br>1 = Enable line style |

| Bits | Descriptions | |
|---|---|---|
| [19] | **M / D** | **Bresenham Line Move / Draw**<br>0 = Move<br>1 = Draw |
| [18] | **LSTP** | **Last Pixel Draw / Move or Scale Up / Down Rectangular Object**<br>0 = Last pixel of Bresenham line will be drawn; 0 = Scaling down object.<br>1 = Last pixel of Bresenham line will not be drawn; 1 = Scaling up object. |
| [17] | **INT_EN** | **Interrupt Enable**<br>When BitBLT/Bresenham Line Draw acceleration is complete or finished. |
| [16] | **ADDR_MD** | **Graphics Engine Addressing Mode**<br>0 = Linear addressing mode<br>1 = X/Y addressing mode |
| [15:14] | **TRANSPX** | **GE Transparency**<br>00 = Disabled<br>01 = Mono transparency<br>10 = Color transparency<br>11 = Reserved |
| [13] | **MTS** | **Mono Transparency Select**<br>0 = Source<br>1 = Pattern |
| [12] | **CTS** | **Color Transparency Select**<br>0 = Source pixels control transparency<br>1 = Destination pixels control transparency |
| [11] | **CTP** | **Color Transparency Polarity**<br>0 = Matching pixels are transparent<br>1 = Matching pixels are opaque |
| [10] | **AU** | **Auto Update**<br>0 = Disable<br>1 = Enable. Destination X, Y register is automatically updated at the end of each BitBLT operation. |

| Bits | Descriptions | |
|---|---|---|
| [9] | **CLIP_EN** | **Clipping Enable**<br>0 = Disabled<br>1 = Enabled |
| [8] | **CLPC** | **Clipping Control**<br>0 = Only pixels inside the clipping rectangle are drawn<br>1 = Only pixels outside the clipping rectangle are drawn |
| [7] | **SDT** | **Source Data Type**<br>0 = Color<br>1 = Mono<br>**Note:** Source and pattern data are not allowed to be both in mono format. |
| [6:5] | **SRCS** | **Source Data Select**<br>00 = Display memory<br>01 = System memory<br>10 = GE background color<br>11 = GE foreground color |
| [4] | **PDT** | **Pattern Data Type**<br>0 = Color (from display memory)<br>1 = Mono (from internal pattern registers) |
| [3:1] | **XY Octant** | **XY Octant**<br>It determines the drawing directions for BitBLT, Bresenham line, and Rotate.<br>000 = Right-down (BitBLT);+ X, + Y, \|DX\| ≥ \|DY\| (Line); Scaling Down<br>001 = Right-down (BitBLT);+ X, + Y, \|DX\| < \|DY\| (Line); Rotate right 45°<br>010 = Left-down (BitBLT); + X, – Y, \|DX\| ≥ \|DY\| (Line); Rotate left 45°<br>011 = Left-down (BitBLT); + X, – Y, \|DX\| < \|DY\| (Line); Rotate left 90°<br>100 = Right-up (BitBLT); – X, + Y, \|DX\| ≥ \|DY\| (Line); Up-Side-Down<br>101 = Right-up (BitBLT); – X, + Y, \|DX\| < \|DY\| (Line); Rotate right 90°<br>110 = Left-up (BitBLT); – X, – Y, \|DX\| ≥ \|DY\| (Line); Rotate 180°<br>111 = Left-up (BitBLT); – X, – Y, \|DX\| < \|DY\| (Line); Mirror or Flop |
| [0] | **DDTO** | **Destination Data Direction, new destination data to**<br>0 = Display memory<br>1 = System memory |

## Graphic Engine Background Color Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEBC | 0xB000_B024 | R/W | Graphic Engine Background Color | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Background Color [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Background Color [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Background Color [7:0] | | | | | | | |

=

| Bits | Descriptions | |
|---|---|---|
| [23:0] | **Background Color** | **24-bit Background Color**<br>These bits specify the background color for Graphics Engine. Only the corresponding number of bits-per-pixel in the display mode is required in the register. In RGB 8:8:8 color mode, bits 23-16 have the red value, bits 15-8 have the green value, and bits 7-0 have the blue value. |

## Graphic Engine Foreground Color Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEFC | 0xB000_B028 | R/W | Graphic Engine Foreground Color | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Foreground Color [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Foreground Color [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Foreground Color [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | Foreground Color | **Graphics Engine Foreground Color**<br>These bits specify the foreground color for Graphics Engine. Only the corresponding number of bits-per-pixel in the display mode is required in the register. In RGB 8:8:8 color mode, bits 23-16 have the red value, bits 15-8 have the green value, and bits 7-0 have the blue value. |

## Graphic Engine Transparency Color Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GETC | 0xB000_B02C | R/W | Graphic Engine Transparency Color | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Transparency Color [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Transparency Color [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Transparency Color [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | **Transparency Color** | **24-bit Transparency Color**<br>These bits specify the transparency color for Graphics Engine. Only the corresponding number of bits-per-pixel in the display mode is required in the register. In RGB 8:8:8 color mode, bits 23-16 have the red value, bits 15-8 have the green value, and bits 7-0 have the blue value. |

## Graphic Engine Transparency Color Mask Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GETCM | 0xB000_B030 | R/W | Graphic Engine Transparency Color Mask | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Transparency Color Mask [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Transparency Color Mask [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Transparency Color Mask [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | **Transparency Color Mask** | **24-bit Transparency Color Mask**<br>These bits specify a mask for use in comparison against the transparency color. Only the corresponding number of bits-per-pixel in the display mode is required in the register. |

## Graphic Engine XY Mode Display Memory Origin Starting Address Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEXYDORG | 0xB000_B034 | R/W | X/Y Addressing Mode Display Origin Starting Address | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| X/Y Display Origin starting address [27:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| X/Y Display Origin starting address [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| X/Y Display Origin starting address [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| X/Y Display Origin starting address [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [27:0] | X/Y Origin Starting Address | **28-bit X/Y Mode Origin Starting Address (byte unit)**<br>This 28-bit byte address specifies the starting address of an object or a picture in the display memory when addressed by X/Y mode. This value is specified by bytes. |

## Graphic Engine Source / Destination Pitch Register

| Register | Address | R / W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GESDP | 0xB000_B038 | R/W | Graphic Engine Source/Destination Pitch | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Destination Pitch [12:8] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Destination Pitch [7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Source Pitch [12:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Source Pitch [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [28:16] | **Destination Pitch** | **Bits 28-16** Destination Pitch<br>This 13-bit register specifies the destination pitch in bytes in linear addressing mode, and in X/Y addressing by pixels. |
| [12:0] | **Source Pitch** | **Bits 12-0** Source Pitch<br>This 13-bit register specifies the source pitch in bytes in linear addressing mode, and in X/Y addressing by pixels. |

## Graphic Engine Source Start XY/Linear Addressing Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **2D_GESSXY** | 0xB000_B03C | R/W | Graphic Engine Source Start in X/Y addressing (pixel) | 0x0000_0000 |
| **2D_GESSL** | 0xB000_B03C | R/W | Graphic Engine Source Start in linear addressing (byte) | 0x0000_0000 |

**(A)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Source Start Y [10:8]** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Source Start Y [7:0]** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Source Start X [10:8]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Source Start X [7:0]** | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **Source Start Y** | **11-bit Source Start Y**<br>For BitBLTs in X/Y addressing, this register specifies the source start Y in pixels. |
| [10:0] | **Source Start X** | **11-bit Source Start X**<br>For BitBLTs in X/Y addressing, this register specifies the source start X in pixels. |

**(B)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Source Linear Start Address [27:24]** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Source Linear Start Address [23:16]** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Source Linear Start Address [15:8]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Source Linear Start Address [7:0]** | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [27:0] | **Source Linear Starting Address** | **28-bit Source Start Address**<br>For BitBLTs in linear addressing, this 28-bit byte address specifies the beginning location of the source. |

## Graphic Engine Destination Start XY/Linear Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEDSXY | 0xB000_B040 | R/W | Graphic Engine Destination Start X/Y addressing (pixel) | 0x0000_0000 |
| 2D_GEDSL | 0xB000_B040 | R/W | Graphic Engine Destination Start Linear address (byte) | 0x0000_0000 |

**(A)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Destination Start Y [10:8] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Destination Start Y [7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Destination Start X [10:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Destination Start X [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **Destination Start Y** | **11-bit Destination Start Y**<br>For BitBLTs and Bresenham line draw in X/Y addressing, this register specifies the destination start Y in pixels. |
| [10:0] | **Destination Start X** | **11-bit Destination Start X**<br>For BitBLTs and Bresenham line draw in X/Y addressing, this register specifies the destination start X in pixels. |

**(B)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Destination Linear Start Address [27:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Destination Linear Start Address [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Destination Linear Start Address [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Destination Linear Start Address [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [27:0] | **Destination Linear Starting Address** | **28-bit Destination Linear Starting Address**<br>For BitBLTs in linear addressing mode, this register specifies the destination linear starting address in bytes. |

## Graphic Engine Dimension for XY/Linear Modes Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEDIXYL | 0xB000_B044 | R/W | Graphic Engine Dimension in XY (pixel) or linear (byte) | 0x0000_0000 |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **Dimension Y** | **11-bit Dimension Y**<br>For BitBLTs, this register specifies the height of rectangle in X/Y addressing (by pixel) or in linear addressing (by byte). |
| [10:0] | **Dimension X** | **11-bit Dimension X**<br>For BitBLTs, this register specifies the width of rectangle in X/Y addressing (by pixel) or in linear addressing (by byte). |

## Graphic Engine Clipping Boundary Top/Left Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GECBTL | 0xB000_B048 | R/W | Graphic Engine Clipping Boundary Top/Left (by X/Y pixel) | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Clipping Boundary Top [10:8] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Clipping Boundary Top [7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Clipping Boundary Left [10:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Clipping Boundary Left [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **Clipping Boundary Top** | **11-bit Clipping Boundary Top**<br>This register specifies the top of the clipping rectangle. |
| [10:0] | **Clipping Boundary Left** | **11-bit Clipping Boundary Left**<br>This register specifies the left limit of the clipping rectangle. |

## Graphic Engine Clipping Boundary Bottom / Right Register

| Register | Address | R / W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GECBBR | 0xB000_B04C | R/W | Graphic Engine Clipping Boundary Bottom/Right (pixel) | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Clipping Boundary Bottom [10:8] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Clipping Boundary Bottom [7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Clipping Boundary Right [10:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Clipping Boundary Right [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [26:16] | **Clipping Boundary Bottom** | **11-bit Clipping Boundary Bottom**<br>This register specifies the bottom of the clipping rectangle. |
| [10:0] | **Clipping Boundary Right** | **11-bit Clipping Boundary Right**<br>This register specifies the right limit of the clipping rectangle. |

## Graphic Engine Pattern Group A Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEPTNA | 0xB000_B050 | R/W | Graphic Engine Pattern Group A | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Pattern3 | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Pattern2 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Pattern1 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Pattern0 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **Pattern Group A** | **Bits 31-24** Pattern 3 Register<br>When pattern is monochrome, this is the 4th line of the 8×8 pattern.<br>**Bits 23-16** Pattern 2 Register<br>When pattern is monochrome, this is the 3rd line of the 8×8 pattern.<br>**Bits 15-8** Pattern 1 Register<br>When pattern is monochrome, this is the 2nd line of the 8×8 pattern.<br>**Bits 7-0** Pattern 0 Register<br>When pattern is monochrome, this is the 1st line of the 8×8 pattern. |

## Graphic Engine Pattern Group B Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEPTNB | 0xB000_B054 | R/W | Graphic Engine Pattern Group B | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Pattern7 | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Pattern6 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Pattern5 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Pattern4 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **Pattern Group B** | **Bits 31-24** Pattern 7 Register<br>This is the 8th line of the 8×8 pattern.<br>**Bits 23-16** Pattern 6 Register<br>This is the 7th line of the 8×8 pattern.<br>**Bits 15-8** Pattern 5 Register<br>This is the 6th line of the 8×8 pattern.<br>**Bits 7-0** Pattern 4 Register<br>This is the 5th line of the 8×8 pattern. |

## Graphic Engine Write Plane Mask Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEWPM | 0xB000_B058 | R/W | Graphic Engine Write Plane Mask | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Write Plane Mask [23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Write Plane Mask [15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Write Plane Mask [7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | **Write Plane Mask** | **24-bit Write Plane Mask**<br>These bits specify which bits within each pixel are subject to update by the Graphics Engine. A one enable writing to the corresponding bit plane and a 0 inhibits writing to the corresponding bit plane. Only the corresponding number of bits-per-pixel in the display mode is required in the register. |

## Graphic Engine Miscellaneous Control Register

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| 2D_GEMC | 0xB000_B05C | R/W | Graphic Engine Miscellaneous Control | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| LINE STYLE PATTERN 1 / Alpha Blending Source Ks | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LINE STYLE PATTERN 0 / Alpha Blending Destination Kd | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FIFO STATUS | | | | EMPTY | FULL | BitBltSTS | BUSY |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RST_GE2D | RST_FIFO | BPP | | BLT_MD | BLT_TYPE | | |

| Bits | Descriptions | |
|---|---|---|
| [31:16] | **Line Style Pattern1**<br>**Line Style Pattern0** | **Bits 31-16**<br>16-bit line style pattern for Bresenham line drawing. |
| [31:16] | **Alpha Blending Ks and Kd** | **Bits 31-24**<br>**Bits 23-16**<br>8-bit alpha blending factor Ks for source data and<br>8-bit alpha blending factor Kd for destination data. |
| [15:12] | **FIFO Status** | **GE FIFO counter status**<br>0000 ~ 0111 = FIFO current level<br>0000 = empty and 1000 = full |
| [11] | **EMPTY** | **FIFO empty status**<br>0 = Not empty<br>1 = Empty |
| [10] | **FULL** | **FIFO full status**<br>0 = Not full<br>1 = Full |

| | | |
|---|---|---|
| [9] | **BitBLT_STS** | **GE BitBLT operation complete status**<br>0 = No complete status occur<br>1 = BitBLT operation complete status occur |
| [8] | **BUSY** | **GE Operation status**<br>0 = Ready, No GE operation<br>1 = Busy, GE operation is still under working |
| [7] | **RST_GE2D** | **Bit 7**<br>1 = Reset GE2D. |
| [6] | **RST_FIFO** | **Bit 6**<br>1 = Reset FIFO. |
| [5:4] | **Bit Per Pixel** | **Bits 5-4** Graphics Engine Pixel Depth<br>00 = 8-bit<br>01 = 16-bit<br>10 = 32-bit<br>11 = reserved |
| [3] | **BLT_MODE** | 0 = BitBLT type is according to GEC control bits<br>1 = BitBLT type follows BLT_TYPE[2:0] setting as below |
| [2:0] | **BLT_TYPE** | **Bits 2-0** BitBLT Type Setting<br>000 = HostBLT (write mode)<br>001 = HostBLT (read mode)<br>010 = SolidFillBLT<br>011 = PatternBLT<br>100 = BlockMoveBLT<br>101 = Color/Font Expansion BLT<br>110 = Monochrome Transparent BLT<br>111 = Color Transparent BLT |

## 7.13UART Controller

The Universal Asynchronous Receiver/Transmitter (UART) performs a serial-to-parallel conversion on data characters received from the peripheral, and a parallel-to-serial conversion on data characters received from the CPU. There are three UART blocks and accessory logic in this chip.

### 7.13.1 UART Feature Description

#### 7.13.1.1 UART0

UART0 is a general UART block without Modem I/O signals.

| UART0 | |
|---|---|
| **Clock Source** | External Crystal |
| **UART Type** | General UART |
| **FIFO Number** | 16-byte receiving FIFO and 16 byte transmitting FIFO |
| **Modem Function** | None |
| **Accessory Function** | None |
| **I/O pin** | TXD0, RXD0 |

#### 7.13.1.2 UART1

UART1 is a high speed UART. The FIFO has 64-byte for receiving and 64-byte for transmitting. The clock source is programmable in chip clock generator.

| UART1 | |
|---|---|
| **Clock Source** | External Crystal or internal PLL (Programmable) |
| **UART Type** | High speed UART |
| **FIFO Number** | 64-byte receiving FIFO and 64 byte transmitting FIFO |
| **Modem Function** | None |
| **Accessory Function** | None |
| **I/O pin** | TXD1, RXD1 |

#### 7.13.1.3 UART2

UART2 is a general UART with IrDA SIR.

| UART2 | |
|---|---|
| **Clock Source** | External Crystal |
| **UART Type** | General UART |
| **FIFO Number** | 16-byte receiving FIFO and 16 byte transmitting FIFO |
| **Modem Function** | none |
| **Accessory Function** | IrDA SIR |

| I/O pin | TXD2, RXD2 |
|---|---|

### 7.13.2 UART Control Registers Map

**R**: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

| Register | Offset | R/W | Description | Condition | Reset Value |
|---|---|---|---|---|---|
| **UART0 : UART_BA = 0xB800_0000** | | | | | |
| **RBR** | 0xB800_0100 | R | Receive Buffer Register | DLAB = 0 | Undefined |
| **THR** | 0xB800_0100 | W | Transmit Holding Register | DLAB = 0 | Undefined |
| **IER** | 0xB800_0104 | R/W | Interrupt Enable Register | DLAB = 0 | 0x0000_0000 |
| **DLL** | 0xB800_0100 | R/W | Divisor Latch Register (LS) | DLAB = 1 | 0x0000_0000 |
| **DLM** | 0xB800_0104 | R/W | Divisor Latch Register (MS) | DLAB = 1 | 0x0000_0000 |
| **IIR** | 0xB800_0108 | R | Interrupt Identification Register | | 0x8181_8181 |
| **FCR** | 0xB800_0108 | W | FIFO Control Register | | Undefined |
| **LCR** | 0xB800_010C | R/W | Line Control Register | | 0x0000_0000 |
| **LSR** | 0xB800_0114 | R | Line Status Register | | 0x6060_6060 |
| **TOR** | 0xB800_011C | R/W | Time Out Register | | 0x0000_0000 |
| **UART1 : UART_BA = 0xB800_0100** | | | | | |
| **RBR** | 0xB800_0100 | R | Receive Buffer Register | DLAB = 0 | Undefined |
| **THR** | 0xB800_0100 | W | Transmit Holding Register | DLAB = 0 | Undefined |
| **IER** | 0xB800_0104 | R/W | Interrupt Enable Register | DLAB = 0 | 0x0000_0000 |
| **DLL** | 0xB800_0100 | R/W | Divisor Latch Register (LS) | DLAB = 1 | 0x0000_0000 |
| **DLM** | 0xB800_0104 | R/W | Divisor Latch Register (MS) | DLAB = 1 | 0x0000_0000 |
| **IIR** | 0xB800_0108 | R | Interrupt Identification Register | | 0x8181_8181 |
| **FCR** | 0xB800_0108 | W | FIFO Control Register | | Undefined |
| **LCR** | 0xB800_010C | R/W | Line Control Register | | 0x0000_0000 |
| **LSR** | 0xB800_0114 | R | Line Status Register | | 0x6060_6060 |
| **TOR** | 0xB800_011C | R/W | Time Out Register | | 0x0000_0000 |

| | | | | | |
|---|---|---|---|---|---|
| **UART2 : UART_BA = 0xB800_0200** | | | | | |
| **RBR** | 0xB800_0200 | R | Receive Buffer Register | DLAB = 0 | Undefined |
| **THR** | 0xB800_0200 | W | Transmit Holding Register | DLAB = 0 | Undefined |

| | | | | | |
|---|---|---|---|---|---|
| **IER** | 0xB800_0204 | R/W | Interrupt Enable Register | DLAB = 0 | 0x0000_0000 |
| **DLL** | 0xB800_0200 | R/W | Divisor Latch Register (LS) | DLAB = 1 | 0x0000_0000 |
| **DLM** | 0xB800_0204 | R/W | Divisor Latch Register (MS) | DLAB = 1 | 0x0000_0000 |
| **IIR** | 0xB800_0208 | R | Interrupt Identification Register | | 0x8181_8181 |
| **FCR** | 0xB800_0208 | W | FIFO Control Register | | Undefined |
| **LCR** | 0xB800_020C | R/W | Line Control Register | | 0x0000_0000 |
| **LSR** | 0xB800_0214 | R | Line Status Register | | 0x6060_6060 |
| **TOR** | 0xB800_021C | R/W | Time Out Register | | 0x0000_0000 |
| **IRCR** | 0xB800_0220 | R/W | IrDA Control Register | | 0x0000_0040 |

### Receive Buffer Register (RBR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RBR** | 0XB800_0x00 | R | Receive Buffer Register (DLAB = 0) | Undefined |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **8-bit Received Data** | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [7:0] | **8-bit Received Data** | By reading this register, the UART will return an 8-bit data received from SIN pin (LSB first). |

### Transmit Holding Register (THR)

| Register | offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| THR | 0XB800_0x00 | W | Transmit Holding Register (DLAB = 0) | Undefined |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 8-bit Transmitted Data | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [7:0] | **8-bit Transmitted Data** | By writing to this register, the UART will send out an 8-bit data through the SOUT pin (LSB first). |

## Interrupt Enable Register (IER)

| Register | offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| IER | 0XB800_0x04 | R/W | Interrupt Enable Register (DLAB = 0) | 0x0000.0000 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | MSIE | RLSIE | THREIE | RDAIE |

| Bits | Descriptions | |
|---|---|---|
| [3] | **MSIE** | **MODEM Status Interrupt (Irpt_MOS) Enable**<br>● 0 = Mask off Irpt_MOS<br>● 1 = Enable Irpt_MOS |
| [2] | **RLSIE** | **Receive Line Status Interrupt (Irpt_RLS) Enable**<br>● 0 = Mask off Irpt_RLS<br>● 1 = Enable Irpt_RLS |
| [1] | **THREIE** | **Transmit Holding Register Empty Interrupt (Irpt_THRE) Enable**<br>● 0 = Mask off Irpt_THRE<br>● 1 = Enable Irpt_THRE |
| [0] | **RDAIE** | **R**eceive **D**ata **A**vailable **I**nterrupt **(Irpt_RDA) E**nable and Time-out Interrupt **(Irpt_TOUT)** Enable<br>● 0 = Mask off Irpt_RDA and Irpt_TOUT<br>● 1 = Enable Irpt_RDA and Irpt_TOUT |

### Divider Latch (Low Byte) Register (DLL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| DLL | 0XB800_0x00 | R/W | Divisor Latch Register (LS) (DLAB = 1) | 0x0000_0000 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Baud Rate Divider (Low Byte) | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [7:0] | **Baud Rate Divisor (Low Byte)** | The low byte of the baud rate divider |

### Divisor Latch (High Byte) Register (DLM)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **DLM** | 0XB800_0x04 | R/W | Divisor Latch Register (MS) (DLAB = 1) | 0x0000_0000 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **Baud Rate Divider (High Byte)** | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [7:0] | **Baud Rate Divisor (High Byte)** | The high byte of the baud rate divider |

This 16-bit divider {DLM, DLL} is used to determine the baud rate as follows

**Baud Rate = Crystal Clock / {16 * [Divisor + 2]}**

Note: This definition is different from 16550

## Interrupt Identification Register (IIR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| IIR | 0XB800_0x08 | R | Interrupt Identification Register | 0x8181_8181 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FMES | RFTLS | | DMS | IID | | | NIP |

| Bits | Descriptions | |
|---|---|---|
| [7] | FMES | **FIFO Mode Enable Status**<br>This bit indicates whether the FIFO mode is enabled or not. Since the FIFO mode is always enabled, this bit always shows the logical 1 when CPU is reading this register. |
| [6:5] | RFTLS | **Rx FIFO Threshold Level Status**<br>These bits show the current setting of receiver FIFO threshold level (RTHO). The meaning of RTHO is defined in the following FCR description. |
| [4] | DMS | **DMA Mode Select**<br>The DMA function is not implemented in this version. When reading IIR, the DMS is always returned 0. |
| [3:1] | IID | **Interrupt Identification**<br>The IID together with NIP indicates the current interrupt request from UART. |
| [0] | NIP | **No Interrupt Pending**<br>There is no pending interrupt. |

## Interrupt Control Functions

| IIR [3:0] | Priority | Interrupt Type | Interrupt Source | Interrupt Reset control |
|---|---|---|---|---|
| ---1 | -- | None | None | -- |
| 0110 | Highest | Receiver Line Status (Irpt_RLS) | Overrun error, parity error, framing error, or break interrupt | Reading the LSR |

| IIR [3:0] | Priority | Interrupt Type | Interrupt Source | Interrupt Reset control |
|---|---|---|---|---|
| 0100 | Second | Received Data Available (Irpt_RDA) | Receiver FIFO threshold level is reached | Receiver FIFO drops below the threshold level |
| 1100 | Second | Receiver FIFO Time-out (Irpt_TOUT) | Receiver FIFO is non-empty and no activities are occurred in the receiver FIFO during the TOR defined time duration | Reading the RBR |
| 0010 | Third | Transmitter Holing Register Empty (Irpt_THRE) | Transmitter holding register empty | Reading the IIR (if source of interrupt is Irpt_THRE) or writing into the THR |
| 0000 | Fourth | MODEM Status (Irpt_MOS) | The CTS bits are changing state. | Reading the MSR **(optional)** |

Note: These definitions of bit 7, bit 6, bit 5, and bit 4 are different from the 16550.

## FIFO Control Register (FCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FCR | 0XB800_0x08 | W | FIFO Control Register | Undefined |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RFITL | | | | DMS | TFR | RFR | FME |

| Bits | Descriptions | |
|---|---|---|
| [7:4] | RFITL | **Rx FIFO Interrupt (Irpt_RDA) Trigger Level**<br>UART0 / UART2:<br>RFITL [7:4] / Trigger Level<br>00xx / 01 bytes<br>01xx / 04 bytes<br>10xx / 08 bytes<br>11xx / 14 bytes<br>UART1:<br>RFITL[7:4] / Trigger Level<br>0000 / 01 bytes<br>0001 / 04 bytes<br>0010 / 08 bytes<br>0011 / 14 bytes<br>0100 / 30 bytes<br>0101 / 46 bytes<br>others / 62 bytes |
| [3] | DMS | **DMA Mode Select**<br>The DMA function is not implemented in this version. |
| [2] | TFR | **Tx FIFO Reset**<br>Setting this bit will generate an OSC cycle reset pulse to reset Tx FIFO. The Tx FIFO becomes empty (Tx pointer is reset to 0) after such reset. This bit is returned to 0 automatically after the reset pulse is generated. |
| [1] | RFR | **Rx FIFO Reset**<br>Setting this bit will generate an OSC cycle reset pulse to reset Rx FIFO. The Rx FIFO becomes empty (Rx pointer is reset to 0) after such reset. This bit is returned to 0 automatically after the reset pulse is generated. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **FME** | **FIFO Mode Enable**<br>Because UART is always operating in the FIFO mode, writing this bit has no effect while reading always gets logical one. This bit must be 1 when other FCR bits are written to; otherwise, they will not be programmed. |

## Line Control Register (LCR)

| Register | offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| LCR | 0XB800_0x0C | R/W | Line Control Register | 0x0000_0000 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DLAB | BCB | SPE | EPE | PBE | NSB | WLS | |

| Bits | Descriptions | |
|---|---|---|
| [7] | **DLAB** | **Divider Latch Access Bit**<br>0 = It is used to access RBR, THR or IER.<br>1 = It is used to access Divisor Latch Registers {DLL, DLM}. |
| [6] | **BCB** | **Break Control Bit**<br>When this bit is set to logic 1, the serial data output (SOUT) is forced to the Spacing State (logic 0). This bit acts only on SOUT and has no effect on the transmitter logic. |
| [5] | **SPE** | **Stick Parity Enable**<br>0 = Disable stick parity<br>1 = Parity bit is transmitted and checked as a logic 1 if bit 4 is 0 (odd parity), or as a logic 0 if bit 4 is 1 (even parity). This bit has effect only when bit 3 (parity bit enable) is set. |
| [4] | **EPE** | **Even Parity Enable**<br>0 = Odd number of logic 1's are transmitted or checked in the data word and parity bits.<br>1 = Even number of logic 1's are transmitted or checked in the data word and parity bits.<br>This bit has effect only when bit 3 (parity bit enable) is set. |
| [3] | **PBE** | **Parity Bit Enable**<br>0 = Parity bit is not generated (transmit data) or checked (receive data) during transfer.<br>1 = Parity bit is generated or checked between the "last data word bit" and "stop bit" of the serial data. |

| Bits | | Descriptions |
|---|---|---|
| [2] | **NSB** | **Number of "STOP bit"**<br>0= One " STOP bit" is generated in the transmitted data<br>1= One and a half " STOP bit" is generated in the transmitted data when 5-bit word length is selected;<br>Two "STOP bit" is generated when 6-, 7- and 8-bit word length is selected. |
| [1:0] | **WLS** | **Word Length Select**<br><br>**WLS[1:0]** / **Character length**<br>00 / 5 bits<br>01 / 6 bits<br>10 / 7 bits<br>11 / 8 bits |

## Line Status Control Register (LSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| LSR | 0XB800_0x14 | R | Line Status Register | 0x6060_6060 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ERR_Rx | TE | THRE | BII | FEI | PEI | OEI | RFDR |

| Bits | Descriptions | |
|---|---|---|
| [7] | **ERR_Rx** | **Rx FIFO Error**<br>0 = Rx FIFO works normally<br>1 = There is at least one parity error (PE), framing error (FE), or break indication (BI) in the FIFO. ERR_Rx is cleared when CPU reads the LSR and if there are no subsequent errors in the Rx FIFO. |
| [6] | **TE** | **Transmitter Empty**<br>0 = Either Transmitter Holding Register (**THR** - Tx FIFO) or Transmitter Shift Register (**TSR**) are not empty.<br>1 = Both THR and TSR are empty. |
| [5] | **THRE** | **Transmitter Holding Register Empty**<br>0 = THR is not empty.<br>1 = THR is empty.<br>THRE is set when the last data word of Tx FIFO is transferred to Transmitter Shift Register (TSR). The CPU resets this bit when the THR (or Tx FIFO) is loaded. This bit also causes the UART to issue an interrupt (Irpt_THRE) to the CPU when IER [1] =1. |
| [4] | **BII** | **Break Interrupt Indicator**<br>This bit is set to a logic 1 whenever the received data input is held in the "spacing state" (logic 0) for longer than a full word transmission time (that is, the total time of "start bit" + data bits + parity + stop bits) and is reset whenever the CPU reads the contents of the LSR. |
| [3] | **FEI** | **Framing Error Indicator**<br>This bit is set to logic 1 whenever the received character does not have a valid "stop bit" (that is, the stop bit following the last data bit or parity bit is detected as a logic 0), and is reset whenever the CPU reads the contents of the LSR. |

| Bits | Descriptions | |
|---|---|---|
| [2] | **PEI** | **Parity Error Indicator**<br>This bit is set to logic 1 whenever the received character does not have a valid "parity bit", and is reset whenever the CPU reads the contents of the LSR. |
| [1] | **OEI** | **Overrun Error Indicator**<br>An overrun error will occur only after the Rx FIFO is full and the next character has been completely received in the shift register. The character in the shift register is overwritten, but it is not transferred to the Rx FIFO. OE is indicated to the CPU as soon as it happens and is reset whenever the CPU reads the contents of the LSR. |
| [0] | **RFDR** | **Rx FIFO Data Ready**<br>0 = Rx FIFO is empty<br>1 = Rx FIFO contains at least 1 received data word. |

LSR [4:2] (BII, FEI, PEI) are revealed to the CPU when its associated character is at the top of the Rx FIFO. These three error indicators are reset whenever the CPU reads the contents of the LSR.

LSR [4:1] (BII, FEI, PEI, OEI) are the error conditions that produce a "receiver line status interrupt" (Irpt_RLS) when IER [2] =1. Reading LSR clears Irpt_RLS. Writing LSR is a null operation (not suggested).

## Time-Out Register (TOR)

| Register | offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| TOR | 0XB800_0x1C | R/W | Time Out Register | 0x0000_0000 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TOIE | TOIC | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [7] | TOIE | **Time Out Interrupt Enable**<br>The feature of receiver time out interrupt is enabled only when TOR [7] = IER [0] = 1. |
| [6:0] | TOIC | **Time Out Interrupt Comparator**<br>The time out counter resets and starts counting (the counting clock = baud rate) whenever the Rx FIFO receives a new data word. Once the content of time out counter (TOUT_CNT) is equal to that of time out interrupt comparator (TOIC), a receiver time out interrupt (Irpt_TOUT) is generated if TOR [7] = IER [0] = 1. A new incoming data word or Rx FIFO empty clears Irpt_TOUT. |

### IrDA Control Register (IRCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| IRCR | 0xB800_0220 | R/W | IrDA Control Register for UART2 | 0x0000_0040 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | INV_Rx | INV_Tx | Reserved | | | Tx_SELECT | IrDA_EN |

| Bits | Descriptions | |
|---|---|---|
| [6] | INV_Rx | **INV_Rx**<br>1: Inverse Rx input signal<br>0: No inversion |
| [5] | INV_Tx | **INV_Tx**<br>1: Inverse Tx output signal<br>0: No inversion |
| [1] | Tx_SELECT | **Tx_SELECT**<br>1: Enable IrDA transmitter<br>0: Enable IrDA receiver |
| [0] | IrDA_EN | **IrDA_EN**<br>1: Enable IrDA block<br>0: Disable IrDA block |

## 7.14 TIMER Controller

### 7.14.1 General Timer Controller

The timer module includes five channels, TIMER0~TIMER4, they can easily be implemented as counting scheme. The timer can perform functions like frequency measurement, event counting, interval

measurement, pulse generation, delay timing, and so on. The timer possesses features such as adjustable resolution, programmable counting period, and detailed information. The timer can generate an interrupt signal upon timeout, or provide the current value of count during operation.

The general TIMER Controller includes the following features

- Five channels with a 24-bit down counter and an interrupt request each
- Independent clock source for each channel
- Maximum uninterrupted time = (1 / 15 MHz) * (255) * (2^24 - 1), if TCLK = 15 MHz

### 7.14.2 Watchdog Timer

The purpose of watchdog timer is to perform a system restart after the software running into a problem. This prevents system from hanging for an indefinite period of time. It is a free running timer with programmable time-out intervals. When the specified time internal expires, a system reset can be generated. If the watchdog timer reset function is enabled and the watchdog timer is not being reset before timing out, then the watchdog reset is activated after 1024 WDT clocks. Setting **WTE** in the register **WTCR** enables the watchdog timer.

The **WTR** should be set before making use of watchdog timer. This ensures that the watchdog timer restarts from a know state. The watchdog timer will start counting and time-out after a specified period of time. The time-out interval is selected by two bits, **WTIS [1:0]**. The **WTR** is self-clearing, i.e., after setting it, the hardware will automatically reset it. When timeout occurs, Watchdog Timer interrupt flag is set. Watchdog Timer waits for an additional **1024 WDT clock cycles** before issuing a reset signal, if the **WTRE** is set. The **WTRF** will be set and the reset signal will last for **15 WDT clock cycles** long. When used as a simple timer, the interrupt and reset functions are disabled. Watchdog Timer will set the **WTIF** each time a timeout occurs. The **WTIF** can be polled to check the status, and software can restart the timer by setting the **WTR**.

### 7.14.3 Timer Control Registers Map

**R**: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

| Register | Address | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| **TMR_BA = 0xB800_1000** | | | | |
| **TCSR0** | 0xB800_1000 | R/W | Timer Control and Status Register 0 | 0x0000_0005 |
| **TCSR1** | 0xB800_1004 | R/W | Timer Control and Status Register 1 | 0x0000_0005 |
| **TICR0** | 0xB800_1008 | R/W | Timer Initial Control Register 0 | 0x0000_0000 |
| **TICR1** | 0xB800_100C | R/W | Timer Initial Control Register 1 | 0x0000_0000 |
| **TDR0** | 0xB800_1010 | R | Timer Data Register 0 | 0x0000_0000 |
| **TDR1** | 0xB800_1014 | R | Timer Data Register 1 | 0x0000_0000 |
| **TISR** | 0xB800_1018 | R/C | Timer Interrupt Status Register | 0x0000_0000 |
| **WTCR** | 0xB800_101C | R/W | Watchdog Timer Control Register | 0x0000_0400 |
| **TCSR2** | 0xB800_1020 | R/W | Timer Control and Status Register 2 | 0x0000_0005 |
| **TCSR3** | 0xB800_1024 | R/W | Timer Control and Status Register 3 | 0x0000_0005 |
| **TICR2** | 0xB800_1028 | R/W | Timer Initial Control Register 2 | 0x0000_0000 |
| **TICR3** | 0xB800_102C | R/W | Timer Initial Control Register 3 | 0x0000_0000 |
| **TDR2** | 0xB800_1030 | R | Timer Data Register 2 | 0x0000_0000 |
| **TDR3** | 0xB800_1034 | R | Timer Data Register 3 | 0x0000_0000 |
| **TCSR4** | 0xB800_1040 | R/W | Timer Control and Status Register 4 | 0x0000_0005 |
| **TICR4** | 0xB800_1048 | R/W | Timer Initial Control Register 4 | 0x0000_0000 |
| **TDR4** | 0xB800_1050 | R | Timer Data Register 4 | 0x0000_0000 |

## Timer Control and Status Register 0~4 (TCR0~TCR4)

| Register | Address | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| **TCSR0** | 0xB800_1000 | R/W | Timer Control and Status Register 0 | 0x0000_0005 |
| **TCSR1** | 0xB800_1004 | R/W | Timer Control and Status Register 1 | 0x0000_0005 |
| **TCSR2** | 0xB800_1020 | R/W | Timer Control and Status Register 2 | 0x0000_0005 |
| **TCSR3** | 0xB800_1024 | R/W | Timer Control and Status Register 3 | 0x0000_0005 |
| **TCSR4** | 0xB800_1040 | R/W | Timer Control and Status Register 4 | 0x0000_0005 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | CE | IE | MODE | | CRST | CACT | RESERVED |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PRESCALE | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [30] | **CE** | **Counter Enable**<br>0 = Stops counting<br>1 = Starts counting |
| [29] | **IE** | **Interrupt Enable**<br>0 = Disables timer interrupt<br>1 = Enables timer interrupt. If timer interrupt is enabled, the timer asserts its interrupt signal when the associated counter decrements to zero. |

<table>
<tr><td>[28:27]</td><td>MODE</td><td>Timer Operating Mode
<table>
<tr><th>MODE [28:27]</th><th>Timer Operating Mode</th></tr>
<tr><td>00</td><td>The timer is operating in the one-shot mode. The associated interrupt signal is generated once (if IE is enabled) and CE is automatically cleared then.</td></tr>
<tr><td>01</td><td>The timer is operating in the periodic mode. The associated interrupt signal is generated periodically (if IE is enabled).</td></tr>
<tr><td>10</td><td>The timer is operating in the toggle mode. The associated interrupt signal is changing back and forth (if IE is enabled) with 50% duty cycle.</td></tr>
<tr><td>11</td><td>Reserved for further use</td></tr>
</table></td></tr>
<tr><td>[26]</td><td>CRST</td><td>Counter Reset<br>Set this bit will reset the TIMER counter, and also force CEN to 0.<br>0 = No effect.<br>1 = Reset Timer’s pre-scale counter, internal 24-bit counter and CEN.</td></tr>
<tr><td>[25]</td><td>CACT</td><td>Timer is in Active<br>This bit indicates the counter status of timer.<br>0 = Timer is not active.<br>1 = Timer is in active.</td></tr>
<tr><td>[7:0]</td><td>PRESCALE</td><td>Clock Pre-scale Divide Count<br>Clock input is divided by PRESCALE + 1 before it is fed to the counter (here PRESCALE is considered as a decimal number). If PRESCALE = 0, then there is no scaling.</td></tr>
</table>

## Timer Initial Count Register 0~4 (TICR0~TICR4)

| Register | Address | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| **TICR0** | 0xB800_1008 | R/W | Timer Initial Control Register 0 | 0x0000_00FF |
| **TICR1** | 0xB800_100C | R/W | Timer Initial Control Register 1 | 0x0000_00FF |
| **TICR2** | 0xB800_1028 | R/W | Timer Initial Control Register 2 | 0x0000_00FF |
| **TICR3** | 0xB800_102C | R/W | Timer Initial Control Register 3 | 0x0000_00FF |
| **TICR4** | 0xB800_1048 | R/W | Timer Initial Control Register 4 | 0x0000_00FF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TIC[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TIC[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TIC[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | **TIC** | **Timer Initial Count**<br>This is a 24-bit value representing the initial count. Timer will reload this value whenever the counter is decremented to zero.<br>**NOTE:**<br>(1) Never write 0x0 in TIC, or the core will run into unknown state.<br>(2) No matter CEN is 0 or 1, whenever software write a new value into this register, Timer will restart counting using this new value and abort previous count. |

### Timer Data Register 0~4 (TDR0~TDR4)

| Register | Address | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| **TDR0** | 0xB800_1010 | R | Timer Data Register 0 | 0x0000_00FF |
| **TDR1** | 0xB800_1014 | R | Timer Data Register 1 | 0x0000_00FF |
| **TDR2** | 0xB800_1030 | R | Timer Data Register 2 | 0x0000_00FF |
| **TDR3** | 0xB800_1034 | R | Timer Data Register 3 | 0x0000_00FF |
| **TDR4** | 0xB800_1050 | R | Timer Data Register 4 | 0x0000_00FF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TDR[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TDR[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TDR[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [23:0] | **TDR** | **Timer Data Register**<br>The current count is registered in this 24-bit value.<br>**NOTE:**<br>Software can read a correct current value on this register only when **CEN = 0**, or the value represents here could not be a correct one. |

## Timer Interrupt Status Register (TISR)

| Register | Address | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| TISR | 0xB800_1018 | R/C | Timer Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | TIF4 | TIF3 | TIF2 | TIF1 | TIF0 |

| Bits | Descriptions | |
|---|---|---|
| [4] | TIF4 | **Timer Interrupt Flag 4**<br>0 = It indicates that the timer 4 does not count down to zero yet. Software can reset this bit after the timer interrupt 4 had occurred.<br>1 = It indicates that the counter of timer 4 is decremented to zero;<br>**NOTE**: This bit is read only, but can be cleared by writing 1 to this bit. |
| [3] | TIF3 | **Timer Interrupt Flag 3**<br>0 = It indicates that the timer 3 does not count down to zero yet. Software can reset this bit after the timer interrupt 3 had occurred.<br>1 = It indicates that the counter of timer 3 is decremented to zero;<br>**NOTE**: This bit is read only, but can be cleared by writing 1 to this bit. |
| [2] | TIF2 | **Timer Interrupt Flag 2**<br>0 = It indicates that the timer 2 does not count down to zero yet. Software can reset this bit after the timer interrupt 2 had occurred.<br>1 = It indicates that the counter of timer 2 is decremented to zero;<br>**NOTE**: This bit is read only, but can be cleared by writing 1 to this bit. |

| | | |
|---|---|---|
| [1] | **TIF1** | **Timer Interrupt Flag 1**<br>0 = It indicates that the timer 1 does not count down to zero yet. Software can reset this bit after the timer interrupt 1 had occurred.<br>1 = It indicates that the counter of timer 1 is decremented to zero;<br>**NOTE**: This bit is read only, but can be cleared by writing 1 to this bit. |
| [0] | **TIF0** | **Timer Interrupt Flag 0**<br>0 = It indicates that the timer 0 does not count down to zero yet. Software can reset this bit after the timer interrupt 0 had occurred.<br>1 = It indicates that the counter of timer 0 is decremented to zero;<br>**NOTE**: This bit is read only, but can be cleared by writing 1 to this bit. |

## Watchdog Timer Control Register (WTCR)

| Register | Address | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| WTCR | 0xB800_101C | R/W | Watchdog Timer Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | WTCLK | RESERVED | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WTE | WTIE | WTIS | | WTIF | WTRF | WTRE | WTR |

| Bits | Descriptions | |
|---|---|---|
| [10] | **WTCLK** | **Watchdog Timer Clock**<br>This bit is used for deciding whether the Watchdog timer clock input is divided by 256 or not. Clock source of Watchdog timer is Crystal input.<br>0 = Using original clock input<br>1 = The clock input will be divided by 256<br>**NOTE:** When WTTME = 1, set this bit has no effect on WDT clock (using original clock input). |
| [7] | **WTE** | **Watchdog Timer Enable**<br>0 = Disable the watchdog timer<br>1 = Enable the watchdog timer |
| [6] | **WTIE** | **Watchdog Timer Interrupt Enable**<br>0 = Disable the watchdog timer interrupt<br>1 = Enable the watchdog timer interrupt |

<table>
<tr>
<td>[5:4]</td>
<td>WTIS</td>
<td>Watchdog Timer Interval Select<br>These two bits select the interval for the watchdog timer. No matter which interval is chosen, the reset time-out is always occurred 1024 clocks later than the interrupt time-out.
<table>
<tr><th>WTIS</th><th>Interrupt Timeout</th><th>Reset Timeout</th><th>Real Time Interval (CLK=15MHz/256)</th></tr>
<tr><td>00</td><td>$2^{14}$ clocks</td><td>$2^{14}$ + 1024 clocks</td><td>0.28 sec.</td></tr>
<tr><td>01</td><td>$2^{16}$ clocks</td><td>$2^{16}$ + 1024 clocks</td><td>1.12 sec.</td></tr>
<tr><td>10</td><td>$2^{18}$ clocks</td><td>$2^{18}$ + 1024 clocks</td><td>4.47 sec.</td></tr>
<tr><td>11</td><td>$2^{20}$ clocks</td><td>$2^{20}$ + 1024 clocks</td><td>17.9 sec.</td></tr>
</table>
</td>
</tr>
<tr>
<td>[3]</td>
<td>WTIF</td>
<td>Watchdog Timer Interrupt Flag<br>If the watchdog interrupt is enabled, then the hardware will set this bit to indicate that the watchdog interrupt has occurred. If the watchdog interrupt is not enabled, then this bit indicates that a time-out period has elapsed.<br>0 = Watchdog timer interrupt does not occur<br>1 = Watchdog timer interrupt occurs<br>NOTE: This bit is read only, but can be cleared by writing 1 to this bit.</td>
</tr>
<tr>
<td>[2]</td>
<td>WTRF</td>
<td>Watchdog Timer Reset Flag<br>When the watchdog timer initiates a reset, the hardware will set this bit. This flag can be read by software to determine the source of reset. Software is responsible to clear it up manually. If WTRE is disabled, then the watchdog timer has no effect on this bit.<br>0 = Watchdog timer reset does not occur<br>1 = Watchdog timer reset occurs</td>
</tr>
<tr>
<td>[1]</td>
<td>WTRE</td>
<td>Watchdog Timer Reset Enable<br>Setting this bit will enable the watchdog timer reset function.<br>0 = Disable watchdog timer reset function<br>1 = Enable watchdog timer reset function<br>NOTE: This bit is read only, but can be cleared by writing 1 to this bit.</td>
</tr>
</table>

| | | |
|---|---|---|
| [0] | **WTR** | **Watchdog Timer Reset**<br>This bit brings the watchdog timer into a known state. It helps reset the watchdog timer before a time-out situation occurring. Failing to set **WTR** before time-out will initiates an interrupt if **WTIE** is set. If **WTRE** is set, a watchdog timer reset will be generated 512 clocks after time-out. This bit is self-clearing.<br>0 = No operation<br>1 = Reset the contents of the watchdog timer |

## 7.15 Advanced Interrupt Controller

An *interrupt* temporarily changes the sequence of program execution to react to a particular event such as power failure, watchdog timer timeout, transmit/receive request from Ethernet MAC Controller, and so on. The CPU processor provides two modes of interrupt, the **Fast Interrupt (FIQ)** mode for critical session and the *Interrupt* **(IRQ)** mode for general purpose. The IRQ request is occurred when the nIRQ input is asserted. Similarly, the FIQ request is occurred when the nFIQ input is asserted. The FIQ has privilege over the IRQ and can preempt an ongoing IRQ. It is possible to ignore the FIQ and the IRQ by setting the F and I bits in the **current program status register (CPSR)**.

The **Advanced Interrupt Controller (AIC)** is capable of processing the interrupt requests up to 32 different sources. Currently, 23 interrupt sources are defined. Each interrupt source is uniquely assigned to an *interrupt channel*. For example, the watchdog timer interrupt is assigned to channel 1. The AIC implements a proprietary eight-level priority scheme that categories the available 23 interrupt sources into eight priority levels. Interrupt sources within the priority level 0 is the highest priority and the priority level 7 is the lowest. In order to make this scheme work properly, a certain priority level must be specified to each interrupt source during power-on initialization; otherwise, the system shall behave unexpectedly. Within each priority level, interrupt source that is positioned in a lower channel has a higher priority. Interrupt source that is active, enabled, and positioned in the lowest channel with priority level 0 is promoted to the FIQ. Interrupt sources within the priority levels other than 0 are routed to the IRQ. The IRQ can be preempted by the occurrence of the FIQ. Interrupt nesting is performed automatically by the AIC.

Though interrupt sources originated from the chip itself are intrinsically high-level sensitive, the AIC can be configured as either low-level sensitive, high-level sensitive, negative-edge triggered, or positive-edge triggered to each interrupt source.

The advanced interrupt controller includes the following features:

- External interrupts can be programmed as either edge-triggered or level-sensitive
- External interrupts can be programmed as either low-active or high-active
- Flags to reflect the status of each interrupt source
- Individual mask for each interrupt source
- Proprietary 8-level interrupt scheme to employ the priority scheme.
- Priority methodology is adopted to allow for interrupt daisy-chaining
- Automatically masking out the lower priority interrupt during interrupt nesting
- Automatically clearing the interrupt flag when the external interrupt source is programmed to be edge-triggered

| Priority | Name | Mode | Source |
|---|---|---|---|
| 1 (Highest) | WDT_INT | Positive Level | Watch Dog Timer Interrupt |
| 2 | nIRQ_Group0 | Positive Level | External Interrupt Group 0 |
| 3 | nIRQ_Group1 | Positive Level | ICE COMMTX/RX Interrupt |
| 4 | ACTL_INT | Positive Level | Audio Controller Interrupt |
| 5 | LCD_INT | Positive Level | LCD Controller Interrupt |
| 6 | --- | --- | Reserved |
| 7 | UART_INT0 | Positive Level | UART Interrupt0 |
| 8 | UART_INT1 | Positive Level | UART Interrupt1 |
| 9 | UART_INT2 | Positive Level | UART Interrupt2 |
| 10 | --- | --- | Reserved |
| 11 | --- | --- | Reserved |
| 12 | T_INT0 | Positive Level | Timer Interrupt 0 |
| 13 | T_INT1 | Positive Level | Timer Interrupt 1 |
| 14 | T_INT_Group | Positive Level | Timer Interrupt Group |
| 15 | USBH_INT_Group | Positive Level | USB Host Interrupt Group |
| 16 | EMCTx_INT | Positive Level | EMC Tx Interrupt |
| 17 | EMCRx_INT | Positive Level | EMC Rx Interrupt |
| 18 | GDMA_INT_Group | Positive Level | GDMA Interrupt Group |
| 19 | DMAC_INT | Positive Level | DMAC Interrupt |
| 20 | FMI_INT | Positive Level | FMI Interrupt |
| 21 | USBD_INT | Positive Level | USB Device Interrupt |
| 22 | --- | --- | Reserved |
| 23 | G2D_INT | Positive Level | 2D Graphic Engine Interrupt |
| 24 | --- | --- | Reserved |
| 25 | --- | --- | Reserved |
| 26 | I2C_INT_Group | Positive Level | I2C Interrupt Group |
| 27 | USI_INT | Positive Level | USI Interrupt |
| 28 | PWM _INT | Positive Level | PWM Timer Interrupt |
| 29 | KPI_INT | Positive Level | Keypad Interrupt |
| 30 | --- | --- | Reserved |
| 31 | --- | --- | Reserved |

| Interrupt Group | Interrupt Sources |
|---|---|
| **External Interrupt Group 0** | External Pins : nIRQ[2:0] |
| **External Interrupt Group 1** | ICE Signals : COMMRX,COMMTX |
| **Timer Interrupt Group** | TIMER2, TIMER3, and TIMER4 |
| **USB Host Interrupt Group** | OHCI and EHCI USB Host Controller |
| **GDMA Interrupt Group** | GDMA0 and GDMA1 |
| **I2C interrupt Group** | I2C Line 0 and I2C Line 1 |

### 7.15.1 AIC Registers Map

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_BA = 0xB800_2000** | | | | |
| **AIC_SCR1** | 0xB800_2004 | R/W | Source Control Register 1 | 0x0000_0047 |
| **AIC_SCR2** | 0xB800_2008 | R/W | Source Control Register 2 | 0x0000_0047 |
| **AIC_SCR3** | 0xB800_200C | R/W | Source Control Register 3 | 0x0000_0047 |
| **AIC_SCR4** | 0xB800_2010 | R/W | Source Control Register 4 | 0x0000_0047 |
| **AIC_SCR5** | 0xB800_2014 | R/W | Source Control Register 5 | 0x0000_0047 |
| **AIC_SCR7** | 0xB800_201C | R/W | Source Control Register 7 | 0x0000_0047 |
| **AIC_SCR8** | 0xB800_2020 | R/W | Source Control Register 8 | 0x0000_0047 |
| **AIC_SCR9** | 0xB800_2024 | R/W | Source Control Register 9 | 0x0000_0047 |
| **AIC_SCR12** | 0xB800_2030 | R/W | Source Control Register 12 | 0x0000_0047 |
| **AIC_SCR13** | 0xB800_2034 | R/W | Source Control Register 13 | 0x0000_0047 |
| **AIC_SCR14** | 0xB800_2038 | R/W | Source Control Register 14 | 0x0000_0047 |
| **AIC_SCR15** | 0xB800_203C | R/W | Source Control Register 15 | 0x0000_0047 |
| **AIC_SCR16** | 0xB800_2040 | R/W | Source Control Register 16 | 0x0000_0047 |
| **AIC_SCR17** | 0xB800_2044 | R/W | Source Control Register 17 | 0x0000_0047 |
| **AIC_SCR18** | 0xB800_2048 | R/W | Source Control Register 18 | 0x0000_0047 |
| **AIC_SCR19** | 0xB800_204C | R/W | Source Control Register 19 | 0x0000_0047 |
| **AIC_SCR20** | 0xB800_2050 | R/W | Source Control Register 20 | 0x0000_0047 |
| **AIC_SCR21** | 0xB800_2054 | R/W | Source Control Register 21 | 0x0000_0047 |
| **AIC_SCR23** | 0xB800_205C | R/W | Source Control Register 23 | 0x0000_0047 |
| **AIC_SCR26** | 0xB800_2068 | R/W | Source Control Register 26 | 0x0000_0047 |

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_SCR27** | 0xB800_206C | R/W | Source Control Register 27 | 0x0000_0047 |
| **AIC_SCR28** | 0xB800_2070 | R/W | Source Control Register 28 | 0x0000_0047 |
| **AIC_SCR29** | 0xB800_2074 | R/W | Source Control Register 29 | 0x0000_0047 |
| **AIC_IRQSC** | 0xB800_2080 | R/W | External Interrupt Control Register | 0x0000_0000 |
| **AIC_GEN** | 0xB800_2084 | R/W | Interrupt Group Enable Control Register | 0x0000_0000 |
| **AIC_GASR** | 0xB800_2088 | R | Interrupt Group Active Status Register | 0x0000_0000 |
| **AIC_GSCR** | 0xB800_208C | W/R | Interrupt Group Status Clear Register | 0x0000_0000 |
| **AIC_IRSR** | 0xB800_2100 | R | Interrupt Raw Status Register | 0x0000_0000 |
| **AIC_IASR** | 0xB800_2104 | R | Interrupt Active Status Register | 0x0000_0000 |
| **AIC_ISR** | 0xB800_2108 | R | Interrupt Status Register | 0x0000_0000 |
| **AIC_IPER** | 0xB800_210C | R | Interrupt Priority Encoding Register | 0x0000_0000 |
| **AIC_ISNR** | 0xB800_2110 | R | Interrupt Source Number Register | 0x0000_0000 |
| **AIC_IMR** | 0xB800_2114 | R | Interrupt Mask Register | 0x0000_0000 |
| **AIC_OISR** | 0xB800_2118 | R | Output Interrupt Status Register | 0x0000_0000 |
| **AIC_MECR** | 0xB800_2120 | W | Mask Enable Command Register | N/A |
| **AIC_MDCR** | 0xB800_2124 | W | Mask Disable Command Register | N/A |
| **AIC_EOSCR** | 0xB800_2130 | W | End of Service Command Register | N/A |

## AIC Source Control Registers (AIC_SCR1 ~ AIC_SCR29)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_SCR1 | 0xB800_2004 | R/W | Source Control Register 1 | 0x0000_0047 |
| AIC_SCR2 | 0xB800_2008 | R/W | Source Control Register 2 | 0x0000_0047 |
| • • • | • • • | • • • | • • • | • • • |
| AIC_SCR29 | 0xB800_2074 | R/W | Source Control Register 29 | 0x0000_0047 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SRCTYPE | | RESERVED | | | PRIORITY | | |

| Bits | Descriptions | |
|---|---|---|
| [7:6] | SRCTYPE | **Interrupt Source Type**<br>Whether an interrupt source is considered active or not by the AIC is subject to the settings of this field. Interrupt sources should be configured as level sensitive during normal operation unless in the testing situation. |

| SRCTYPE [7:6] | | Interrupt Source Type |
|---|---|---|
| 0 | 0 | Low-level Sensitive |
| 0 | 1 | High-level Sensitive |
| 1 | 0 | Negative-edge Triggered |
| 1 | 1 | Positive-edge Triggered |

| Bits | Descriptions | |
|---|---|---|
| [2:0] | **PRIORITY** | **Priority Level**<br>Every interrupt source must be assigned a priority level during initiation. Among them, priority level 0 has the highest priority and priority level 7 the lowest. Interrupt sources with priority level 0 are promoted to FIQ. Interrupt sources with priority level other than 0 belong to IRQ. For interrupt sources of the same priority level, which located in the lower channel number has higher priority. |

## External Interrupt Control Register (AIC_IRQSC)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_IRQSC | 0xB800_2080 | R/W | External Interrupt Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | nIRQ2 | | nIRQ1 | | nIRQ0 | |

| Bits | Descriptions | |
|---|---|---|
| [5:0] | **nIRQ*x*** | **External Interrupt Source Type**<br><br>nIRQx: 0 0 — Low-level Sensitive<br>0 1 — High-level Sensitive<br>1 0 — Negative-edge Triggered<br>1 1 — Positive-edge Triggered |

## Interrupt Group Enable Control Register (AIC_GEN)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_GEN | 0xB800_2084 | R/W | Interrupt Group Enable Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | I2C | | RESERVED | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| COMMTX | COMMRX | GDMA | | RESERVED | | TIMER | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | USBH | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | nIRQ[2:0] | | |

| Bits | Descriptions | |
|---|---|---|
| [27:26] | **I2C** | **I2C Controller Interrupt Group**<br>Bit[27] is for I2C Line 1, Bit[26] is for Line 0<br>1: Interrupt Enable for each bit<br>0: Interrupt Disable for each bit |
| [23] | **COMMTX** | **ICE Communications Channel Transmit Interrupt**<br>1: COMMTX Interrupt Enable<br>0: COMMTX Interrupt Disable |
| [22] | **COMMRX** | **ICE Communications Channel Receive Interrupt**<br>1: COMMRX Interrupt Enable<br>0: COMMRX Interrupt Disable |
| [21:20] | **GDMA** | **GDMA Controller Interrupt Group**<br>Bit[21] is for GDMA Channel 1, Bit[20] is for GDMA Channel 0<br>1: Interrupt Enable for each bit<br>0: Interrupt Disable for each bit |

| Bits | Descriptions | |
|---|---|---|
| [18:16] | **TIMER** | **TIMER Controller Interrupt Group**<br>Bit[18] is for TIMER4, Bit[17] is for TIMER3, and Bit[16] is for TIME2<br>1: Interrupt Enable for each bit<br>0: Interrupt Disable for each bit |
| [9:8] | **USBH** | **USB Host Controller Interrupt Group**<br>Bit[9] is for OHCI Host Controller, Bit[9] is for EHCI Host Controller<br>1: Interrupt Enable for each bit<br>0: Interrupt Disable for each bit |
| [2:0] | **nIRQ[2:0]** | **External Interrupt Group 0**<br>1: Interrupt Enable for each bit<br>0: Interrupt Disable for each bit |

## Interrupt Group Active Status Register (AIC_GASR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_GASR | 0xB800_2088 | R | Interrupt Group Active Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | I2C | | RESERVED | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| COMMTX | COMMRX | GDMA | | RESERVED | TIMER | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | USBH | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | nIRQ[2:0] | | |

| Bits | Descriptions | |
|---|---|---|
| [27:26] | I2C | **I2C Controller Interrupt Group**<br>Bit[27] is for I2C Line 1, Bit[26] is for Line 0 |
| [23] | COMMTX | **ICE Communications channel transmit Interrupt**<br>This bit denotes that the comms channel transmit buffer is empty. |
| [22] | COMMRX | **ICE Communications channel Receive Interrupt**<br>This bit denotes that the comms channel receive buffer contains valid data waiting to be read. |
| [21:20] | GDMA | **GDMA Controller Interrupt Group**<br>Bit[21] is for GDMA Channel 1, Bit[20] is for GDMA Channel 0 |
| [18:16] | TIMER | **TIMER Controller Interrupt Group**<br>Bit[18] is for TIMER4, Bit[17] is for TIMER3, and Bit[16] is for TIME2 |
| [9:8] | USBH | **USB Host Controller Interrupt Group**<br>Bit[9] is for OHCI Host Controller, Bit[9] is for EHCI Host Controller |
| [2:0] | nIRQ[2:0] | **External Interrupt Group 0** |

## Interrupt Group Status Clear Register (AIC_GSCR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_GSCR | 0xB800_208C | R/W | Interrupt Group Status Clear Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | nIRQ[2:0] | | |

| Bits | Descriptions | |
|---|---|---|
| [2:0] | nIRQ[2:0] | **External Interrupt Group 0**<br>Write 1: Clear the relative status bit, and this bit is auto clear to 0<br>Write 0: No action |

## AIC Interrupt Raw Status Register (AIC_IRSR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_IRSR | 0xB800_2100 | R | Interrupt Raw Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IRS31 | IRS30 | IRS29 | IRS28 | IRS27 | IRS26 | IRS25 | IRS24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IRS23 | IRS22 | IRS21 | IRS20 | IRS19 | IRS18 | IRS17 | IRS16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IRS15 | IRS14 | IRS13 | IRS12 | IRS11 | IRS10 | IRS9 | IRS8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IRS7 | IRS6 | IRS5 | IRS4 | IRS3 | IRS2 | IRS1 | RESERVED |

| Bits | Descriptions | |
|---|---|---|
| [31:1] | **IRS*x*** | **Interrupt Status**<br>Indicate the intrinsic status of the corresponding interrupt source<br>0 = Interrupt channel is in the voltage level 0<br>1 = Interrupt channel is in the voltage level 1 |

This register records the intrinsic state within each interrupt channel.

:

## AIC Interrupt Active Status Register (AIC_IASR)

This register indicates the status of each interrupt channel in consideration of the interrupt source type as defined in the corresponding Source Control Register, but regardless of its mask setting.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_IASR | 0xB800_2104 | R | Interrupt Active Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IAS31 | IAS30 | IAS29 | IAS28 | IAS27 | IAS26 | IAS25 | IAS24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IAS23 | IAS22 | IAS21 | IAS20 | IAS19 | IAS18 | IAS17 | IAS16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IAS15 | IAS14 | IAS13 | IAS12 | IAS11 | IAS10 | IAS9 | IAS8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IAS7 | IAS6 | IAS5 | IAS4 | IAS3 | IAS2 | IAS1 | RESERVED |

| Bits | Descriptions | |
|---|---|---|
| [31:1] | **IAS*x*** | **Interrupt Active Status**<br>Indicate the status of the corresponding interrupt source<br>0 = Corresponding interrupt channel is inactive<br>1 = Corresponding interrupt channel is active |

:

## AIC Interrupt Status Register (AIC_ISR)

This register identifies all interrupt channels whose are both active and enabled.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_ISR | 0xB800_2108 | R | Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IS31 | IS30 | IS29 | IS28 | IS27 | IS26 | IS25 | IS24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IS23 | IS22 | IS21 | IS20 | IS19 | IS18 | IS17 | IS16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IS15 | IS14 | IS13 | IS12 | IS11 | IS10 | IS9 | IS8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IS7 | IS6 | IS5 | IS4 | IS3 | IS2 | IS1 | RESERVED |

| Bits | Descriptions | |
|---|---|---|
| [31:1] | **IS*x*** | **Interrupt Status**<br>Indicates the status of corresponding interrupt channel<br>0 = Two possibilities:<br>(1) The corresponding interrupt channel is inactive no matter whether it is enabled or disabled;<br>(2) It is active but not enabled<br>1 = Corresponding interrupt channel is both active and enabled (can assert an interrupt) |

## AIC IRQ Priority Encoding Register (AIC_IPER)

When the AIC generates the interrupt, **VECTOR** represents the interrupt channel number that is active, enabled, and has the highest priority. If the representing interrupt channel possesses a priority level 0, then the interrupt asserted is FIQ; otherwise, it is IRQ. The value of **VECTOR** is copied to the register AIC_ISNR thereafter by the AIC. *This register was restored a value 0 after it was read by the interrupt handler*. This register can help indexing into a branch table to quickly jump to the corresponding interrupt service routine.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_IPER | 0xB800_210C | R | Interrupt Priority Encoding Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | VECTOR | | | | | 0 | 0 |

| Bits | Descriptions | |
|---|---|---|
| [6:2] | **VECTOR** | **Interrupt Vector**<br>0 = no interrupt occurs<br>1 ~ 31 = representing the interrupt channel that is active, enabled, and having the highest priority |

## AIC Interrupt Source Number Register (AIC_ISNR)

The purpose of this register is to record the interrupt channel number that is active, enabled, and has the highest priority.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_ISNR | 0xB800_2110 | R | Interrupt Source Number Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | IRQID | | | | |

| Bits | Descriptions | |
|---|---|---|
| [4:0] | IRQID | **IRQ Identification**<br>Stands for the interrupt channel number |

## AIC Interrupt Mask Register (AIC_IMR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_IMR | 0xB800_2114 | R | Interrupt Mask Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IM31 | IM30 | IM29 | IM28 | IM27 | IM26 | IM25 | IM24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IM23 | IM22 | IM21 | IM20 | IM19 | IM18 | IM17 | IM16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IM15 | IM14 | IM13 | IM12 | IM11 | IM10 | IM9 | IM8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IM7 | IM6 | IM5 | IM4 | IM3 | IM2 | IM1 | RESERVED |

| Bits | Descriptions | |
|---|---|---|
| [31:1] | **IM*x*** | **Interrupt Mask**<br>This bit determines whether the corresponding interrupt channel is enabled or disabled. Every interrupt channel can be active no matter whether it is enabled or disabled. If an interrupt channel is enabled, it does not definitely mean it is active. Every interrupt channel can be authorized by the AIC only when it is both active and enabled.<br><br>0 = Corresponding interrupt channel is disabled<br>1 = Corresponding interrupt channel is enabled |

:

## AIC Output Interrupt Status Register (AIC_OISR)

The AIC classifies the interrupt into FIQ and IRQ. This register indicates whether the asserted interrupt is FIQ or IRQ. If both IRQ and FIQ are equal to 0, it means there is no interrupt occurred.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_OISR | 0xB800_2118 | R | Output Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | IRQ | FIQ |

| Bits | Descriptions | |
|---|---|---|
| [1] | IRQ | **Interrupt Request**<br>0 = nIRQ line is inactive.<br>1 = nIRQ line is active. |
| [0] | FIQ | **Fast Interrupt Request**<br>0 = nFIQ line is inactive.<br>1 = nFIQ line is active |

## AIC Mask Enable Command Register (AIC_MECR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_MECR | 0xB800_2120 | W | Mask Enable Command Register | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MEC31 | MEC30 | MEC29 | MEC28 | MEC27 | MEC26 | MEC25 | MEC24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MEC23 | MEC22 | MEC21 | MEC20 | MEC19 | MEC18 | MEC17 | MEC16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MEC15 | MEC14 | MEC13 | MEC12 | MEC11 | MEC10 | MEC9 | MEC8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MEC7 | MEC6 | MEC5 | MEC4 | MEC3 | MEC2 | MEC1 | RESERVED |

| Bits | Descriptions | |
|---|---|---|
| [31:1] | **MEC*x*** | **Mask Enable Command**<br>0 = No effect<br>1 = Enables the corresponding interrupt channel<br>MEC6, MEC10, MEC11, MEC22, MEC24, MEC29, MEC30 and MEC31 have to set 0 for the reserved interrupt source. They should not be enabled the corresponding interrupt channels. |

:

## AIC Mask Disable Command Register (AIC_MDCR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_MDCR | 0xB800_2124 | W | Mask Disable Command Register | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MDC31 | MDC30 | MDC29 | MDC28 | MDC27 | MDC26 | MDC25 | MDC24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MDC23 | MDC22 | MDC21 | MDC20 | MDC19 | MDC18 | MDC17 | MDC16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MDC15 | MDC14 | MDC13 | MDC12 | MDC11 | MDC10 | MDC9 | MDC8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MDC7 | MDC6 | MDC5 | MDC4 | MDC3 | MDC2 | MDC1 | RESERVED |

| Bits | Descriptions | |
|---|---|---|
| [31:1] | **MDC*x*** | **Mask Disable Command**<br>0 = No effect<br>1 = Disables the corresponding interrupt channel |

### AIC End of Service Command Register (AIC_EOSCR)

This register is used by the interrupt service routine to indicate that it is completely served. Thus, the interrupt handler can write any value to this register to indicate the end of its interrupt service.

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_EOSCR | 0xB800_2130 | W | End of Service Command Register | N/A |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| --- | --- | --- | --- | --- | --- | --- | --- |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| --- | --- | --- | --- | --- | --- | --- | --- |

## 7.16 General-Purpose Input/Output (GPIO)

### 7.16.1 Overview

The General-Purpose Input/Output (**GPIO**) module possesses 54 pins, and serves as multiple function purposes. Each port can be easily configured by software to meet various system configurations and design requirements. Software must define which function of each pin is used before starting the main program. If a pin is not used for multiplexed functions, the pin can be configured as I/O ports.

These 52 IO pins are divided into 6 groups according to its peripheral interface definition.

- PortC: 15-pin input/output port
- PortD: 8-pin input/output port
- PortE: 7-pin input/output port
- PortF: 10-pin input/output port
- PortG: 9-pin input/output port
- PortH: 3-pin input/output port

### 7.16.2 GPIO Multiplexed Functions Table

| GPIO Group | Shared Interface |
|---|---|
| GPIOC (15 pins) | NAND Flash / KPI Interface |
| GPIOC[0] | SM_CS0n / KPI_ROW[0] |
| GPIOC[1] | SM_ALE / KPI_ROW[1] |
| GPIOC[2] | SM_CLE / KPI_ROW[2] |
| GPIOC[3] | SM_WEn / KPI_ROW[3] |
| GPIOC[4] | SM_REn |
| GPIOC[5] | SM_WPn |
| GPIOC[6] | SM_RBn |
| GPIOC[7] | SM_D[0] / KPI_COL[0] |
| GPIOC[8] | SM_D[1] / KPI_COL[1] |
| GPIOC[9] | SM_D[2] / KPI_COL[2] |
| GPIOC[10] | SM_D[3] / KPI_COL[3] |
| GPIOC[11] | SM_D[4] / KPI_COL[4] |
| GPIOC[12] | SM_D[5] / KPI_COL[5] |
| GPIOC[13] | SM_D[6] / KPI_COL[6] |
| GPIOC[14] | SM_D[7] / KPI_COL[7] |
| | |
| GPIOD (8 pins) | SD(SDIO) Interface |
| GPIOD[0] | SD_CMD |
| GPIOD[1] | SD_CLK |
| GPIOD[2] | SD_DAT0 |
| GPIOD[3] | SD_DAT1 |
| GPIOD[4] | SD_DAT2 |
| GPIOD[5] | SD_DAT3 |
| GPIOD[6] | SD_CDn |
| GPIOD[8] | SD_nPWR |
| | |
| GPIOE (7 pins) | UART Interface |
| GPIOE[0] | TXD0 |
| GPIOE[1] | RXD0 |

| | |
|---|---|
| GPIOE[2] | TXD1 |
| GPIOE[3] | RXD1 |
| GPIOE[6] | TXD2(IrDA) |
| GPIOE[7] | RXD2(IrDA) |
| GPIOE[13] | GPIOE13 |
| | |
| GPIOF (10 pins) | RMII Interface |
| GPIOF[0] | PHY_MDC |
| GPIOF [1] | PHY_MDIO |
| GPIOF [3:2] | PHY_TXD [1:0] |
| GPIOF [4] | PHY_TXEN |
| GPIOF [5] | PHY_REFCLK |
| GPIOF [7:6] | PHY_RXD [1:0] |
| GPIOF [8] | PHY_CRSDV |
| GPIOF [9] | PHY_RXERR |
| | |
| GPIOG (9 pins) | I2C/USI<br>AC97/I2S/PWM Interface |
| GPIOG[0] | SCL0 /<br>SFRM |
| GPIOG[1] | SDA0 /<br>SSPTXD |
| GPIOG[2] | SCL1 /<br>SCLK |
| GPIOG[3] | SDA1 /<br>SSPRXD |
| GPIOG[12] | AC97_nRESET /<br>I2S_SYSCLK |
| GPIOG[13] | AC97_DATAI /<br>I2S_DATAI /<br>PWM [0] |
| GPIOG[14] | AC97_DATAO /<br>I2S_DATAO /<br>PWM [1] |
| GPIOG[15] | AC97_SYNC /<br>I2S_WS /<br>PWM [2] |

| | |
|---|---|
| GPIOG[16] | AC97_BITCLK / I2S_BITCLK / PWM [3] |
| | |
| GPIOH (3 pins) | nIRQ Interface |
| GPIOH[2:0] | nIRQ[2:0] |
| | |

### 7.16.3 GPIO Control Registers Map

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIO_BA = 0xB800_3000 | | | | |
| GPIOC_DIR | 0xB800_3004 | R/W | GPIO portC direction control register | 0x0000_0000 |
| GPIOC_DATAOUT | 0xB800_3008 | R/W | GPIO portC data output register | 0x0000_0000 |
| GPIOC_DATAIN | 0xB800_300C | R | GPIO portC data input register | N/A |
| GPIOD_DIR | 0xB800_3014 | R/W | GPIO portD direction control register | 0x0000_0000 |
| GPIOD_DATAOUT | 0xB800_3018 | R/W | GPIO portD data output register | 0x0000_0000 |
| GPIOD_DATAIN | 0xB800_301C | R | GPIO portD data input register | N/A |
| GPIOE_DIR | 0xB800_3024 | R/W | GPIO portE direction control register | 0x0000_0000 |
| GPIOE_DATAOUT | 0xB800_3028 | R/W | GPIO portE data output register | 0x0000_0000 |
| GPIOE_DATAIN | 0xB800_302C | R | GPIO portE data input register | 0x0000_0000 |
| GPIOF_DIR | 0xB800_3034 | R/W | GPIO portF direction control register | 0x0000_0000 |
| GPIOF_DATAOUT | 0xB800_3038 | R/W | GPIO portF data output register | 0x0000_0000 |
| GPIOF_DATAIN | 0xB800_303C | R | GPIO portF data input register | N/A |
| GPIOG_DIR | 0xB800_3044 | R/W | GPIO portG direction control register | 0x0000_0000 |
| GPIOG_DATAOUT | 0xB800_3048 | R/W | GPIO portG data output register | 0x0000_0000 |
| GPIOG_DATAIN | 0xB800_304C | R | GPIO portG data input register | N/A |
| GPIOH_DBNCE | 0xB800_3050 | R/W | GPIO portH input de-bounce control reg. | 0x0000_0000 |
| GPIOH_DIR | 0xB800_3054 | R/W | GPIO portH direction control register | 0x0000_0000 |
| GPIOH_DATAOUT | 0xB800_3058 | R/W | GPIO portH data output register | 0x0000_0000 |
| GPIOH_DATAIN | 0xB800_305C | R | GPIO portH data input register | N/A |

## GPIO PortC Direction Control Register (GPIOC_DIR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOC_DIR | 0xB800_3004 | R/W | GPIO portC in/out direction control register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | OUTEN | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTEN | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [14:0] | **OUTEN** | **GPIO PortC Output Enable Control**<br>Each GPIO pin can be enabled individually by setting the corresponding control bit.<br>0 = Input Mode<br>1 = Output Mode |

## GPIO PortC Data Output Register (GPIOC_DATAOUT)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOC_DATAOUT | 0xB800_3008 | R/W | GPIO portC data output register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | DATAOUT | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATAOUT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [14:0] | DATAOUT | **GPIO PortC Data Output Value**<br>Writing data to this register will reflect the data value on the corresponding pin when it is configured as general output pin. And writing data to reserved bits is not effective. |

## GPIO PortC Data Input Register (GPIOC_DATAIN)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOC_DATAIN | 0xB800_300C | R | GPIO portC data input register | N/A |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | DATAIN | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATAIN | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [14:0] | DATAIN | **GPIO PortC Data Input Value**<br>The DATAIN indicates the status of each GPIO portC pin regardless of its operation mode. The reserved bits will be read as "0". |

## GPIO PortD Direction Control Register (GPIOD_DIR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOD_DIR | 0xB800_3014 | R/W | GPIO portD in/out direction control register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | OUTEN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTEN | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [8:0] | **OUTEN** | **GPIO PortD Output Enable Control**<br>Each GPIO pin can be enabled individually by setting the corresponding control bit.<br>0 = Input Mode<br>1 = Output Mode<br>Bit7 is no action |

## GPIO PortD Data Output Register (GPIOD_DATAOUT)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOD_DATAOUT | 0xB800_3018 | R/W | GPIO portD data output register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | DATAOUT |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATAOUT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [8:0] | **DATAOUT** | **GPIO PortD Data Output Value**<br>Writing data to this register will reflect the data value on the corresponding pin when it is configured as general output pin. And writing data to reserved bits is not effective.<br>Bit7 is no action |

## GPIO PortD Data Input Register (GPIOD_DATAIN)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOD_DATAIN | 0xB800_301C | R | GPIO portD data input register | N/A |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | DATAIN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATAIN | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [8:0] | DATAIN | **GPIO PortD Data Input Value**<br>The DATAIN indicates the status of each GPIO portD pin regardless of its operation mode. The reserved bits will be read as "0".<br>Bit7 is reserved. |

## GPIO PortE Direction Control Register (GPIOE_DIR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOE_DIR | 0xB800_3024 | R/W | GPIO portE in/out direction control register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | OUTEN | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTEN | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [13:0] | **OUTEN** | **GPIO PortE Output Enable Control**<br>Each GPIO pin can be enabled individually by setting the corresponding control bit.<br>0 = Input Mode<br>1 = Output Mode<br>Bit12-bit8 and bit5-bit4 are no action |

## GPIO PortE Data Output Register (GPIOE_DATAOUT)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOE_DATAOUT | 0xB800_3028 | R/W | GPIO portE data output register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | DATAOUT | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATAOUT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [13:0] | DATAOUT | **GPIO PortE Data Output Value**<br>Writing data to this register will reflect the data value on the corresponding pin when it is configured as general output pin. And writing data to reserved bits is not effective.<br>Bit12-bit8 and bit5-bit4 are no action |

## GPIO PortE Data Input Register (GPIOE_DATAIN)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOE_DATAIN | 0xB800_302C | R | GPIO portE data input register | N/A |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | DATAIN | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATAIN | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [13:0] | **DATAIN** | **GPIO PortE Data Input Value**<br>The DATAIN indicates the status of each GPIO portE pin regardless of its operation mode. The reserved bits will be read as "0".<br>Bit12-bit8 and bit5-bit4 are reserved |

## GPIO PortF Direction Control Register (GPIOF_DIR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOF_DIR | 0xB800_3034 | R/W | GPIO portF in/out direction control register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | OUTEN | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTEN | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [9:0] | **OUTEN** | **GPIO PortF Output Enable Control**<br>Each GPIO pin can be enabled individually by setting the corresponding control bit.<br>0 = Input Mode<br>1 = Output Mode |

## GPIO PortF Data Output Register (GPIOF_DATAOUT)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOF_DATAOUT | 0xB800_3038 | R/W | GPIO portF data output register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | DATAOUT | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATAOUT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [9:0] | DATAOUT | **GPIO PortF Data Output Value**<br>Writing data to this register will reflect the data value on the corresponding pin when it is configured as general output pin. And writing data to reserved bits is not effective. |

## GPIO PortF Data Input Register (GPIOF_DATAIN)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOF_DATAIN | 0xB800_303C | R | GPIO portF data input register | 0xxxxx_xxxx |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | DATAIN | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATAIN | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [9:0] | DATAIN | **GPIO PortF Data Input Value**<br>The DATAIN indicates the status of each GPIO portF pin regardless of its operation mode. The reserved bits will be read as "0". |

## GPIO PortG Direction Control Register (GPIOG_DIR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOG_DIR | 0xB800_3044 | R/W | GPIO portG in/out direction control register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | OUTEN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| OUTEN | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTEN | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [16:0] | **OUTEN** | **GPIO PortG Output Enable Control**<br>Each GPIO pin can be enabled individually by setting the corresponding control bit.<br>0 = Input Mode<br>1 = Output Mode<br>Bit11 ~ bit4 are no action. |

## GPIO PortG Data Output Register (GPIOG_DATAOUT)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOG_DATAOUT | 0xB800_3048 | R/W | GPIO portG data output register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | DATAOUT |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DATAOUT | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATAOUT | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [16:0] | DATAOUT | **GPIO PortG Data Output Value**<br>Writing data to this register will reflect the data value on the corresponding pin when it is configured as general output pin. And writing data to reserved bits is not effective.<br>Bit11 ~ bit4 are no action. |

## GPIO PortG Data Input Register (GPIOG_DATAIN)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOG_DATAIN | 0xB800_304C | R | GPIO portG data input register | N/A |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | DATAIN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DATAIN | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATAIN | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [16:0] | DATAIN | **GPIO PortG Data Input Value**<br>The DATAIN indicates the status of each GPIO portG pin regardless of its operation mode. The reserved bits will be read as "0".<br>Bit11-bit4 are reserved |

## GPIO PortH De-bounce Enable Control Register (GPIOH_DBNCE)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOH_DBNCE | 0xB800_3050 | R/W | GPIO PortH de-bounce control register | N/A |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | DBCLKSEL | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | DBEN2 | DBEN1 | DBEN0 |

| Bits | Descriptions | |
|---|---|---|
| [10:8] | **DBCLKSEL** | **De-bounce Clock Selection**<br><br>These 3 bits are used to select the clock rate for de-bouncer circuit. The relationship between the system clock HCLK and the de-bounce clock TCLK_BUN is as follows:<br><br>TCLK_BUN = HCLK / $2^{DBCLKSEL}$ |
| [2] | **DBEN2** | **De-bounce Circuit Enable for GPIOH2 (nIRQ2) Input**<br>1 = Enable De-bounce<br>0 = Disable De-bounce |
| [1] | **DBEN1** | **De-bounce Circuit Enable for GPIOH1 (nIRQ1) Input**<br>1 = Enable De-bounce<br>0 = Disable De-bounce |
| [0] | **DBEN0** | **De-bounce Circuit Enable for GPIOH0 (nIRQ0) Input**<br>1 = Enable De-bounce<br>0 = Disable De-bounce |

## GPIO PortH Direction Control Register (GPIOH_DIR)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOH_DIR | 0xB800_3054 | R/W | GPIO portH in/out direction control register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | OUTEN | | |

| Bits | Descriptions | |
|---|---|---|
| [2:0] | **OUTEN** | **GPIO PortH Output Enable Control**<br>Each GPIO pin can be enabled individually by setting the corresponding control bit.<br>0 = Input Mode<br>1 = Output Mode |

## GPIO PortH Data Output Register (GPIOH_DATAOUT)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOH_DATAOUT | 0xB800_3058 | R/W | GPIO portH data output register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | DATAOUT | | |

| Bits | Descriptions | |
|---|---|---|
| [2:0] | **DATAOUT** | **GPIO PortH Data Output Value**<br>Writing data to this register will reflect the data value on the corresponding pin when it is configured as general output pin. And writing data to reserved bits is not effective. |

### GPIO PortH Data Input Register (GPIOH_DATAIN)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIOH_DATAIN | 0xB800_305C | R | GPIO portH data input register | N/A |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | DATAIN | | |

| Bits | Descriptions | |
|---|---|---|
| [2:0] | **DATAIN** | **GPIO PortH Data Input Value**<br>The DATAIN indicates the status of each GPIO portH pin regardless of its operation mode. The reserved bits will be read as "0". |

## 7.17 I2C Synchronous Serial Interface Controller

$I^2C$ is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. The $I^2C$ standard is a true multi-master bus including collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously.

Serial, 8-bit oriented bi-directional data transfers can be up to 100 Kb/s in Standard-mode, 400 Kb/s in the Fast-mode, or 3.4 Mb/s in the High-speed mode. Only 100kbps and 400kbps modes are supported directly in this chip.

Data transfer is synchronized to SCL signal between a Master and a Slave with **byte-by-byte** basis. Each data byte is 8 bits long. There is one SCL clock pulse for each data bit with the **MSB being transmitted first**. An acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; therefore, the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (START or STOP).

The $I^2C$ Master Core includes the following features:

Provide multi-byte transmit operation, up to 4 bytes can be transmitted in a single transfer

Software programmable acknowledge bit.

Arbitration lost interrupt, with automatic transfer cancellation.

Start/Stop/Repeated Start/Acknowledge generation.

Start/Stop/Repeated Start detection.

Bus busy detection.

Supports 7 bit addressing mode.

Software mode $I^2C$.

### 7.17.1 I2C Serial Interface Control Registers Map

**R**: read only, **W**: write only, **R/W**: both read and write

| Register | Offset | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| **I2C Port0 : I2C_BA = 0xB800_6000**<br>**I2C Port1 : I2C_BA = 0xB800_6100** | | | | |
| **CSR** | 0xB800_6x00 | R/W | Control and Status Register | 0x0000_0000 |
| **DIVIDER** | 0XB800_6x04 | R/W | Clock Pre-scale Register | 0x0000_0000 |
| **CMDR** | 0XB800_6x08 | R/W | Command Register | 0x0000_0000 |
| **SWR** | 0XB800_6x0C | R/W | Software Mode Control Register | 0x0000_003F |
| **RxR** | 0XB800_6x10 | R | Data Receive Register | 0x0000_0000 |
| **TxR** | 0XB800_6x14 | R/W | Data Transmit Register | 0x0000_0000 |

NOTE: The reset value of SWR is 0x3F only when SCR, SDR and SER are connected to pull high resistor.

## Control and Status Register (CSR)

| Register | Offset | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| CSR | 0XB800_6x00 | R/W | Control and Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | I2C_RxACK | I2C_BUSY | I2C_AL | I2C_TIP |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | Tx_NUM | | Reserved | IF | IE | I2C_EN |

| Bits | Descriptions | |
|---|---|---|
| [11] | I2C_RxACK | **Received Acknowledge From Slave (Read only)**<br>This flag represents acknowledge from the addressed slave.<br>• 0 = Acknowledge received (ACK).<br>• 1 = Not acknowledge received (NACK). |
| [10] | I2C_BUSY | **$I^2C$ Bus Busy (Read only)**<br>• 0 = After STOP signal detected.<br>• 1 = After START signal detected. |
| [9] | I2C_AL | **Arbitration Lost (Read only)**<br>This bit is set when the $I^2C$ core lost arbitration. Arbitration is lost when:<br>• A STOP signal is detected, but no requested.<br>• The master drives SDA high, but SDA is low. |

| | | |
|---|---|---|
| [8] | **I2C_TIP** | **Transfer In Progress (Read only)**<br>• 0 = Transfer complete.<br>• 1 = Transferring data.<br>**NOTE:** When a transfer is in progress, you will not allow writing to any register of the $I^2C$ master core except SWR. |

| | | |
|---|---|---|
| [5:4] | **Tx_NUM** | **Transmit Byte Counts**<br>These two bits represent how many bytes are remained to transmit. When a byte has been transmitted, the Tx_NUM will decrease 1 until all bytes are transmitted (Tx_NUM = 0x0) or NACK received from slave. Then the interrupt signal will assert if IE was set.<br>0x0 = Only one byte is left for transmission.<br>0x1 = Two bytes are left to for transmission.<br>0x2 = Three bytes are left for transmission.<br>0x3 = Four bytes are left for transmission. |
| [2] | **IF** | **Interrupt Flag**<br>The Interrupt Flag is set when:<br>• Transfer has been completed.<br>• Transfer has not been completed, but slave responded NACK (in multi-byte transmit mode).<br>• Arbitration is lost.<br>**NOTE:** This bit is read only, but can be cleared by writing 1 to this bit. |
| [1] | **IE** | **Interrupt Enable**<br>• 0 = **Disable** $I^2C$ Interrupt.<br>• 1 = **Enable** $I^2C$ Interrupt. |
| [0] | **I2C_EN** | **$I^2C$ Core Enable**<br>• 0 = **Disable** $I^2C$ core, serial bus outputs are controlled by SDW/SCW.<br>• 1 = **Enable** $I^2C$ core, serial bus outputs are controlled by $I^2C$ core. |

## Pre-scale Register (DIVIDER)

| Register | Offset | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| DIVIDER | 0XB800_6x04 | R/W | Clock Pre-scale Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DIVIDER[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DIVIDER[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | DIVIDER | **Clock Pre-scale Register**<br>It is used to pre-scale the SCL clock line. Due to the structure of the $I^2C$ interface, the core uses a 5*SCL clock internally. The pre-scale register must be programmed to this 5*SCL frequency (minus 1). Change the value of the pre-scale register only when the "I2C_EN" bit is cleared.<br>Example: pclk = 32MHz, desired SCL = 100KHz<br>$prescale = \frac{32\ MHz}{5 * 100\ KHz} - 1 = 63\ (dec) = 3F\ (hex)$ |

## Command Register (CMDR)

| Register | Offset | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| CMDR | 0XB800_6x08 | R/W | Command Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | START | STOP | READ | WRITE | ACK |

**NOTE:** Software can write this register only when I2C_EN = 1.

| Bits | Descriptions | |
|---|---|---|
| [4] | **START** | **Generate Start Condition**<br>Generate (repeated) start condition on $I^2C$ bus. |
| [3] | **STOP** | **Generate Stop Condition**<br>Generate stop condition on $I^2C$ bus. |
| [2] | **READ** | **Read Data From Slave**<br>Retrieve data from slave. |
| [1] | **WRITE** | **Write Data To Slave**<br>Transmit data to slave. |
| [0] | **ACK** | **Send Acknowledge To Slave**<br>When $I^2C$ behaves as a receiver, sent ACK (ACK = '0') or NACK (ACK = '1') to slave. |

NOTE: The START, STOP, READ and WRITE bits are cleared automatically while transfer finished. READ and WRITE cannot be set concurrently.

## Software Mode Register (SWR)

| Register | Offset | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| SWR | 0XB800_6x0C | R/W | Software Mode Control Register | 0x0000_003F |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | SER | SDR | SCR | SEW | SDW | SCW |

**NOTE:** This register is used as software mode of $I^2C$. Software can read/write this register no matter I2C_EN is 0 or 1. But SCL and SDA are controlled by software only when I2C_EN = 0.

| Bits | Descriptions | |
|---|---|---|
| [5] | **SER** | **Serial Interface SDO Status (Read only)**<br>0 = SDO is Low.<br>1 = SDO is High. |
| [4] | **SDR** | **Serial Interface SDA Status (Read only)**<br>0 = SDA is Low.<br>1 = SDA is High. |
| [3] | **SCR** | **Serial Interface SCK Status (Read only)**<br>0 = SCL is Low.<br>1 = SCL is High. |
| [2] | **SEW** | **Serial Interface SDO Output Control**<br>0 = SDO pin is driven Low.<br>1 = SDO pin is tri-state. |
| [1] | **SDW** | **Serial Interface SDA Output Control**<br>0 = SDA pin is driven Low.<br>1 = SDA pin is tri-state. |

| Bits | Descriptions | |
|---|---|---|
| [0] | **SCW** | **Serial Interface SCK Output Control**<br>0 = SCL pin is driven Low.<br>1 = SCL pin is tri-state. |

### Data Receive Register (RxR)

| Register | Offset | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| RxR | 0XB800_6x10 | R | Data Receive Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Rx[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [7:0] | **Rx** | **Data Receive Register**<br>The last byte received via $I^2C$ bus will put on this register. The $I^2C$ core only used 8-bit receive buffer. |

### Data Transmit Register (TxR)

| Register | Offset | R/W/C | Description | Reset Value |
|---|---|---|---|---|
| TxR | 0XB800_6x14 | R/W | Data Transmit Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Tx[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Tx[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Tx[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Tx[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **Tx** | **Data Transmit Register**<br>The $I^2C$ core used 32-bit transmit buffer and provide multi-byte transmit function. Set CSR [Tx_NUM] to a value that you want to transmit. $I^2C$ core will always issue a transfer from the highest byte first. For example, if CSR [Tx_NUM] = 0x3, Tx [31:24] will be transmitted first, then Tx [23:16], and so on.<br>In case of a data transfer, all bits will be treated as data.<br>In case of a slave address transfer, the first 7 bits will be treated as 7-bit address and the LSB represent the R/W bit. In this case,<br>LSB = 1, reading from slave<br>LSB = 0, writing to slave |

## 7.18Universal Serial Interface Controller (USI)

The USI is a synchronous serial interface performs a serial-to-parallel conversion on data characters received from the peripheral, and a parallel-to-serial conversion on data characters received from CPU. This interface generates an interrupt signal when data transfer is finished and can be cleared by writing 1 to the interrupt flag. The active level of device/slave select signal can be chosen to low active or high active, which depends on the peripheral it's connected. Writing a divisor into DIVIDER register can program the frequency of serial clock output. This master core contains four 32-bit transmit/receive

buffers, and can provide burst mode operation. The maximum bits can be transmitted/received is 32 bits, and can transmit/receive data up to four times successive.

The USI (MICROWIRE/SPI) Master Core includes the following features:

Support MICROWIRE/SPI master mode

Full duplex synchronous serial data transfer

Variable length of transfer word up to 32 bits

Provide burst mode operation, transmit/receive can be executed up to four times in one transfer

MSB or LSB first data transfer

Rx and Tx on rising or falling edge of serial clock independently

### 7.18.1 USI Control Registers Map

**R**: read only, **W**: write only, **R/W**: both read and write

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USI_BA = 0xB800_6200** | | | | |
| **CNTRL** | 0xB800_6200 | R/W | Control and Status Register | 0x0000_0004 |
| **DIVIDER** | 0xB800_6204 | R/W | Clock Divider Register | 0x0000_0000 |
| **SSR** | 0xB800_6208 | R/W | Slave Select Register | 0x0000_0000 |
| **Rx0** | 0xB800_6210 | R | Data Receive Register 0 | 0x0000_0000 |
| **Rx1** | 0xB800_6214 | R | Data Receive Register 1 | 0x0000_0000 |
| **Rx2** | 0xB800_6218 | R | Data Receive Register 2 | 0x0000_0000 |
| **Rx3** | 0xB800_621C | R | Data Receive Register 3 | 0x0000_0000 |
| **Tx0** | 0xB800_6210 | W | Data Transmit Register 0 | 0x0000_0000 |
| **Tx1** | 0xB800_6214 | W | Data Transmit Register 1 | 0x0000_0000 |
| **Tx2** | 0xB800_6218 | W | Data Transmit Register 2 | 0x0000_0000 |
| **Tx3** | 0xB800_621C | W | Data Transmit Register 3 | 0x0000_0000 |

NOTE 1: When software programs CNTRL, the GO_BUSY bit should be written last.

## Control and Status Register (CNTRL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CNTRL | 0xB800_6200 | R/W | Control and Status Register | 0x0000_0004 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CLK_POL | Reserved | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | IE | IF |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SLEEP | | | | Reserved | LSB | Tx_NUM | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Tx_BIT_LEN | | | | | Tx_NEG | Rx_NEG | GO_BUSY |

| Bits | Descriptions | |
|---|---|---|
| [31] | **CLK_POL** | **Clock Polarity**<br>• 0 = Normal polarity.<br>• 1 = Reverse polarity. |
| [17] | **IE** | **Interrupt Enable**<br>• 0 = **Disable** USI Interrupt.<br>• 1 = **Enable** USI Interrupt. |
| [16] | **IF** | **Interrupt Flag**<br>• 0 = It indicates that the transfer dose not finish yet.<br>• 1 = It indicates that the transfer is done. The interrupt flag is set if it was enable.<br>**NOTE:** This bit is read only, but can be cleared by writing 1 to this bit. |

| Bits | Descriptions | |
|---|---|---|
| [15:12] | **SLEEP** | **Suspend Interval**<br>These four bits provide the configuration of suspend interval between two successive transmit/receive in a transfer. The default value is 0x0. When CNTRL [Tx_NUM] = 00, setting this field has no effect on transfer. The desired interval is obtained according to the following equation (from the last falling edge of current **SCLK** to the first rising edge of next **SCLK**):<br>(CNTRL[SLEEP] + 2)*period of **SCLK**<br>SLEEP = 0x0 … 2 SCLK clock cycle<br>SLEEP = 0x1 … 3 SCLK clock cycle<br>……<br>SLEEP = 0xe … 16 SCLK clock cycle<br>SLEEP = 0xf … 17 SCLK clock cycle |
| [10] | **LSB** | **Send LSB First**<br>0 = The **MSB** is transmitted/received first (which bit in TxX/RxX register that is depends on the Tx_BIT_LEN field in the CNTRL register).<br>1 = The **LSB** is sent first on the line (bit TxX [0]), and the first bit received from the line will be put in the LSB position in the Rx register (bit RxX [0]). |
| [9:8] | **Tx_NUM** | **Transmit/Receive Numbers**<br>This field specifies how many transmit/receive numbers should be executed in one transfer.<br>00 = Only one transmit/receive will be executed in one transfer.<br>01 = Two successive transmit/receive will be executed in one transfer.<br>10 = Three successive transmit/receive will be executed in one transfer.<br>11 = Four successive transmit/receive will be executed in one transfer. |

| Bits | Descriptions | |
|---|---|---|
| [7:3] | **Tx_BIT_LEN** | **Transmit Bit Length**<br>This field specifies how many bits are transmitted in one transmit/receive. Up to 32 bits can be transmitted.<br>Tx_BIT_LEN = 0x01 … 1 bit<br>Tx_BIT_LEN = 0x02 … 2 bits<br>……<br>Tx_BIT_LEN = 0x1f … 31 bits<br>Tx_BIT_LEN = 0x00 … 32 bits |
| [2] | **Tx_NEG** | **Transmit On Negative Edge**<br>0 = The **SSPTXD** signal is changed on the **rising** edge of **SCLK**.<br>1 = The **SSPTXD** signal is changed on the **falling** edge of **SCLK**. |
| [1] | **Rx_NEG** | **Receive On Negative Edge**<br>0 = The **SSPRXD** signal is latched on the **rising** edge of **SCLK**.<br>1 = The **SSPRXD** signal is latched on the **falling** edge of **SCLK**. |
| [0] | **GO_BUSY** | **Go and Busy Status**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit starts the transfer. This bit remains set during the transfer and is automatically cleared after transfer finished.<br>**NOTE:** All registers should be set before writing 1 to the GO_BUSY bit in the CNTRL register. When a transfer is in progress, writing to any register of the USI master core has no effect. |

## Divider Register (DIVIDER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **DIVIDER** | 0xB800_6204 | R/W | Clock Divider Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **DIVIDER[15:8]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **DIVIDER[7:0]** | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **DIVIDER** | **Clock Divider Register**<br>The value in this field is the frequency divider of the system clock pclk to generate the serial clock on the output **SCLK**. The desired frequency is obtained according to the following equation:<br>$f_{sclk} = \frac{f_{pclk}}{(DIVIDER+1)*2}$<br>**NOTE:** Suggest DIVIDER should be at least 1. |

## Slave Select Register (SSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SSR | 0xB800_6208 | R/W | Slave Select Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | ASS | SS_LVL | SSR[1:0] | |

| Bits | Descriptions | |
|---|---|---|
| [3] | **ASS** | **Automatic Slave Select**<br>0 = If this bit is cleared, slave select signals are asserted and de-asserted by setting and clearing related bits in SSR register.<br>1 = If this bit is set, **SSPTXD** signals are generated automatically. It means that device/slave select signal, which is set in SSR register is asserted by the USI controller when transmit/receive is started by setting CNTRL [GO_BUSY], and is de-asserted after every transmit/receive is finished. |
| [2] | **SS_LVL** | **Slave Select Active Level**<br>It defines the active level of device/slave select signal (**SSPTXD**).<br>• 0 = The **SSPTXD** slave select signal is active **Low**.<br>• 1 = The **SSPTXD** slave select signal is active **High**. |

| Bits | Descriptions | |
|---|---|---|
| [1:0] | **SSR** | **Slave Select Register**<br>If SSR [ASS] bit is cleared, writing 1 to any bit location of this field sets the proper **SSPTXD** line to an active state and writing 0 sets the line back to inactive state.<br>If SSR[ASS] bit is set, writing 1 to any bit location of this field will select appropriate mw_ss_o line to be automatically driven to active state for the duration of the transmit/receive, and will be driven to inactive state for the rest of the time. (The active level of **SSPTXD** is specified in SSR [SS_LVL]).<br>**NOTE:** This interface can only drive one device/slave at a given time. Therefore, the slave select of the selected device must be set to its active level before starting any read or write transfer. |

### Data Receive Register 0 (Rx0)

### Data Receive Register 1 (Rx1)

### Data Receive Register 2 (Rx2)

### Data Receive Register 3 (Rx3)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **Rx0** | 0xB800_6210 | R | Data Receive Register 0 | 0x0000_0000 |
| **Rx1** | 0xB800_6214 | R | Data Receive Register 1 | 0x0000_0000 |
| **Rx2** | 0xB800_6218 | R | Data Receive Register 2 | 0x0000_0000 |
| **Rx3** | 0xB800_621C | R | Data Receive Register 3 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Rx[31:24] | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Rx[23:16] | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Rx[15:8] | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Rx[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **Rx** | **Data Receive Register**<br>The Data Receive Registers hold the value of received data of the last executed transfer. Valid bits depend on the transmit bit length field in the CNTRL register. For example, if CNTRL [Tx_BIT_LEN] is set to 0x08 and CNTRL [Tx_NUM] is set to 0x0, bit Rx0 [7:0] holds the received data.<br>**NOTE:** The Data Receive Registers are **read only** registers. A Write to these registers will actually modify the Data Transmit Registers because those registers share the same FFs. |

### Data Transmit Register 0 (Tx0)

### Data Transmit Register 1 (Tx1)

### Data Transmit Register 2 (Tx2)

### Data Transmit Register 3 (Tx3)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **Tx0** | 0xB800_6210 | W | Data Transmit Register 0 | 0x0000_0000 |
| **Tx1** | 0xB800_6214 | W | Data Transmit Register 1 | 0x0000_0000 |
| **Tx2** | 0xB800_6218 | W | Data Transmit Register 2 | 0x0000_0000 |
| **Tx3** | 0xB800_621C | W | Data Transmit Register 3 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Tx[31:24] | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Tx[23:16] | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Tx[15:8] | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Tx[7:0] | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:0] | **Tx** | **Data Transmit Register**<br>The Data Transmit Registers hold the data to be transmitted in the next transfer. Valid bits depend on the transmit bit length field in the CNTRL register. For example, if CNTRL [Tx_BIT_LEN] is set to 0x08 and the CNTRL [Tx_NUM] is set to 0x0, the bit Tx0 [7:0] will be transmitted in next transfer. If CNTRL [Tx_BIT_LEN] is set to 0x00 and CNTRL [Tx_NUM] is set to 0x3, the core will perform four 32-bit transmit/receive successive using the same setting (the order is Tx0 [31:0], Tx1 [31:0], Tx2 [31:0], Tx3 [31:0]).<br>**NOTE:** The RxX and TxX registers share the same flip-flops. Which means that what is received from the input data line in one transfer will be transmitted on the output data line in the next transfer if no write access to the TxX register is executed between the transfers. |

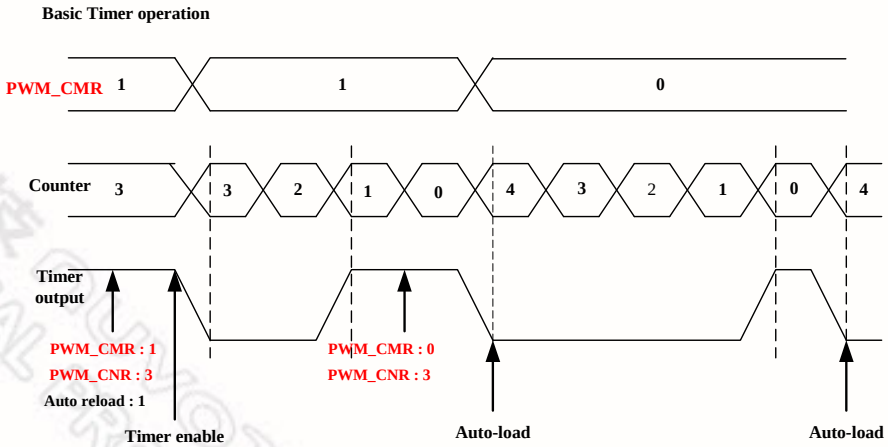
Pulse Width Modulation (PWM)

This Controller includes 4 channels PWM Timers. They can be divided into two groups. Each group has 1 Pre-scalar, 1 clock divider, 2 clock selectors, 2 16-bit counters, 2 16-bit comparators, 1 Dead-Zone generator. They are all driven by APB system clock (PCLK) in chip. Each channel can be used as a timer and issue interrupt independently.

Two channels PWM Timers in one group share the same pre-scalar. Clock divider provides each channel with 5 clock sources (1, 1/2, 1/4, 1/8, 1/16). Each channel receives its own clock signal from clock divider which receives clock from 8-bit pre-scalar. The 16-bit counter in each channel receive clock signal from clock selector and can be used to handle one PWM period. The 16-bit comparator compares number in counter with threshold number in register loaded previously to generate PWM duty cycle. The clock signal from clock divider is called PWM clock. Dead-Zone generator utilize PWM clock as clock source. Once Dead-Zone generator is enabled, output of two PWM timers in one group is blocked. Two output pins are all used as Dead-Zone generator output signal to control off-chip power device.

To prevent PWM driving output pin with unsteady waveform, 16-bit counter and 16-bit comparator are implemented with double buffering feature. User can feel free to write data to counter buffer register and comparator buffer register without generating glitch. When 16-bit down counter reaches zero, the interrupt request is generated to inform CPU that time is up. When counter reaches zero, if counter is set as toggle mode, it is reloaded automatically and start to generate next cycle. User can set counter as one-shot mode instead of toggle mode. If counter is set as one-shot mode, counter will stop and generate one interrupt request when it reaches zero. The value of comparator is used for pulse width modulation. The counter control logic changes the output level when down-counter value matches the value of compare register.

### 7.18.2 Basic Timer Operation



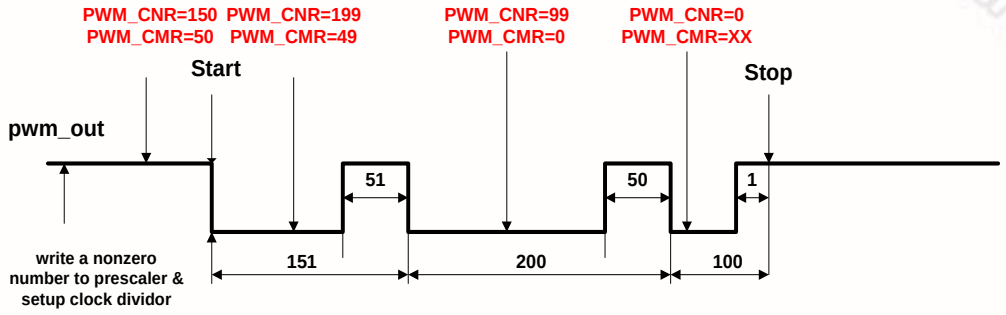

### 7.18.3 PWM Double Buffering and Reload Automatically

The PWM Timers have a double buffering function, enabling the reload value changed for next timer operation without stopping current timer operation. Although new timer value is set, current timer operation still operate successfully.

The counter value can be written into PWM_CNR0, PWM_CNR1, PWM_CNR2, PWM_CNR3 and current counter value can be read from PWM_PDR0, PWM_PDR1, PWM_PDR2, PWM_PDR3.

The auto-reload operation copies from PWM_CNR0, PWM_CNR1, PWM_CNR2, PWM_CNR3 to down-counter when down-counter reaches zero. If PWM_CNR0~3 are set as zero, counter will be halt when counter count to zero. If auto-reload bit is set as zero, counter will be stopped immediately.

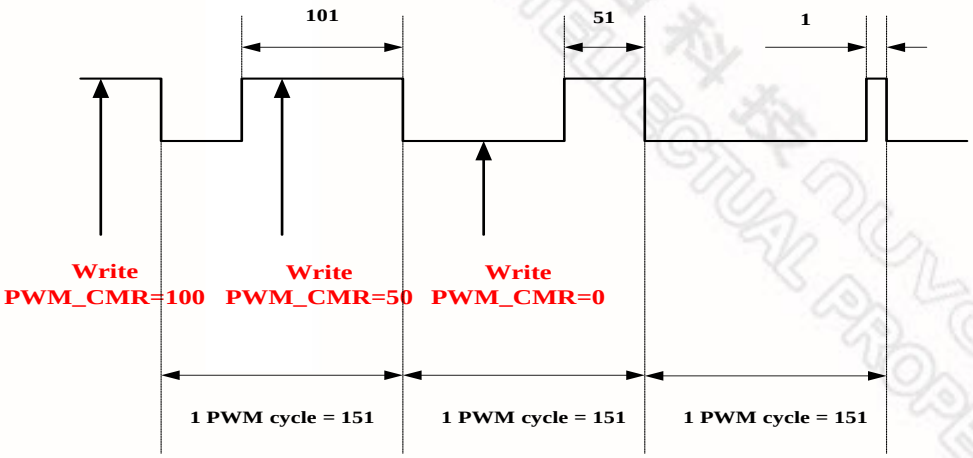
**PWM double buffering**




### 7.18.4 Modulate Duty Ratio

The double buffering function allows PWM_CMR written at any point in current cycle. The loaded value will take effect from next cycle.

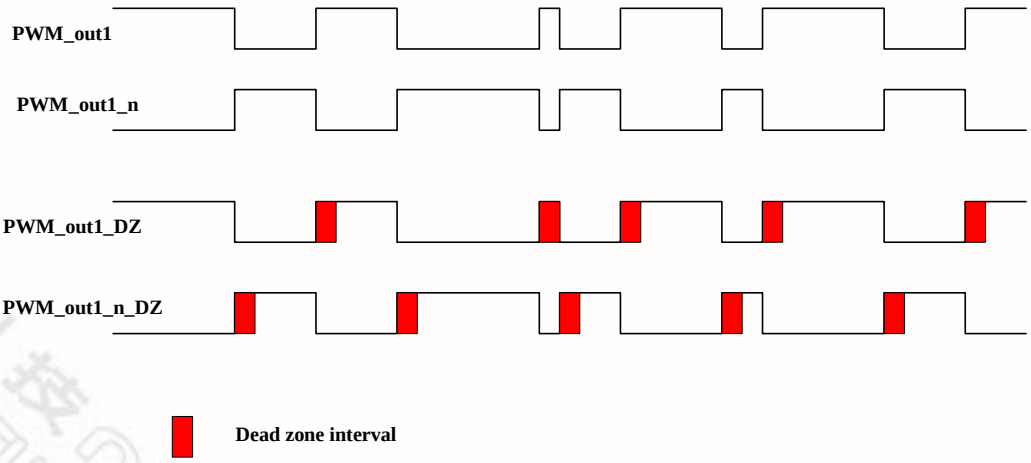
Modulate PWM controller ouput duty ratio(PWM_CNR = 150)




### 7.18.5 Dead Zone Generator

The PWM Controller is implemented with Dead Zone generator. They are built for power device protection. This function enables generation of a programmable time gap at the rising of PWM output waveform. User can program PWM_PPR [31:24] and PWM_PPR [23:16] to determine the Dead Zone interval.

Dead zone generator operation




### 7.18.6 PWM Register Map

| Register | Address | R/W | Description | Reset value |
|---|---|---|---|---|
| **PPR** | 0xB800_7000 | R/W | PWM Pre-scale Register 0 | 0000_0000 |
| **CSR** | 0xB800_7004 | R/W | PWM Clock Select Register | 0000_0000 |
| **PCR** | 0xB800_7008 | R/W | PWM Control Register | 0000_0000 |

| Register | Address | R/W | Description | Reset value |
|---|---|---|---|---|
| **CNR0** | 0xB800_700C | R/W | PWM Counter Register 0 | 0000_0000 |
| **CMR0** | 0xB800_7010 | R/W | PWM Comparator Register 0 | 0000_0000 |
| **PDR0** | 0xB800_7014 | R | PWM Data Register 0 | 0000_0000 |
| **CNR1** | 0xB800_7018 | R/W | PWM Counter Register 1 | 0000_0000 |
| **CMR1** | 0xB800_701C | R/W | PWM Comparator Register 1 | 0000_0000 |
| **PDR1** | 0xB800_7020 | R | PWM Data Register 1 | 0000_0000 |
| **CNR2** | 0xB800_7024 | R/W | PWM Counter Register 2 | 0000_0000 |
| **CMR2** | 0xB800_7028 | R/W | PWM Comparator Register 2 | 0000_0000 |
| **PDR2** | 0xB800_702C | R | PWM Data Register 2 | 0000_0000 |
| **CNR3** | 0xB800_7030 | R/W | PWM Counter Register 3 | 0000_0000 |
| **CMR3** | 0xB800_7034 | R/W | PWM Comparator Register 3 | 0000_0000 |
| **PDR3** | 0xB800_7038 | R | PWM Data Register 3 | 0000_0000 |
| **PIER** | 0xB800_703C | R/W | PWM Timer Interrupt Enable Register | 0000_0000 |
| **PIIR** | 0xB800_7040 | R/C | PWM Timer Interrupt Identification Register | 0000_0000 |

## PWM Pre-Scale Register (PPR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| PPR | 0xB800_7000 | R/W | PWM Pre-scale Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DZL1 | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DZL0 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Pre-Scale23 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Pre-Scale01 | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [31:24] | **DZL1** | **Dead Zone Length Register 1.**<br>Inserted data range: 255~0.<br>(Unit : One PWM clock cycle)<br>If DZL1=0, then Dead zone length = 0 |
| [23:16] | **DZL0** | **Dead Zone Length Register 0.**<br>Inserted data range: 255~0.<br>(Unit : One PWM clock cycle)<br>If DZL0=0, then Dead zone length = 0 |
| [15:8] | **Pre-Scale23** | **Pre-scale register for Channel 2 & 3.**<br>Pre-scale output clock frequency = PCLK / (pre-scale23 + 1)<br>If PPR=0, then the pre-scale output clock will be stopped. |
| [7:0] | **Pre-Scale01** | **Pre-scale register for Channel 0 & 1.**<br>Pre-scale output clock frequency = PCLK / (pre-scale01 + 1)<br>If PPR=0, then the pre-scale output clock will be stopped. |

## PWM Clock Selector Register (CSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CSR | 0xB800_7004 | R/W | PWM Clock Selector Register (CSR) | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | CH3 | | | Reserved | CH2 | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | CH1 | | | Reserved | CH0 | | |

| Bits | Descriptions | |
|---|---|---|
| [14:12] | **CH3** | **Channel 3 Clock Source Selection**<br>Select PWM clock source for PWM timer channel 3 |

| CH3[14:12] | Pre-scale Output Divide by |
|---|---|
| 100 | 1 |
| 011 | 16 |
| 010 | 8 |
| 001 | 4 |
| 000 | 2 |

| | | |
|---|---|---|
| [10:8] | **CH2** | **Channel 2 Clock Source Selection**<br>Select PWM clock source for PWM timer channel 2<br>(Table is the same as CH3) |
| [6:4] | **CH1** | **Channel 1 Clock Source Selection**<br>Select PWM clock source for PWM timer channel 1<br>(Table is the same as CH3) |

| Bits | Descriptions | |
|---|---|---|
| [2:0] | **CH0** | **Channel 0 Clock Source Selection**<br>Select PWM clock source for PWM timer channel 0<br>(Table is the same as CH3) |

## PWM Control Register (PCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| PCR | 0xB800_7008 | R/W | PWM Control Register (PCR) | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | CH3MOD | CH3INV | Reserved | CH3EN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CH2MOD | CH2INV | Reserved | CH2EN | CH1MOD | CH1INV | Reserved | CH1EN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | DZ1EN | DZ0EN | CH0MOD | CH0INV | Reserved | CH0EN |

| Bits | Descriptions | |
|---|---|---|
| [19] | **CH3MOD** | **Channel 3 Toggle/One-Shot Mode**<br>1: Toggle Mode<br>0: One-Shot Mode |
| [18] | **CH3INV** | **Channel 3 Inverter ON/OFF**<br>1: Inverter ON<br>0: Inverter OFF |
| [16] | **CH3EN** | **Channel 3 Enable/Disable**<br>1: Enable<br>0: Disable |
| [15] | **CH2MOD** | **Channel 2 Toggle/One-Shot Mode**<br>1: Toggle Mode<br>0: One-Shot Mode |
| [14] | **CH2INV** | **Channel 2 Inverter ON/OFF**<br>1: Inverter ON<br>0: Inverter OFF |

| Bits | Descriptions | |
|---|---|---|
| [12] | **CH2EN** | **Channel 2 Enable / Disable**<br>1: Enable<br>0: Disable |
| [11] | **CH1MOD** | **Channel 1 Toggle / One-Shot Mode**<br>1: Toggle Mode<br>0: One-Shot Mode |
| [10] | **CH1INV** | **Channel 1 Inverter ON / OFF**<br>1: Inverter ON<br>0: Inverter OFF |
| [8] | **CH1EN** | **Channel 1 Enable / Disable**<br>1: Enable<br>0: Disable |
| [5] | **DZ1EN** | **Dead-Zone 1 Generator Enable / Disable**<br>1: Enable<br>0: Disable |
| [4] | **DZ0EN** | **Dead-Zone 0 Generator Enable / Disable**<br>1: Enable<br>0: Disable |
| [3] | **CH0MOD** | **Channel 0 Toggle / One-Shot Mode**<br>1: Toggle Mode<br>0: One-Shot Mode |
| [2] | **CH0INV** | **Channel 0 Inverter ON / OFF**<br>1: Inverter ON<br>0: Inverter OFF |
| [0] | **CH0EN** | **Channel 0 Enable / Disable**<br>1: Enable<br>0: Disable |

## PWM Counter Register 3-0 (CNR3-0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CNR0** | 0xB800_700C | R/W | PWM Counter Register 0 | 0x0000_0000 |
| **CNR1** | 0xB800_7018 | R/W | PWM Counter Register 1 | 0x0000_0000 |
| **CNR2** | 0xB800_7024 | R/W | PWM Counter Register 2 | 0x0000_0000 |
| **CNR3** | 0xB800_7030 | R/W | PWM Counter Register 3 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **CNR** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **CNR** | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **CNR** | **PWM Counter/Timer Loaded Value**<br>Inserted data range : 65535~0<br>(Unit : 1 PWM clock cycle)<br><br>Note 1: One PWM cycle width = CNR + 1.<br>If CNR equal zero, PWM counter/timer will be stopped.<br>Note 2: Programmer can feel free to write a data to CNR at any time, and it will take effect in next cycle. |

## PWM Comparator Register 3-0 (CMR3-0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CMR0** | 0xB800_7010 | R/W | PWM Comparator Register 0 | 0x0000_0000 |
| **CMR1** | 0xB800_701C | R/W | PWM Comparator Register 1 | 0x0000_0000 |
| **CMR2** | 0xB800_7028 | R/W | PWM Comparator Register 2 | 0x0000_0000 |
| **CMR3** | 0xB800_7034 | R/W | PWM Comparator Register 3 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **CMR** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **CMR** | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **CMR** | **PWM Comparator Register**<br>Inserted data range : 65535~0<br>(Unit : 1 PWM clock cycle)<br>Assumption : PWM output initial : high<br>CMR >= CNR : PWM output is always high<br>CMR < CNR : PWM output high => (CMR + 1) unit<br>CMR = 0 : PWM output high => 1 unit |

## PWM Data Register 3-0 (PDR 3-0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PDR0** | 0xB800_7014 | R | PWM Data Register 0 | 0x0000_0000 |
| **PDR1** | 0xB800_7020 | R | PWM Data Register 1 | 0x0000_0000 |
| **PDR2** | 0xB800_702C | R | PWM Data Register 2 | 0x0000_0000 |
| **PDR3** | 0xB800_7038 | R | PWM Data Register 3 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| PDR | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| PDR | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [15:0] | **PDR** | **PWM Data Register**<br>PDR means the PWM counter number. |

## PWM Interrupt Enable Register (PIER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| PIER | 0xB800_703C | R/W | PWM Interrupt Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | PIER3 | PIER2 | PIER1 | PIER0 |

| Bits | Descriptions | |
|---|---|---|
| [3] | PIER3 | PWM Timer Channel 3 Interrupt Enable<br>1: Enable<br>0: Disable |
| [2] | PIER2 | PWM Timer Channel 2 Interrupt Enable<br>1: Enable<br>0: Disable |
| [1] | PIER1 | PWM Timer Channel 1 Interrupt Enable<br>1: Enable<br>0: Disable |
| [0] | PIER0 | PWM Timer Channel 0 Interrupt Enable<br>1: Enable<br>0: Disable |

## PWM Interrupt Indication Register (PIIR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| PIIR | 0xB800_7040 | R/W | PWM Interrupt Indication Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | PIIR3 | PIIR2 | PIIR1 | PIIR0 |

| Bits | Descriptions | |
|---|---|---|
| [3] | PIIR3 | PWM Timer Channel 3 Interrupt Flag<br>1: Interrupt Flag ON<br>0: Interrupt Flag OFF |
| [2] | PIIR2 | PWM Timer Channel 2 Interrupt Flag<br>1: Interrupt Flag ON<br>0: Interrupt Flag OFF |
| [1] | PIIR1 | PWM Timer Channel 1 Interrupt Flag<br>1: Interrupt Flag ON<br>0: Interrupt Flag OFF |
| [0] | PIIR0 | PWM Timer Channel 0 Interrupt Flag<br>1: Interrupt Flag ON<br>0: Interrupt Flag OFF |
| **Note: User can clear each interrupt flag by writing a zero to corresponding bit in PIIR** | | |

## 7.19 Keypad Interface (KPI)

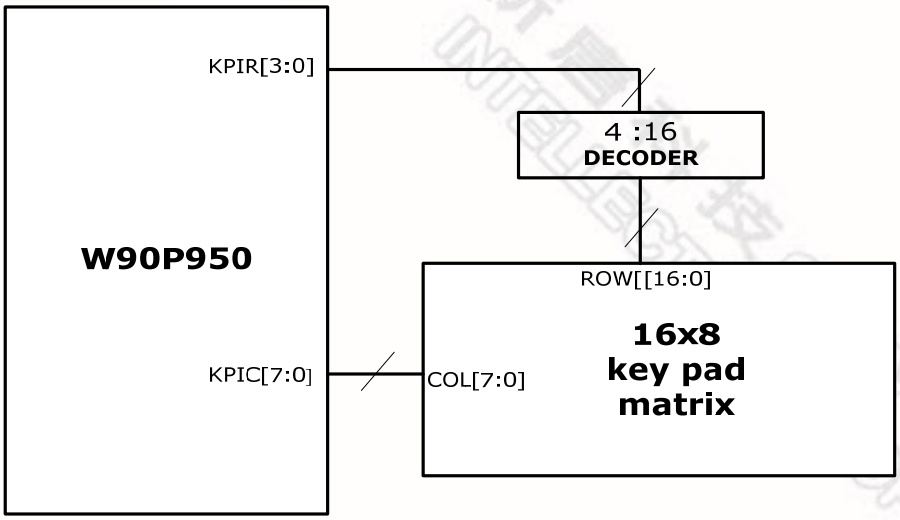
The Keypad Interface (**KPI**) is an APB slave with 4-row scan output and 8-column scan input. KPI scans an array up to 16x8 with an external 4 to 16 decoder. It can also be programmed to scan 8x8 or 4x8 key array. If the 4x8 array is selected then external decoder is not necessary because the scan signals are driven by chip itself. Any 1 or 2 keys in the array that pressed are de-bounced and encoded. If more than 2 keys are pressed, only the keys or apparent keys in the array with the lowest address will be decoded.

The KPI supports 2-keys scan interrupt and specified 3-keys interrupt or chip reset. If the 3 pressed keys matches with the 3 keys defined in **KPI3KCONF**, it will generate an interrupt or chip reset to nWDOG reset output depend on the **ENRST** setting. The interrupt is generated whenever the scanner detects a key is pressed and then after the key is released. The interrupt conditions are 1 key, or 2 keys and no keys.

This chip provides one keypad connecting interface, which is allocated in GPIOC interface and shared with NAND Flash Interface.

The keypad interface has the following features:

- maximum 16x8 array with an external 4 to 16 decoder
- programmable de-bounce time
- low-power wakeup mode for 4x8 array
- programmable three-key reset




**Keypad Interface**

### 7.19.1 Keypad Interface Register Map

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| **KPI_BA = 0xB800_8000** | | | | |
| **KPICONF** | 0xB800_8000 | R/W | Keypad controller configuration Register | 0x0000_0000 |
| **KPI3KCONF** | 0xB800_8004 | R/W | Keypad controller 3-keys configuration register | 0x0000_0000 |
| **KPILPCONF** | 0xB800_8008 | R/W | Keypad controller low power configuration register | 0x0000_0000 |
| **KPISTATUS** | 0xB800_800C | R | Keypad controller status register | 0x0000_0000 |

## Keypad Controller Configuration Register (KPI_CONF)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| KPICONF | 0xB800_8000 | R/W | Keypad configuration register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | KPSEL | ENKP | KSIZE | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DBTC | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PRESCALE | | | | | | | |

| Bits | Descriptions | |
|---|---|---|
| [19] | **KPSEL** | **Keypad Select**<br>This bit has to write 0 to select GPIOC interface to be the keypad interface. |
| [18] | **ENKP** | **Keypad Scan Enable**<br>Setting this bit high enable the key scan function.<br>1 = Enable keypad scan<br>0 = Disable keypad scan |
| [17:16] | **KSIZE** | **Key Array Size**<br><br>KSIZE: Key array size<br>00: 4x8, 3x8, 2x8, 1x8<br>01: 8x8, 7x8, 6x8, 5x8<br>1x: 16x8, 15x8, 14x8, 13x8, 12x8, 11x8, 10x8, 9x8 |
| [15:8] | **DBTC** | **De-bounce Terminal Count**<br>De-bounce counter counts the number of consecutive scans that decoded the same keys. When de-bounce counter is equal to terminal counter, it will generate a key scan interrupt. |

| Bits | Descriptions | |
|---|---|---|
| [7:0] | **PRESCALE** | **Row Scan Cycle Pre-scale Value**<br>This value is used to pre-scale row scan cycle. The pre-scale counter is clocked by 0.9375MHz clock.<br>Key array scan time = 1.067us x **PRESCALE** x16 ROWS<br>Example scan time for PRESCALE = 0xFA<br>Scan time = 1.067us x 250 x16 = 4.268ms<br>If de-bounce terminal count = 0x05, key detection interrupt is fired in approximately 21.34ms.<br>The array scan time can range from 17.07us to 1.118 sec. |

## Keypad Controller 3-keys configuration Register (KPI3KCONF)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| KPI3KCONF | 0xB800_8004 | W/R | Three-key configuration register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| | | | | | | EN3KY | ENRST |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | K32R | | | | K32C | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | K31R | | | | K31C | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | K30R | | | | K30C | | |

| Bits | Descriptions | |
|---|---|---|
| [25] | **EN3KY** | **Enable Three-keys Detection**<br>Setting this bit enables hardware to detect 3 keys specified by software |
| [24] | **ENRST** | **Enable Three-key Reset**<br>Setting this bit enable hardware reset when three-key is detected<br><br>EN3KY / ENRST / Function:<br>0 / x / Three-key function is disable<br>1 / 0 / Generate three-key interrupt<br>1 / 1 / Hardware reset by three-key-reset |
| [22:19] | **K32R** | **The #3 Key Row Address** |
| [18:16] | **K32C** | **The #3 Key Column Address** |
| [14:11] | **K31R** | **The #2 Key Row Address** |
| [10:8] | **K31C** | **The #2 Key Column Address** |
| [6:3] | **K30R** | **The #1 Key Row Address** |
| [2:0] | **K30C** | **The #1 Key Column Address** |

## Keypad Interface Low Power Mode Configuration Register (KPILPCONF)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| KPILPCOF | 0xB800_8008 | W/R | Low power configuration register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | WAKE |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LPWCEN | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | LPWR | | | |

| Bits | Descriptions | |
|---|---|---|
| [16] | **WAKE** | **Lower Power Wakeup Enable**<br>Setting this bit enables low power wakeup<br>1 = Wakeup enable<br>0 = Not enable |
| [15:8] | **LPWCEN** | **Low Power Wakeup Column Enable**<br>Enable column[7:0] low power wakeup |
| [3:0] | **LPWR** | **Low Power Wakeup Row Address**<br>Define the row address keys used to wakeup |

## Key Pad Interface Status Register (KPISTATUS)

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| KPISTATUS | 0xB800_800C | R/O | key pad status register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | INT | 3KRST | PDWAKE | 3KEY | 2KEY | 1KEY |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | KEY1R | | | | KEY1C | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | KEY0R | | | | KEY0C | | |

| Bits | Descriptions | |
|---|---|---|
| [21] | INT | **Key Interrupt**<br>This bit indicates the key scan interrupt is active and that one or two keys have changed status. |
| [20] | 3KRST | **3-Keys Reset Flag**<br>This bit will be set after 3-keys reset occur.<br>1 = 3 keys reset<br>0 = Not reset |
| [19] | PDWAKE | **Power Down Wakeup Flag**<br>This flag indicates the chip is wakeup from power down by keypad<br>1 = Wakeup up by keypad<br>0 = Not wakeup |
| [17] | 2KEY | **Double-key Press**<br>This bit indicates that 2 keys have been detected. |
| [16] | 1KEY | **Single-key Press**<br>This bit indicates that 1 key has been detected. |
| [14:11] | KEY1R | **KEY1 Row Address**<br>This value indicates key1 row address |

| Bits | Descriptions | |
|---|---|---|
| [10:8] | **KEY1C** | **KEY1 Column Address**<br>This value indicates key1 column address |
| [6:3] | **KEY0R** | **KEY0 Row Address**<br>This value indicates key0 row address |
| [2:0] | **KEY0C** | **KEY0 Column Address**<br>This value indicates key0 column address. |

# 8. ELECTRICAL SPECIFICATIONS

## 8.1 Absolute Maximum Ratings

| | |
|---|---|
| Ambient temperature | -20 °C ~ 70 °C |
| Storage temperature | -50 °C ~ 125°C |
| Voltage on any pin | -0.5V ~ 6V |
| Power supply voltage (Core logic) | -0.5V ~ 2.5V |
| Power supply voltage (IO Buffer) | -0.5V ~ 4.6V |
| Injection current (latch-up testing) | 100mA |
| Crystal Frequency | 4MHz ~ 30MHz |

## 8.2 DC Specifications

### 8.2.1 Digital DC Characteristics

(Normal test conditions: VDD33 = 3.3V+/- 10%, VDD18/PLLVDD18 = 1.8V+/- 10%, USBVDDC0/USBVDDC1/USBVDDT0/USBVDDT1= 3.3V+/-5%, TA = -20 °C ~ 70 °C unless otherwise specified)

| Symbol | Parameter | Condition | Min | TYP | Max | Unit |
|---|---|---|---|---|---|---|
| VDD33 | Power Supply | | 2.97 | - | 3.63 | V |
| VDD18/ PLLVDD18 | Power Supply | | 1.62 | - | 1.98 | V |
| USBVDDC0/ USBVDDC1/ USBVDDT0/ USBVDDT1 | Power Supply | | 3.13 | - | 3.46 | V |
| $V_{IL}$ | Input Low Voltage | | -0.3 | - | 0.8 | V |
| $V_{IH}$ | Input High Voltage | | 2.0 | - | 5.5 | V |
| VT+ | Schmitt Trigger positive-going threshold | | 1.5 | - | 1.62 | V |
| VT- | Schmitt trigger negative-going threshold | | 1.14 | - | 1.27 | V |
| $V_{OL}$ | Output Low Voltage | Depend on driving | - | - | 0.4 | V |
| $V_{OH}$ | Output High Voltage | Depend on driving | 2.4 | - | - | V |
| $I_{IH}$ | Input High Current | $V_{IN}$ = 2.4 V | -1 | - | 1 | uA |

| | | | | | | |
|---|---|---|---|---|---|---|
| $I_{IL}$ | Input Low Current | $V_{IN}$ = 0.4 V | -1 | - | 1 | uA |
| $I_{OH}$ | Output High Current | EBI, GPIOC, GPIOD | - | 35 | - | mA |
| $I_{OL}$ | Output Low Current | EBI, GPIOC, GPIOD | - | 26 | - | mA |
| $I_{OH}$ | Output High Current | The other port | - | 25 | - | mA |
| $I_{OL}$ | Output Low Current | The other port | - | 17 | - | mA |
| $I_{OC}$ | Operation Current | Note 1 | - | 340 | - | mA |
| $I_{SC}$ | Standby Current | Note 2 | - | 100 | - | uA |

Note1:

This operation current is measured on VDD18 @ 1.8V, and all of IP clocks are enable with CPU clock/system clock @ 200MHz / 100MHz.

Note2:

The standby current is measured on VDD18 @1.8V, and all of IP clocks are disabled with power-down mode, all of GPIO pins are set to output and clock pins keep at 0V.

### 8.2.2 USB Low-/Full-Speed DC Electrical Specifications

| Symbol | Parameter | Conditions | Min | Nom | Max |
|---|---|---|---|---|---|
| $V_{IH}$ | Pad input high voltage | | 2.0V | | |
| $V_{IL}$ | Pad input low voltage | | | | 0.8V |
| $V_{DI}$ | Differential input sensitivity | \|PADP-PADM\| | 0.2V | | |
| $V_{CM}$ | Common mode voltage range | include $V_{DI}$ range | 0.8V | | 2.5V |
| $V_{SE}$ | Single-ended receiver threshold | | 0.8V | | 2.0V |
| $V_{OL}$ | Pad output low voltage | | 0V | | 0.3V |
| $V_{OH}$ | Pad output high voltage | | 2.8V | | 3.6V |
| $V_{CRS}$ | Differential output signal cross-point voltage | | 1.3V | | 2.0V |
| $R_{PU}$ | Internal pull-up resistor | Bus idle | 900Ω | | 1575Ω |
| | | Receiving | 1425Ω | | 3090Ω |
| $R_{PD}$ | Internal pull-down resistor | | 14.25KΩ | | 24.80KΩ |
| $Z_{DRV}$ | Driver output resistance‡ | Steady state drive | | 10Ω | |
| $C_{IN}$ | Transceiver pad capacitance | Pad to ground | | | 20pF |

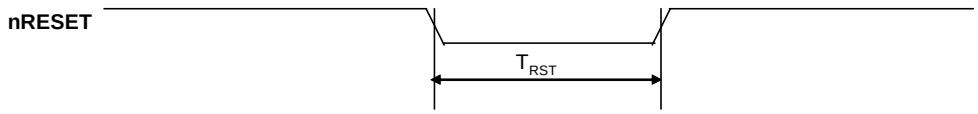
### 8.2.3 USB High-Speed DC Electrical Specifications

| Symbol | Parameter | Conditions | Min | Typ | Max |
|---|---|---|---|---|---|
| $V_{HSDI}$ | High-speed differential input signal level | \|PADP-PADM\| | 150mV | | |
| $V_{HSSQ}$ | High-speed SQ detection threshold | \|PADP-PADM\| | 100mV | | 150mV |
| $V_{HSCM}$ | High-speed common mode voltage range | | -50mV | | 500mV |

| | | | | | |
|---|---|---|---|---|---|
| $V_{HSOH}$ | High-speed data signaling high | | 360mV | | 440mV |
| $V_{HSOL}$ | High-speed data signaling low | | -10mV | | 10mV |
| $V_{CHIRPJ}$ | Chirp J level | | 700mV | | 1100mV |
| $V_{CHIRPK}$ | Chirp K level | | -900mV | | -500mV |
| $Z_{HSDRV}$ | High-speed driver output resistance | 45Ω±10% | 40.5Ω | | 49.5Ω |

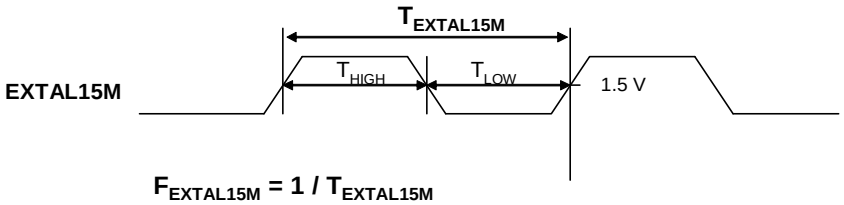
## 8.3 AC Specifications

### 8.3.1 RESET AC Characteristics




| Symbol | Parameter | Min. | Max. | Unit |
|---|---|---|---|---|
| $T_{RST}$ | Reset Pulse Width after Power stable | 1.0 | - | ms |

### 8.3.2 Clock Input Characteristics



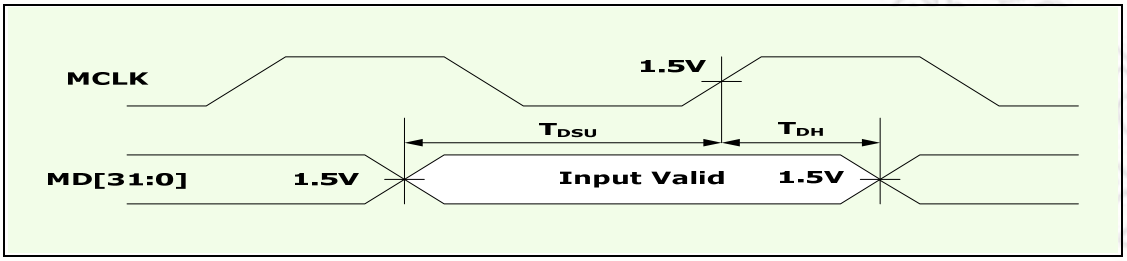

$F_{EXTAL15M} = 1 / T_{EXTAL15M}$

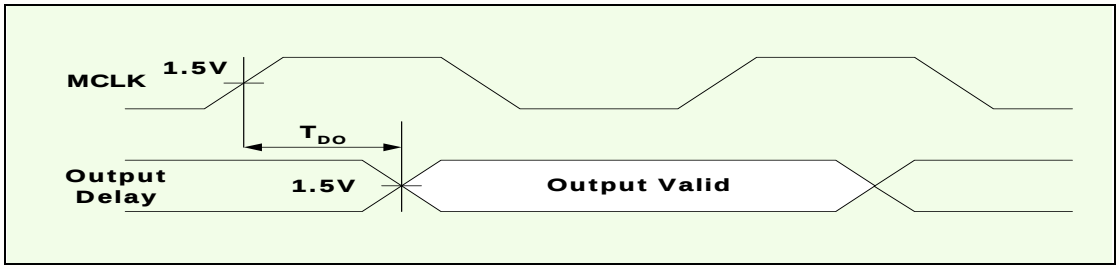
$EXTAL15M_{DUTY} = T_{HIGH} / (T_{HIGH} + T_{LOW})$

| Symbol | Parameter | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $F_{EXTAL15M}$ | Clock Input Frequency | - | 15.0 | - | MHz |
| $EXTAL15M_{DUTY}$ | Clock Input Duty Cycle | 45 | 50 | 55 | % |

| | | | | | |
|---|---|---|---|---|---|
| **$V_{IL}$ (EXTAL15M)** | EXTAL15M Input Low Voltage | 0 | - | 0.8 | V |
| **$V_{IH}$ (EXTAL15M)** | EXTAL15M Input High Voltage | 2.0 | - | VDD33+0.3 | V |

### 8.3.3 EBI/SDRAM Interface AC Characteristics

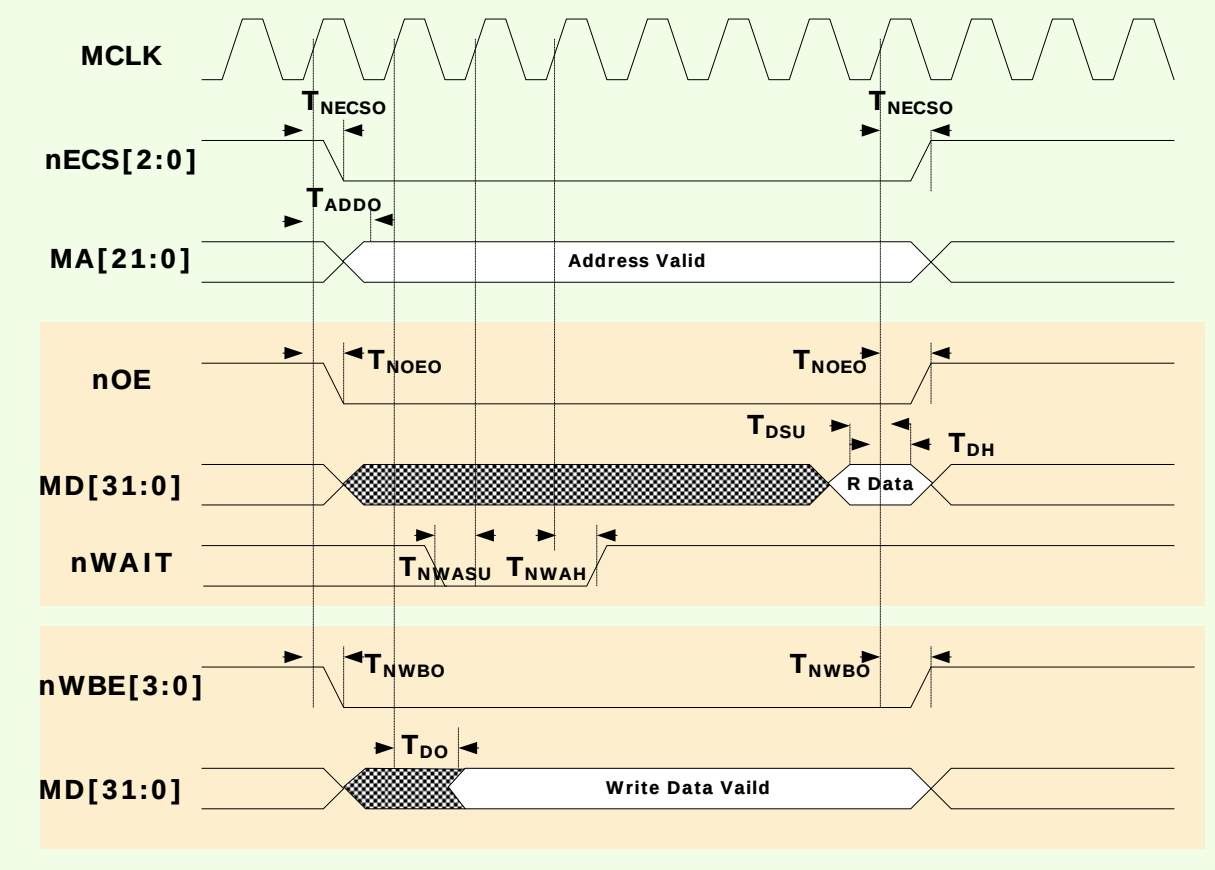



| Symbol | Parameter | Min. | Max. | Unit |
|---|---|---|---|---|
| **$F_{MCLK}$** | SDRAM Clock Output Frequency | - | 100 | MHz |
| **$T_{DSU}$** | MD[31:0]] Input Setup Time | 2 | - | ns |
| **$T_{DH}$** | MD[31:0] Input Hold Time | 2 | - | ns |
| **$T_{OSU}$** | SDRAM Output Signal Valid Delay Time | 2* | 7* | ns |

* The above $T_{OSU}$ is based on the EBI CKSKEW register default setting on 0x48 and **$F_{MCLK}$** at 100MHz

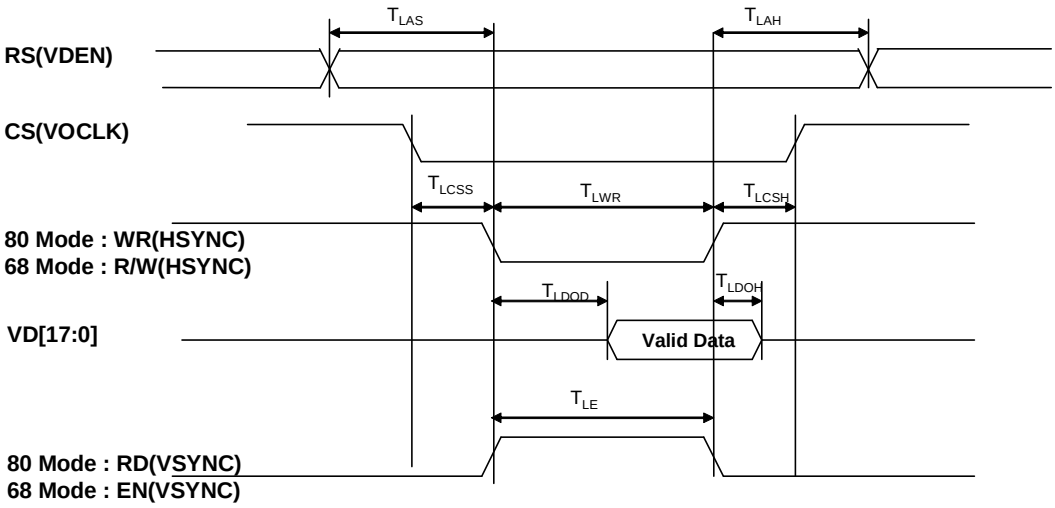
### 8.3.4 EBI / (ROM/SRAM/External I/O) AC Characteristics




| Symbol | Parameter | Min. | Max. | Unit |
|---|---|---|---|---|
| $T_{ADDO}$ | Address Output Delay Time | 2* | 7* | ns |
| $T_{NCSO}$ | ROM/SRAM/Flash or External I/O Chip Select Delay Time | 2* | 7* | ns |
| $T_{NOEO}$ | ROM/SRAM or External I/O Bank Output Enable Delay | 2* | 7* | ns |
| $T_{NWBO}$ | ROM/SRAM or External I/O Bank Write Byte Enable Delay | 2* | 7* | ns |
| $T_{DH}$ | Read Data Hold Time | 5 | | ns |
| $T_{DSU}$ | Read Data Setup Time | 1 | | ns |
| $T_{DO}$ | Write Data Output Delay Time (SRAM or External I/O) | 2* | 7* | ns |
| $T_{NWASU}$ | External Wait Setup Time | 3 | | ns |
| $T_{NWAH}$ | External Wait Hold Time | 1 | | ns |

The above data are based on the EBI CKSKEW register default setting on 0x48 and $F_{MCLK}$ at 100MHz

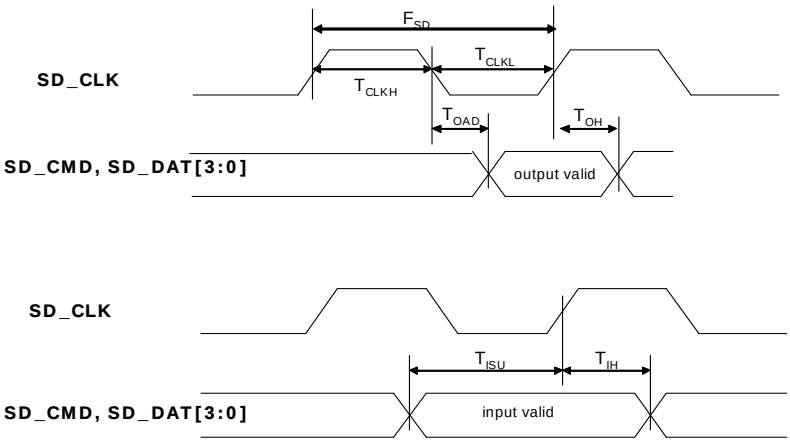
### 8.3.5 LCD Interface: MPU Type AC Characteristics




| Symbol | Parameter | Conditions | Min. | Max. | Unit |
|---|---|---|---|---|---|
| $T_{LCSS}$ | Chip Select Set-up Time | - | 1/2 | - | *PCLK |
| $T_{LCSH}$ | Chip Select Hold Time | - | 1/2 | - | *PCLK |
| $T_{LAS}$ | Address Set-up Time | - | 1 | - | *PCLK |
| $T_{LAH}$ | Address Hold Time | - | 1 | - | *PCLK |
| $T_{LDOD}$ | Write Data Active Delay | - | 0 | 1/2 | *PCLK |
| $T_{LDOH}$ | Write Data Hold Time | - | 1/2 | - | *PCLK |
| $T_{LWR}$ | WR Pulse Width | 80 Mode | 1 | - | *PCLK |
| $T_{LE}$ | LE Pulse Width | 68 Mode | 1/2 | - | *PCLK |

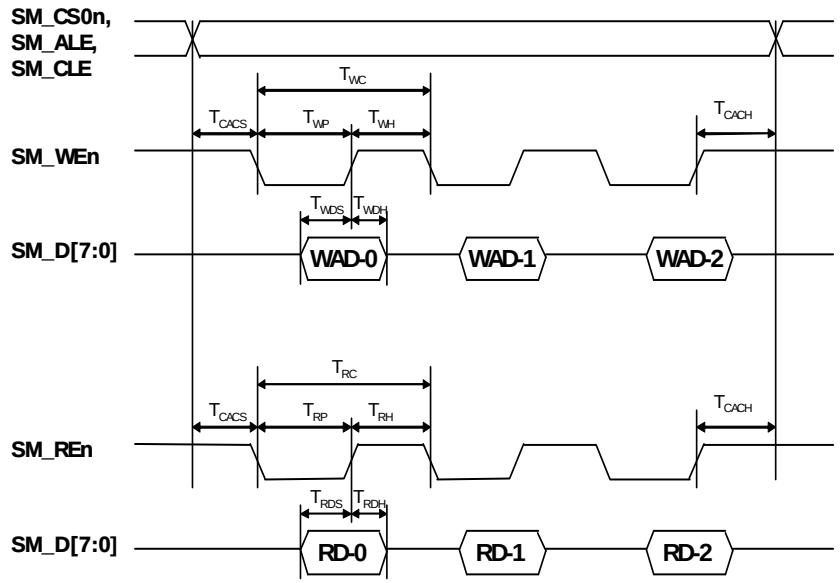
*PCLK is the engine clock of the LCD Controller

### 8.3.6 SD Host Interface AC Characteristics




| Symbol | Parameter | Conditions | Min. | Max. | Unit |
|---|---|---|---|---|---|
| $F_{SD}$ | SD Clock Frequency | Identification Mode | **100** | **400** | KHz |
| $F_{SD}$ | SD Clock Frequency | Data Transfer Mode | - | **50** | MHz |
| $T_{CLKH}$ | SD Clock High Time | - | **10** | - | ns |
| $T_{CLKL}$ | SD Clock Low Time | - | **10** | - | ns |
| $T_{ISU}$ | SD CMD & Data Input Setup Time | - | **5** | - | ns |
| $T_{IH}$ | SD CMD & Data Input Hold Time | - | **5** | - | ns |
| $T_{OAD}$ | SD Output Active Delay (Falling Edge) | - | - | **14** | ns |
| $T_{OH}$ | SD Output Hold Time | - | **0** | - | ns |

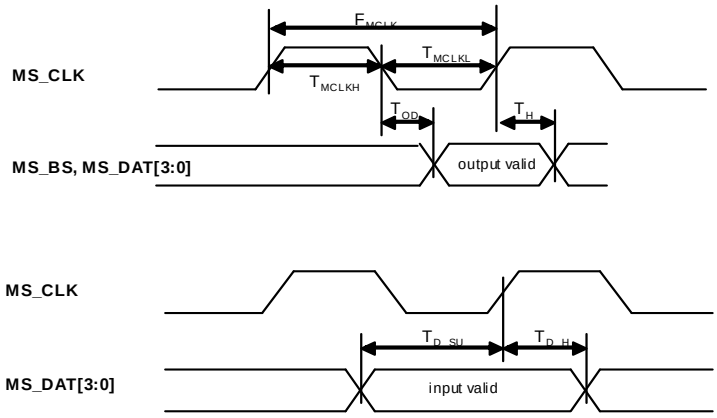
### 8.3.7 NAND Flash Memory Interface AC Characteristics




| Symbol | Parameter | Min. | Max. | Unit |
|---|---|---|---|---|
| $T_{CACS}$ | SM_CS0n, SM_ALE, SM_CLE Setup Time before SM_WEn, SM_REn Low | **20** | - | ns |
| $T_{CACH}$ | SM_CS0n, SM_ALE, SM_CLE Hold Time after SM_WEn, SM_REn High | **40** | - | ns |
| $T_{WP}$ | Write Pulse Width | **40** | - | ns |
| $T_{WH}$ | SM_WEn High Time | **20** | - | ns |
| $T_{WC}$ | Write Cycle Time | **80** | - | ns |
| $T_{WDS}$ | Write Data Output Setup Time | **30** | - | ns |
| $T_{WDH}$ | Write Data Output Hold Time | **20** | - | ns |
| $T_{RP}$ | Read Pulse Width | **60** | - | ns |
| $T_{RH}$ | SM_REn High Time | **20** | - | ns |

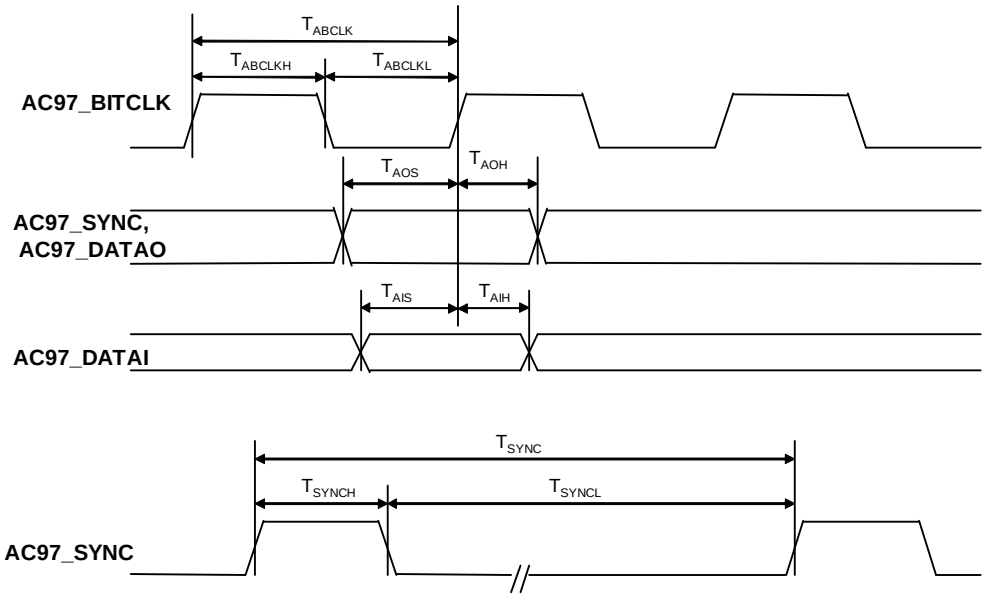
| | | | | |
|---|---|---|---|---|
| $T_{RC}$ | Read Cycle Time | **80** | - | ns |
| $T_{RDS}$ | Read Data Input Setup Time | **6** | - | ns |
| $T_{RDH}$ | Read Data Input Hold Time | **20** | - | ns |

### 8.3.8 Memory Stick Interface AC Characteristics



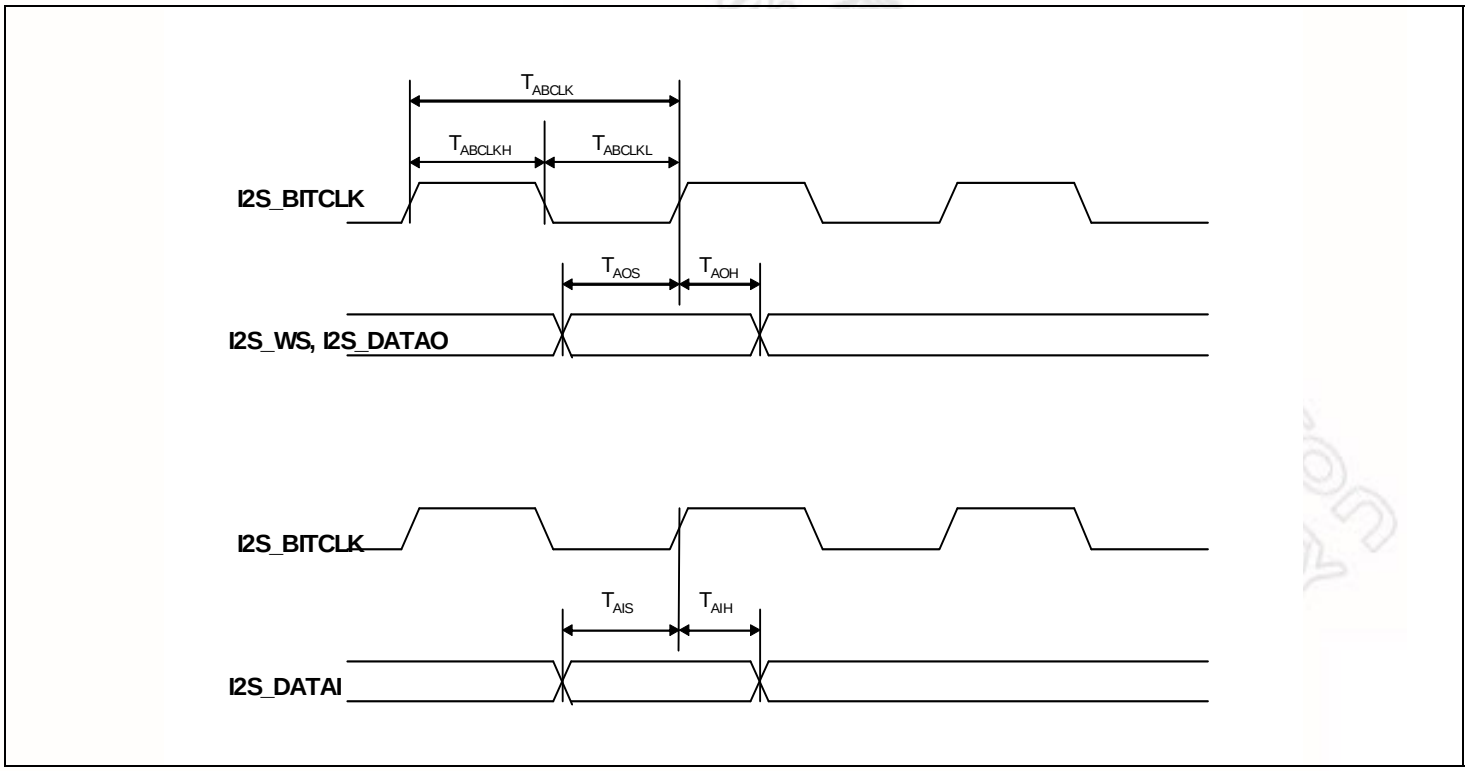

| Symbol | Parameter | Conditions | Min. | Max. | Unit |
|---|---|---|---|---|---|
| $F_{MCLK}$ | MS_CLK Clock Frequency | Serial Mode | **5** | **20** | MHz |
| $F_{MCLK}$ | MS_CLK Clock Frequency | Parallel Mode | **10** | **40** | MHz |
| $T_{MCLKH}$ | MS_CLK Clock High Time | | **5** | - | ns |
| $T_{MCLKL}$ | MS_CLK Clock Low Time | | **5** | - | ns |
| $T_{BS\_OD}$ | MS_BS Output Delay (Falling Edge) | | **5** | **15** | ns |
| $T_{BS\_H}$ | MS_BS Output Hold Time | | **1** | - | ns |
| $T_{D\_SU}$ | Data Input Setup Time | | **8** | - | ns |
| $T_{D\_H}$ | Data input Hold Time | | **1** | - | ns |
| $T_{D\_OD}$ | Data Output Delay (Falling Edge) | | **8** | **15** | ns |
| $T_{D\_OD}$ | Data Output Hold Time | | **1** | - | ns |

### 8.3.9 Audio AC-Link Interface AC Characteristics



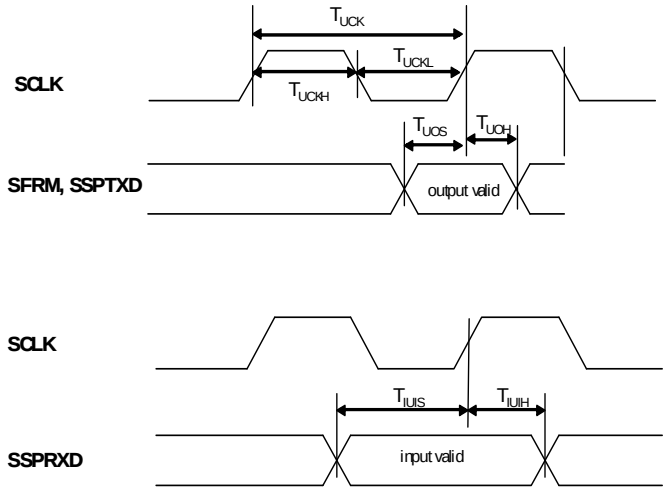

| Symbol | Parameter | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $T_{ABCLKH}$ | Audio Bit Clock Input High Time | **36.6** | **40.7** | **44.8** | ns |
| $T_{ABCLKH}$ | Audio Bit Clock Input Low Time | **36.6** | **40.7** | **44.8** | ns |
| $T_{ABCLK}$ | Audio Bit Clock Input Cycle Time | - | **81.4** | - | ns |
| $T_{AOS}$ | Audio Output Signal (AC97_SYNC, AC97_DATAO) Setup Time | **15** | - | - | ns |
| $T_{AOH}$ | Audio Output Signal (AC97_SYNC, AC97_DATAO) Hold Time | **5** | - | - | ns |
| $T_{AIS}$ | Audio Data Input Setup Time | **15** | - | - | ns |
| $T_{AIH}$ | Audio Data Input Hold Time | **5** | - | - | ns |
| $T_{SYNCH}$ | Sync Signal Output High Time | - | **20.8** | - | ns |
| $T_{SYNCH}$ | Sync Signal Output Low Time | - | **1.3** | - | ns |
| $T_{SYNC}$ | Sync Signal Output Cycle Time | - | **19.5** | - | ns |

### 8.3.10 Audio I2S Interface AC Characteristics




| Symbol | Parameter | Min. | Max. | Unit |
|---|---|---|---|---|
| $T_{ABCLKH}$ | Audio Bit Clock Output High Time | **18.3** | - | ns |
| $T_{ABCLKH}$ | Audio Bit Clock Output Low Time | **18.3** | - | ns |
| $T_{ABCLK}$ | Audio Bit Clock Output Cycle Time | **40.7** | - | ns |
| $T_{AOS}$ | Audio Data Output Setup Time | **4.5** | - | ns |
| $T_{AOH}$ | Audio Data Output Hold Time | **4.5** | - | ns |
| $T_{AIS}$ | Audio Data Input Setup Time | **4.5** | - | ns |
| $T_{AIH}$ | Audio Data Input Hold Time | **4.5** | - | ns |

### 8.3.11 USI (SPI/MW) Interface AC Characteristics




| Symbol | Parameter | Min. | Max. | Unit |
|---|---|---|---|---|
| $T_{CLKH}$ | Clock Output High Time | **14.6** | - | ns |
| $T_{CLKL}$ | Clock Output Low Time | **15.8** | - | ns |
| $T_{CLK}$ | Clock Cycle Time | **30.4** | **-** | ns |
| $T_{UOS}$ | SFRM, SSPTXD Output Setup Time | **15** | - | ns |
| $T_{UOH}$ | SFRM, SSPTXD Output Hold Time | **13** | - | ns |
| $T_{UIS}$ | SSPRXD Input Setup Time | **10** | - | ns |
| $T_{UIH}$ | SSPRXD Input Hold Time | **10** | - | ns |

### 8.3.12 USB Transceiver AC Characteristics

**USB Transceiver: Low-Speed AC Electrical Specifications**

| Symbol | Parameter | Conditions | Min | Typ | Max |
|---|---|---|---|---|---|
| $T_{LR}$ | Low-speed driver rise time | $C_L$=50pF | 75ns | | 300ns |
| $T_{LF}$ | Low-speed driver fall time | $C_L$=50pF | 75ns | | 300ns |
| $T_{LRFM}$ | Low-speed rise/fall time matching | $T_{LRFM} = T_{LR} / T_{LF}$ | 80% | | 125% |

**USB Transceiver: Full-Speed AC Electrical Specifications**

| Symbol | Parameter | Conditions | Min | Typ | Max |
|---|---|---|---|---|---|
| $T_{FR}$ | Full-speed driver rise time | $C_L$=50pF | 4ns | | 20ns |
| $T_{FF}$ | Full-speed driver fall time | $C_L$=50pF | 75ns | | 20ns |
| $T_{FRFM}$ | Full-speed rise/fall time matching | $T_{FRFM} = T_{FR} / T_{FF}$ | 90% | | 111.11 % |

**USB Transceiver: High-Speed AC Electrical Specifications**

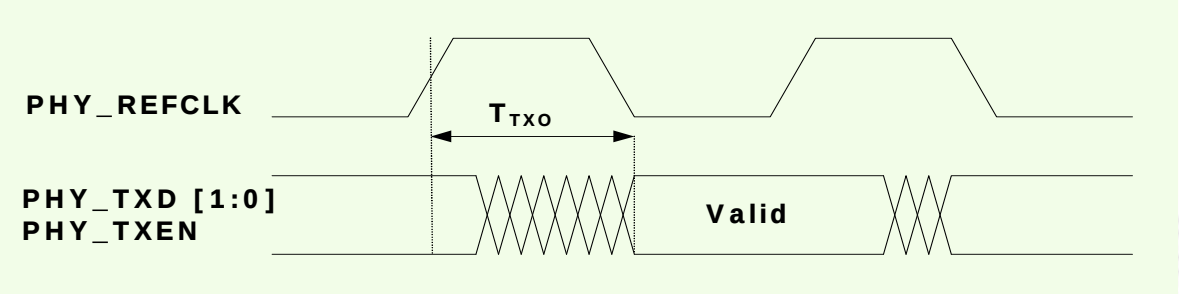
| Symbol | Parameter | Conditions | Min | Typ | Max |
|---|---|---|---|---|---|
| $T_{HSR}$ | High-speed driver rise time | $Z_{HSDRV}$=45Ω | 500ps | | 900ps |
| $T_{HSF}$ | High-speed driver fall time | $Z_{HSDRV}$=45Ω | 500ps | | 900ps |
| | High-speed driver waveform requirement | | Eye diagram of template 1** | | |
| | High-speed receiver waveform requirement | | Eye diagram of template 4†† | | |
| | High-speed jitter requirement | Data source end | Eye diagram of template 1** | | |
| | | Receiver end | Eye diagram of template 4†† | | |

** Check "Universal Serial Bus Specification Revision 2.0" in page 133.

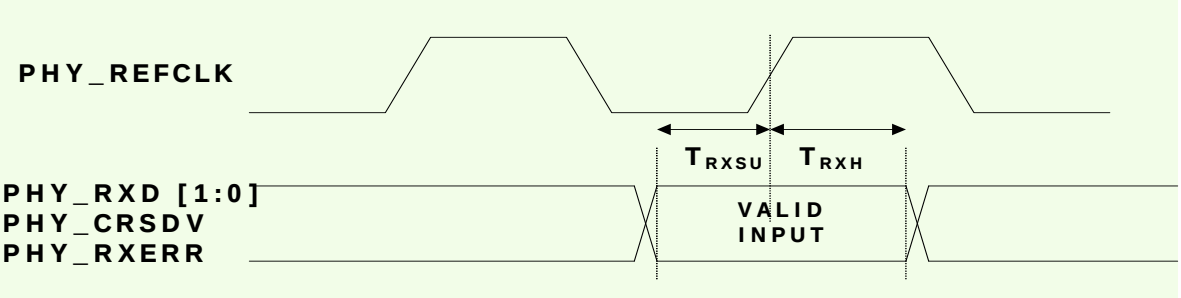
†† Check "Universal Serial Bus Specification Revision 2.0" in page 136.

### 8.3.13 EMC RMII AC Characteristics

The signal timing characteristics conforms to the guidelines specified in IEEE Std. 802.3.

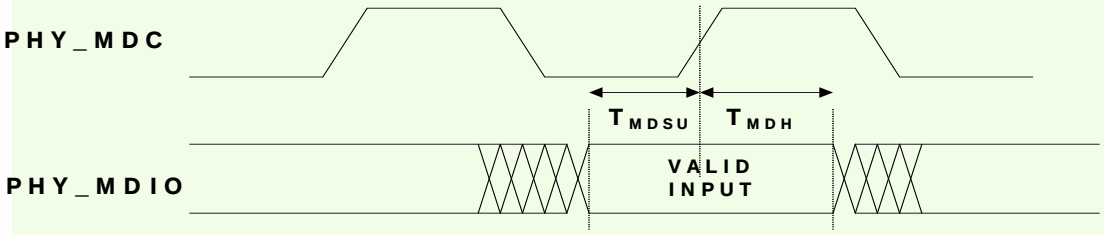



**Transmit Signal Timing Relationships at RMII**

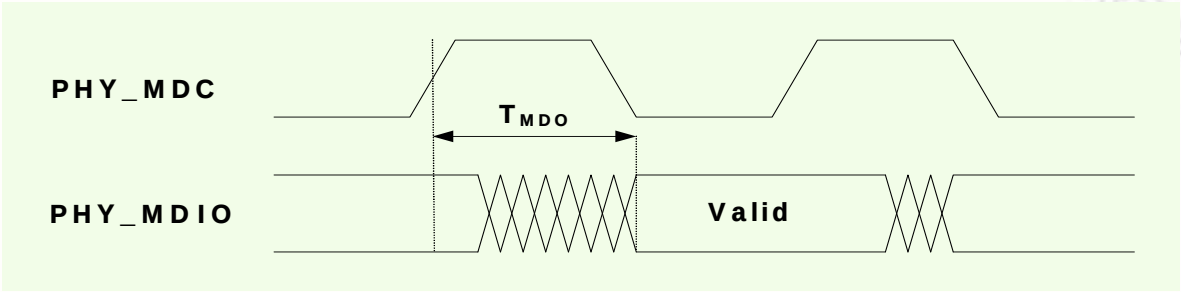



**Receive Signal Timing Relationships at RMII**

| Symbol | Parameter | Min | Max | Unit |
|---|---|---|---|---|
| $T_{TxO}$ | Transmit Output Delay Time | 7 | 14 | ns |
| $T_{RxSU}$ | Receive Setup Time | 4 | | ns |
| $T_{RxH}$ | Receive Hold Time | 2 | | ns |




**PHY_MDIO Read from PHY Timing**

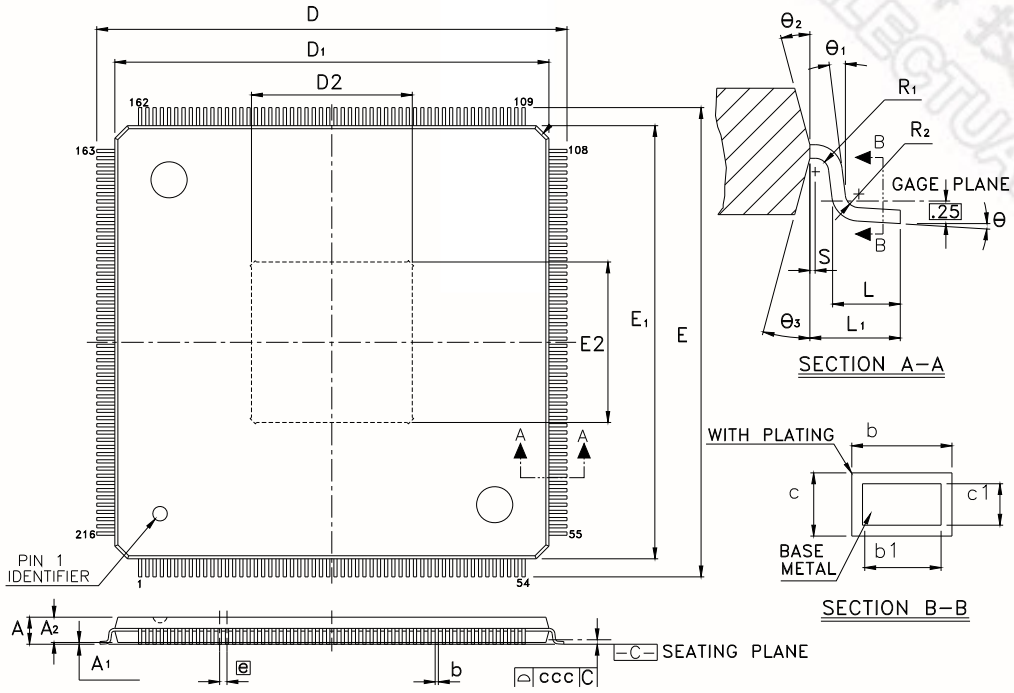



**PHY_MDIO Write to PHY Timing**

| Symbol | Parameter | Min | Max | Unit |
|---|---|---|---|---|
| $T_{MDO}$ | PHY_MDIO Output Delay Time | 0 | 15 | ns |
| $T_{MDSU}$ | PHY_MDIO Setup Time | 5 | | ns |
| $T_{MDH}$ | PHY_MDIO Hold Time | 5 | | ns |

## 9. PACKAGE SPECIFICATIONS

### NUC950ADN LQFP216L (24X24X1.4 mm, footprint 2.0mm)




| Symbol | Dimension in mm | | | Dimension in inch | | |
|---|---|---|---|---|---|---|
| | Min | Nom | Max | Min | Nom | Max |
| A | — | — | 1.60 | — | — | 0.063 |
| $A_1$ | 0.05 | — | 0.15 | 0.002 | — | 0.006 |
| $A_2$ | 1.35 | 1.40 | 1.45 | 0.053 | 0.055 | 0.057 |
| b | 0.13 | 0.18 | 0.23 | 0.005 | 0.007 | 0.009 |
| $b_1$ | 0.13 | 0.16 | 0.19 | 0.005 | 0.006 | 0.007 |
| c | 0.09 | 0.14 | 0.20 | 0.004 | 0.006 | 0.008 |
| $c_1$ | 0.09 | 0.12 | 0.16 | 0.004 | 0.005 | 0.006 |
| D | 25.85 | 26.00 | 26.15 | 1.018 | 1.024 | 1.030 |
| $D_1$ | 23.90 | 24.00 | 24.10 | 0.941 | 0.945 | 0.949 |
| E | 25.85 | 26.00 | 26.15 | 1.018 | 1.024 | 1.030 |
| $E_1$ | 23.90 | 24.00 | 24.10 | 0.941 | 0.945 | 0.949 |
| e | 0.40 BSC | | | 0.016 BSC | | |
| L | 0.45 | 0.60 | 0.75 | 0.018 | 0.024 | 0.030 |
| $L_1$ | 1.00 REF | | | 0.039 REF | | |
| $R_1$ | 0.08 | — | — | 0.003 | — | — |
| $R_2$ | 0.08 | — | — | 0.003 | — | — |
| S | 0.20 | — | — | 0.008 | — | — |
| θ | 0° | 3.5° | 7° | 0° | 3.5° | 7° |
| $θ_1$ | 0° | — | — | 0° | — | — |
| $θ_2$ | 11° | 12° | 13° | 11° | 12° | 13° |
| $θ_3$ | 11° | 12° | 13° | 11° | 12° | 13° |
| ccc | 0.08 | | | 0.003 | | |

| Exposed Pad Size | | |
|---|---|---|
| L/F | Dimension in mm | Dimension in inch |
| D2/E2 | 6.50 REF | 0.256 REF |

## 10. REVISION HISTORY

| REVISION | DATE | COMMENTS |
|---|---|---|
| A | 2008/07/09 | First Release |
| A1 | 2008/07/18 | 1. Update Chapter 7.10<br>Display size: Maximum size 1024x768 |
| A2 | 2008/10/07 | 1. Update Chapter 2<br>Add text "Pb free"<br>2. Update Chapter 4<br>Correct typo: KPI_ROW[0], KPI_ROW[1], KPI_ROW[2], KPI_ROW[3]<br>3. Correct Typo: Spelling and grammar check<br>4. Update Chapter 9<br>Package Specifications |
| A3 | 2009/04/13 | 1. Update Chapter 7.10.1.4<br>Display Pin Assignment Table<br>2. Rename nWE to nSWE<br>3. Change Part Number from W90P950CDG to NUC950ADN<br>4. Update Chapter 7.10<br>Display size: Maximum size 1024x600<br>5. Update Chapter 8.3.5<br>LCD Interface: MPU Type AC Characteristics |
| A4 | 2010/06/18 | 1. Add GPIOI[16]<br>2. Add IOH, IOL current value |

nuvoTon

# 32-BIT ARM926EJ-S BASED MCU

## Important Notice

**Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".**

**Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.**

**All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.**