ASPEED TECHNOLOGY Inc.

AST2050 / AST1100

Integrated Remote Management Processor

A3 Datasheet

Version 1.05

May 25, 2010

Headquarters
2F,No.15,Industry East Road 4.,Hsinchu Science Park,
Hsinchu City 30077, Taiwan, R. O. C.
TEL: 886-3-5789568
FAX: 886-3-5789586
http://www.ASPEEDtech.com

Part Number Information

Part number	:AST2050A3-GP
Solder ball type	:Lead-free
Substrate type	:RoHS Green package
Package size	:19mm x 19mm
Ball pitch	: 0.8mm

Topside mark

ASPEED
AST2050
XXXXXX.XX-X
WWYY TAN A3 GP

Part number	:AST1100A3-GP
Solder ball type	:Lead-free
Substrate type	:RoHS Green package
Package size	:19mm x 19mm
Ball pitch	: 0.8mm

ASPEED
AST1100
XXXXXX.XX-X
WWYY TAN A3 GP

Revision History

Date	Revision	Description
Feb. 14, 2007	0.1	Initial draft.
Mar. 23, 2007	0.2	Initial draft.
Apr. 20, 2007	0.5	Preliminary release.
Apr. 27, 2007	0.8	Normal release.
Nov. 2, 2007	0.9	1. Fixed SMBus ALT1 and ALT2 pin mapping error at Page46, exchange them. 2. Add dedicated LPC Reset function pin at Page46. 3. Modify DACRSET external pull-low resistor value to 2.43K ohm.
Jan. 22, 2008	0.91	1. Add new power up sequence description at Section 4.5. 2. Modify GPIO summary table at Section 3.5. 3. Update SCL7 ball number error at Page46, it is consist with SCL6. 4. Modify the Core operating voltage from 1.2V to 1.26V. 5. Modify the DDR-I operating voltage from 2.5V to 2.6V to meet DDR400 specification. 6. Modify PCI AD bus swap description at Page 45 7. Modify DDR/DDR2 AC timing specification at Page 69.
Mar. 05, 2008	0.92	1. Add thermal specification at Section 4.6.
Apr. 17, 2008	0.93	1. Add DAC IO specification at Page63. 2. Add power consumption information at Section 4.3.
Jul. 01, 2008	1.00	1. Rev. A2 chip released. 2. Modify the USB2.0 USBVRES pin external resistor from 6.195KΩ to 8.2KΩ. 3. Modify the description of multi-function pin mapping table for more easier understand. See Section 7. 4. Modify the feature description about Virtual UART.
Aug. 07, 2008	1.01	1. Rev. A3 chip released.
Nov. 24, 2008	1.03	1. Modify power up sequence for flash reset timing at Section 4.5.

Revision History of Software Programming Guide

Date	Revision	Description
Jan. 22, 2008	0.91	1. Add register definition at VUART20.bit[1]. 2. Add USB1.1/2.0 controller new device address updating information at USBL04, HUB04 and DEV00. 3. Add I2C software programming guide at Section 31. 4. Add software programming guide for USB2.0 controller at Section 15. 5. Update MIC register specification. 6. Update MDMA register specification. 7. Add Timer controller programming note. 8. Add Video register VR054. 9. Add a reserved and don't use register VIC30 at VIC controller.
Mar. 05, 2008	0.92	1. Remove PWM registers PTRC40 $\sim$ PTRC7C, they don't exist.
Apr. 17, 2008	0.93	1. Add description about MII management interface at Section 14.4.6. 2. Add table about GPIO interrupt trigger mode at Page269. 3. Add programming notice for USB2.0 remote wakeup function at register HUB00.bit[2]. 4. Add USB2.0 controller registers reset control table at Section 15.3.6. 5. Add programming guide for USB2.0 Hub disconnect sequence at Page 175. 6. Add programming notice about USB2.0 endpoint number assignment at Section 15.5.4.
Jul, 01, 2008	1.00	1. Add 3 programming notes for USB2.0 programming, start from Section 15.4.5. 2. Add 1 hardware limitation for USB2.0 programming at Section 15.5.5. 3. Add UART function control at SCU2C[15:12]. 4. Add VUART/PUART function control at LHCR0[12:8]. 5. Re-writing Virtual UART registers set at Section 29.
Aug. 07, 2008	1.01	1. Rev. A3 chip released.
Sep. 19, 2008	1.02	1. Modify the chip revision number (SCU7C) to A3. 2. Modify the USB2.0 clock enable delay from 3 ms to 10 ms. 3. Add a hardware limitation on the USB controller in supporting high speed high bandwidth transfer type. please reference Section 15.5.6.
Nov. 24, 2008	1.03	1. Add SOC scratch register definition at registers SCU40 and SCU44. 2. Add AC timing specification of flash interface. 3. Modify NOR flash timing diagram graph.
May. 04, 2010	1.04	1. Add RTC programming guide.
May. 25, 2010	1.05	1. Add I2C high speed mode programming guide at Section 31.6.14.

AST2050 / AST1100 Migration List from A1 to A2

Functional Bug Fixed in A2 List

1. GPIOE multi-function failed.
   - This issue is only happened at the following multi-function pin combination:
     - GPIOE[7:0](group 1) to TACH[7:0] and set GPIOE[7:0](group 2) to GPIOE[7:0].
   - GPIOE has 2 optional group pins. The first group is multi-function with FAN Tachometer input, the second group is multi-function with MAC interface. There is a chip design error on the first group pins. When the first group pins are used as FAN Tachometer input, then the second group GPIOE(2) can only be used as GPI function, GPO is not allowed.
   - This is caused by the output enable control of GPIOE will affect both pin groups at the same time. If GPIOE set to GPO mode, then both the same bit of GPIOE(1) and GPIOE(2) pins will be set to output mode, thus cause FAN Tachometer function not work.
   - Will be fixed in rev A2.
2. Some GPIO pins drive output at low state when power up.
   - Most GPIO pins stay at output tri-state/input mode when power up. So it is easily to use external pull up or down resistors to control the power up state of these pins stay at high or low voltage level. But the following GPIO pins will drive output at low state when power up, which can not be pulled high by external resistor.
     - GPIOE0(C4), GPIOE1(D4) and GPIOE4(D5) at GPIOE pin group 2.
   - These pins will keep driving low until SW changes them. It will limit the usage of these pins. They can not be used to those applications that need power up at high state.
   - Will be fixed in rev A2.
3. VUART interrupt decoding table error when Modem Status changed.
   - Modem changed interrupt status in the interrupt identity register (VUART08[3:1]) should be ”000”, but VUART returns ”001”.
   - Both the host side and the slave side of VUART will be fixed in rev A2.
4. VUART Host-Tx-discard mode.
   - This is a new kind of enable bit in the ARM side of VUART. In AST2050 / AST1100 VUART, the Host Tx and the SP(BMC) Rx share the same FIFO. The Tx data of the Host side will stay in the FIFO until the ARM receives/retrieves it. With enabling this new host-Tx-discard mode, the output data from the Host is simply discarded or is thrown away by the ARM, rather than being stuck from the Host viewpoint. This mode is designed for the ARM enabling VUART but not intending to listen the data.
   - The ARM driver can prevent the data pending symptom. SW can fix.
   - There is a new enable bit of VUART in rev A2 that can prevent the symptom. This new enable bit will throw data away automatically if the Host Tx FIFO is not empty.
5. VUART Transmitter Holding Register Empty (THRE) Interrupt missing.
   - HW should raise the THRE interrupt when an AP programs the enable bit of THRE interrupt. But this function was not implemented in revision A1.
   - Both the host side and the slave side of VUART will be fixed in rev A2.
6. VUART Modem Status Data Carrier Detect (DCD) and Delta DCD (DDCD).
   - VUART18[7] (non-loopback mode) and VUART18[3] are reserved (always zero) in revision A1. It should be DCD and DDCD in traditional UART.

- Will be fixed in rev A2. In rev A2, VUART18[7] is redefined as "Complement of the nDSR input (non-loopback mode) or equals to VUART10[3] Out2 (loopback mode)". Also, VUART18[3] is redefined as "Delta Data Set Ready (DDSR) indicator (non-loopback mode)". That is the DCD status on both sides of VUART mimics a null-modem cable. The DTR output from one side now drives both the DSR and DCD status bits of the other side. Both the host side and the slave side of VUART will be fixed in rev A2.

7. No reset FIFO when toggling Loopback mode.
   - In AST2050 / AST1100 VUART, the Host Tx and the SP Rx share the same FIFO, FIFOA. Also, the Host Rx and the SP Tx share the same another FIFO, FIFOB. If an AP at Host side toggles the loopback mode without reset Tx/Rx FIFO, it will confuse the data in Tx/Rx FIFO. FIFO reset auto-generation is needed for this issue.
   - Will be fixed in rev A2. Only the Host side of VUART will be fixed in rev A2.
   - But the SP side of VUART will not be fixed in rev A2.
8. Clock speed limitation for the bridge of APB to LPC master.
   - There is a clock speed limitation for the bridge of APB to LPC master. APB clock has to be not faster than LPC master clock (APB PCLK $<=$ 33MHz if LPC LHCLK = 33MHz). This limitation can be ignored if not enabling this feature.
   - Will be fixed in rev A2.
9. VUART and PUART reset when ARM reboots.
   - VUART20[0]/PUART20[0] (enable bit) and VUART20[1]/PUART20[1] (SerIRQ polarity) will be reset when ARM reboots in rev. A1. In this case, VUARTs will not respond to the Host and an abort may occur on the LPC bus.
   - Will be fixed in rev A2.
10. VUART Delta Modem Status fails in the loopback mode.
    - In the loopback mode (VUART10[4]=1), Delta Modem Status VUART18[3:0] should be set to 1 when VUART10[3:0] changed. But this behavior fails in rev. A1.
    - Only the Host side of VUART will be fixed in rev A2.
    - But the SP side of VUART will not be fixed in rev A2.
11. VUART/PUART SerIRQ low-pulse stretchers in the rising-edge trigger mode.
    - To assure that the Host does not miss a SerIRQ in the rising-edge trigger mode, VUART/PUART should hold any high to low transition and keep it held low until the low state has successfully been transmitted to the Host. But this behavior fails in rev. A1.
    - Both VUART and PUART will be fixed in rev A2.
12. VUART FIFO counters corruption.
    - It occurs only when FIFO is full or empty in rev. A1. A FIFO write pulse from the one side accompanying a FIFO read pulse from the other side will corrupt the FIFO counters.
    - Both the Host side and the SP side of VUART will be fixed in rev A2.

Improved Function List

1. Improve USB PHY signal immunity. This improvement need to change the external resistor connected on pin USBVRES from 6.195KΩ to 8.2KΩ.

New Feature List

1. VUART Tx full indicator.

- This is a new status bit of VUART in rev A2. It means Tx FIFO full (16 bytes) if true. A read-only status bit located at MCR[7] of the Host side indicates that the host-to-SP FIFO is full. The other read-only status bit located at VUART30[7] of the SP side indicates that the SP-to-host FIFO is full.

2. VUART THRE interrupt threshold (1/2 full).

   - This is a new function of VUART in rev A2. With enabling this function, THRE interrupt trigger level becomes under 1/2 full (8 bytes). Only the function FIFO 1/2 full on both sides of VUART is implemented. The FCR[5:4] of VUART do not exist and can not be used to adjust the threshold. So the threshold will be 1/2-full only if this function enabled. The enable bit is located at IER[7] for the Host side and at VUART34[7] for the SP side.

3. Reference clock option for UART1 and UART2. There is a new selection bit, SCU2C[12], of the reference clock for UART1 and UART2 baud rate. In rev A1 or no enabling SCU2C[12] in rev A2, baud rate = 24MHz / (16 * divisor). With SCU2C[12] enabled in rev A2, baud rate = (24MHz / 13) / (16 * divisor).

4. UARTx remapped as COMx through PUART and LPC-to-AHB bridge. PUART is redefined as COMx from the Host viewpoint in the rev A2. The Host can see and directly control AST2050 / AST1100 UART1 or UART2 like SIO COM1 (say 0x03f8∼0x3ff) on the LPC bus. Here is an example of the register settings for Host COM1 (0x03f8 and IRQ4) on AST2050 / AST1100 UART2:

   - PUART_20[1:0]=2'b11
   - PUART_24[7:4]=4'h4
   - PUART_28[7:0]=8'hf8
   - PUART_2C[7:0]=8'h03
   - PUART_34[7]=1'b1 */* 1: UART2; 0: UART1 */*
   - HICR5[8]=1'b1
   - HICR7[31:16]=16'h1e78
   - HICR8[31:16]=16'hffff

5. MUX function of UART1 pins. UART1 pins (NCTS1, NDCD1, NDSR1, NRI1, NDTR1, NRTS1, TXD1, and RXD1) are reserved for AST2050 / AST1100 UART1 only in rev A1. In rev A2, these pins can be switched to AST2050 / AST1100 UART2 with enabling the new bit, SCU2C[14]. With this function, AST2050 / AST1100 UART1 and UART2 can share the same UART1 pins and the connector. In the meantime, UART1 still monitor the UART1 chip input pins.

6. Internal link function between UART1 and UART2. This is a new mode in rev A2. With enabling SCU2C[15], AST2050 / AST1100 UART1 (NCTS1, NDSR1, NDTR1, NRTS1, TXD1, and RXD1) is connected to AST2050 / AST1100 UART2 (NRTS2, NDTR2, NDSR2, NCTS2, RXD2, and TXD2) inside the chip. Also, NDCD1 is connected to NDTR2; NDCD2 is connected to NDTR1. Both NRI1 and NRI2 are in the idle state. Besides, UART1 still drives NRTS1, NDTR1, and TXD1 if SCU2C[15]=1. UART2 still drives TXD2 if SCU2C[15]=1 and drives NRTS2 and NDTR2 if SCU2C[15]=1 and SCU74[24]=1.

7. Vector interrupt controller output, nIRQ, connected to system serial IRQ. This is a new mode in rev A2. With enabling LHCR0[12] (active low), AST2050 / AST1100 VIC output, nIRQ, can be fed to the Host through the KCS channel #2 IRQX (defined in HICR5[19:16] and HICR5[13:12]). It is not just to benefit a test program running on the host CPU; it also provides an option for the host CPU to use AST2050 / AST1100 modules (ex.: GPIO) when ARM disabled.

AST2050 / AST1100 Migration List from A2 to A3

- Fix internal layout guard-ring short circuit issue.

Contents

List of Figures

Part I
Functional Specification

1 General Information

1.1 Introduction

This manual provides the related technical information for both AST2050 and AST1100 Integrated Remote Management Processor. Its intended for product planners, system designers, and software developers who are going to adopt or have adopted this device to support graphics acceleration & display, baseboard management, virtual storage functions, and/or KVM-over-IP functions for developing highly manageable server platforms.

1.2 Chip Architecture

AST2050 and AST1100 are the 2nd generation of Integrated Remote Management Processor introduced by ASPEED Technology Inc. They are high performance and highly integrated SOC devices to support various management functions required for highly manageable server platforms. Figure-1 clearly illustrates the primary chip architecture of the device. The detailed functions of the individual internal blocks will be descried in chapter 2.

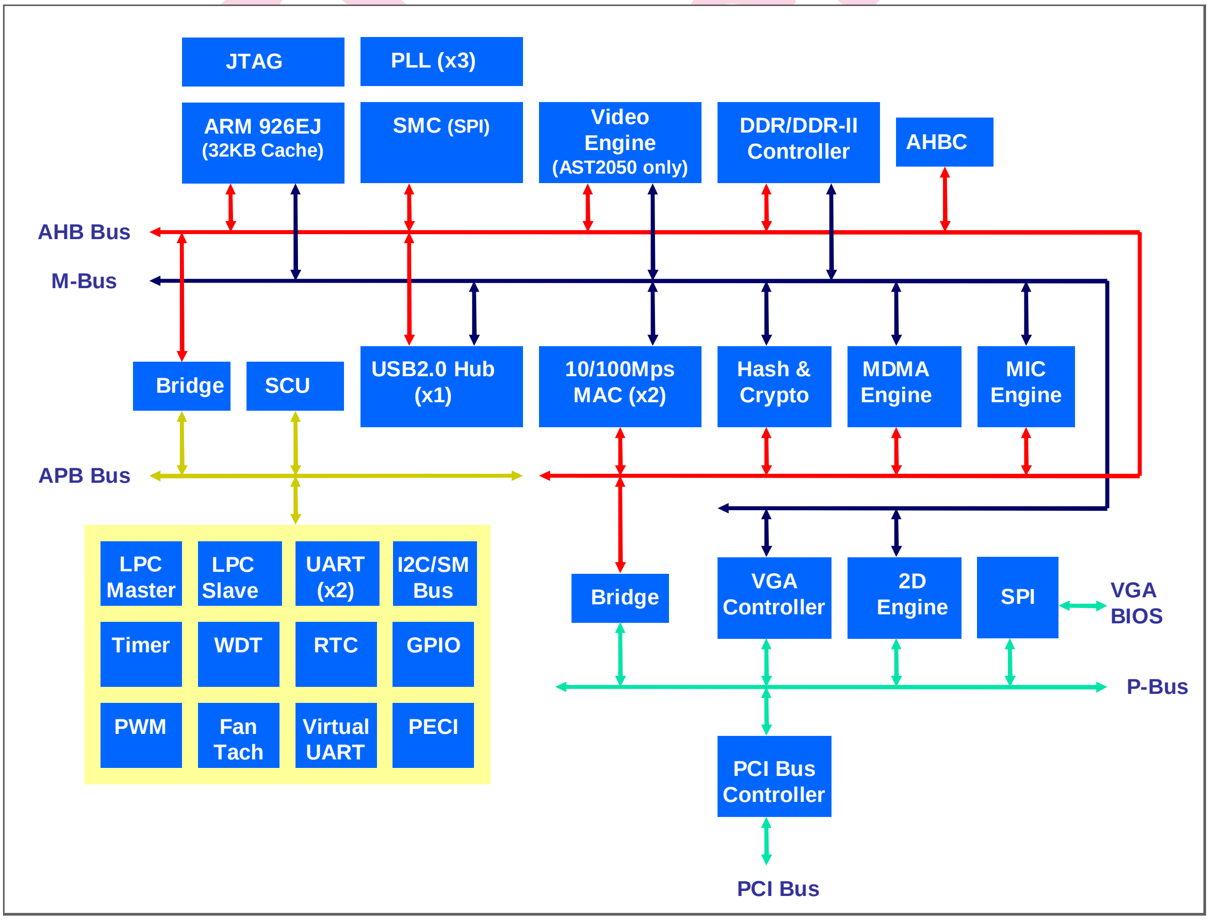


Figure 1: AST2050/AST1100 Chip Architecture

1.3 Summary of Feature Set

1.3.1 Process & Package

- TSMC 0.13um Logic Salicide 1.2V/3.3V 1P6M CMOS process
- 19mmx19mm 355-pin TFBGA package
- 0.8 mm ball pitch
- 4-layer substrate with RoHS compliance
- Lead-free soldering balls

1.3.2 Design for Testability

- Full scan chains for internal logic tests by Automatic Test Pattern Generation (ATPG)
- Built-In Self-Test (BIST) for internal SRAM macro tests
- Support XOR-tree for pin scan tests

1.3.3 PCI Bus Controller

- Support 32-bit 33 MHz PCI bus salve interface with PCI 2.3 specification compliant
- Support big-endian & little-endian which can be enabled by register settings
- Support AD[31:0] bus reverse option for PCB layout optimization

1.3.4 VGA Display Controller

- Fully IBM VGA compliant
- Maximum Display resolution: 1600x1200@60Hz
- Integrate one dedicted PLL for video clock generation which can be directly turned off by ARM CPU for power saving
- Support VESA DDC
- Support 64x64 hardware overlay cursor with mono and color formats
- RGB analog output
  - Integrate 165MHz triple DACs compliant with VESA monitor timing specification
  - Integrate 1.2V reference voltage generator
  - Need external analog comparators for monitor sense
  - Support DAC power down function directly controlled by ARM CPU or host CPU
- Digital video output options
  - 165MHz 18-bit single-edge DVO (3.3V digital signals, RGB666)
  - 165MHz 12-bit dual-edge DVO (3.3V digital signals, RGB888)

1.3.5 DDR/DDR2 SDRAM Controller

- Support external 16-bit DDR/DDR2 SDRAM data bus width
- Maximum memory clock frequency
  - DDR : 200/166MHz (DDR400/DDR333)
  - DDR2 : 200MHz (DDR2 400)
- Embedded Delay-Lock-Loop (DLL) for precise DRAM strobe timing optimization
- Embedded one dedicated PLL for memory cock generation
- Embedded programmable on-die terminators (ODT)
- Support dynamic power down control signal (CKE)
- Programmable driving strength for output buffers
- Programmable priority setting for DRAM arbiter
- Support Internal 64-bit DRAM data bus width
- Support DDR/DDR2 DRAM Types: 8MBx16, 16MBx16, 32MBx16, 64MBx16
- Support memory configurations:

Capacity	Configuration
16MB	8Mx16 @1pcs
32MB	16Mx16 @1pcs
64MB	32Mx16 @1pcs
128MB	64Mx16 @1pcs

1.3.6 Video Compression Engine (AST2050 only)

- Directly connected to AHB bus interface for register programming
- Directly access video data through M-Bus
- Maximum engine frequency: 200MHz
- Video source can be from internal VGA output or from DVO input
- Engine clock can be from CPU clock or memory clock
- Engine clock can be turned off when engine is idle
- Support two video compression formats
  - YUV420: for lower video quality but higher frame rate
  - YUV444: for higher video quality but lower frame rate
- Support high resolution video compression up to 1600x1200x16bpp@60Hz
- Target frame rate: 30 frame/sec for 1280x1024@60Hz
- Support intelligent scene change detection scheme by comparing CRC code of each video block, significantly reducing memory bandwidth requirement
- Support Quick Cursor (Patent pending by ASPEED)
  - Achieve 60 frames/sec cursor refresh rate
  - Directly transmit cursor patterns and X/Y locations to remote site

  - Directly overlay cursor pattern at remote site
- Support Quick Fetch (Patent pending by ASPEED)
  - Significantly reducing memory requirements for video compression
  - 16 bits of DRAM bus width is enough for 1600x1200x16bpp video compression
  - Only enabled for high resolution modes (high color and true color modes)
  - Quick Cursor must be enabled (cursor overlay will be done in client site)
  - Regular VGA display refresh can be turned off when feasible
- Support arbitrary video down scaling with horizontal & vertical video filtering option
- Integrate one RC4 encryption engine for video stream encryption
  - 1 set of loadable 256x8 SRAM for expanded key buffers
  - Key expansion is done by firmware
  - Provide enable/disable option

1.3.7 Embedded ARM926EJ-S CPU

- Embedded ARM926EJ 32-bit RISC CPU
- Maximum running frequency: 200MHz
- Instruction cache size: 16KB
- Data cache size: 16KB
- Support Memory Management Unit(MMU)
- Interface: Dual AHB buses for both instruction and data access
- Integrate a dedicated PLL for CPU clock generation
- Power-on running frequency can be set by external trapping resistors
- Default power frequency options: 100/133/166/200MHz
- AHB bus clock speed divider options: 1/1, 1/2, 1/3, 1/4 (by trapping)
- Integrate standard AMBA APB bus with embedded AHB to APB bridge
- JTAG interface with CPU reset control for code debugging

1.3.8 System Control Unit (SCU)

- Directly connected to internal APB bus
- Centralize clock and reset control registers of all modules
- Support programmable CPU clock divider (1/1 ∼ 1/16) for power saving
- Integrate all PLL control registers
- Integrate all power saving control registers
- Integrate 2 sets of ring oscillators for process window monitoring
- Integrate hardware trapping control registers
- Integrate PCI ID setting registers
- Integrate 16 bytes of scratch registers for Host CPU to ARM CPU message passing
- Integrate 8 bytes of scratch registers for ARM CPU to Host CPU message passing

1.3.9 AHB Controller (AHBC)

- Directly connected to internal AHB bus
- AHB master and slave controller
- AHB bus multiplexer
- AHB slave address decoder
- AHB master controller with two-level arbitration (round-robin arbitration for each arbitration level)
- AHB memory address remapping control with register-write protection

1.3.10 Vector Interrupt Controller (VIC)

- Directly connected to AHB bus interface
- Support up to 32 interrupt sources
- Support rise/fall edge-triggered and high/low level-triggered interrupt settings

1.3.11 Static Memory Controller (SMC) - SPI Flash Memory Controller

- Support code boot for ARM CPU
- Programmable flash timing parameters
- Support 2 chip select pins
- Support dual input SPI flash memory
- Maximum clock frequency: AHB Bus frequency / 2

1.3.12 USB2.0 Virtual Hub Controller

- Compliant with USB Specification Revision 2.0
- Integrate 1 set of USB2.0 Virtual Hub Controller
- Integrate 1 set of USB2.0 PHY
- Directly connected to AHB bus interface for register programming
- Directly access DMA data through M-Bus
- Support USB2.0 standard and backward compatible with USB1.1 standard
- Support one hub port and 7 downstream ports with configurable endpoint type
- Support total 21 configurable endpoints
- Support each downstream port with:
  - Control endpoint : 1 set
  - Interrupt/Bulk/Isochronous endpoint : 1-15 sets (total 21 sets in endpoint pool)
- Integrated DMA engine for direct memory bus accesses (bypass AHB bus)
- Support automatic retry of failure packets and PING flow control
- Support USB remote wake-up (Suspend/Resume)

1.3.13 64-bit 2D Graphics Accelerator

- Directly access data through M-Bus
- High performance pipelined one-cycle 64-bit 2D engine
- Optimized for RGB565 and XRGB8888 pixel formats
- 2D engine commands
  - BitBlt Rectangle Fill
  - BitBlt Pattern Fill
  - BitBlt Rectangle Copy from Source to Destination
  - Support 256 Raster Operations
  - Integrate 8x8 Pattern Registers
  - Integrate 8x8 Mask Registers
  - Support Rectangle Clip
  - Support Color Expansion
  - Support Enhanced Color Expansion
  - Support Line Drawing with Style Pattern
- Integrate 16 stages of hardware command queue for 2D command pre-fetch
- Integrate 64x16 source buffer and 64x16 destination buffer to improve 2D engine performance

1.3.14 10/100 Mbps Fast Ethernet MAC

- Integrate dual MAC compliant with IEEE802.3 specification
- Support 10/100M bps transfer rate
- Support Media Independent Interface (MII x1) or Reduced Media Independent Interface (RMII x2)
- AHB bus interface supports bus master and slave mode
- DMA engine for transmitting and receiving packets
- Integrated link list DMA engine for M-Bus access
- Support IEEE 802.1Q VLAN tag insertion and removal
- Support High Priority Transmit Queue for QoS and CoS applications
- Independent TX/RX FIFO
- Support half and full duplex
- Support flow control for full duplex and backpressure for half duplex
- Support zero-copy data transfer
- Support IP, TCP, UDP receive checksum offloads
- Support Jumbo packets (9K bytes)

1.3.15 I2C/SMBus/FML Serial Interface Controller

- Directly connected to APB bus
- Integrate 7 sets of multi-function I2C/SMBus bus controllers
  - Each controller can be programmed as a master or slave controller
  - Data bit rate can be up to 1M bps
  - Embedded 256 bytes of FIFO with dynamic FIFO length allocation for the 7 I2C controllers
  - Support 2 alert pins for 2 sets of SMBus controller (shared with 1 set I2C pins)
  - 2 out of the 7 I2C controller can be programmed as FML controllers
- Schmitt type of input data buffer and input clock buffer
- Optional anti-glitch input data filter
- Support recovery capability for SDA data line locked case
- Need external pull-up resistors

1.3.16 GPIO Controller

- Directly connected to APB bus
- Support 46 shared GPIO pins
- Programmable reset tolerance option for each GIPO pin
- Support interrupt triggered by all the 46 GPIO pins
- Each input pin is with 0ms/1us/1ms/5ms/10ms de-bouncing logic option
- 8 out of the 46 GPIO pins are with 16mA driving current, others are with 8mA.

1.3.17 UART (16550)

- Directly connected to APB bus
- Support two UART, UART1 has full flow control pins, and UART2 only has TX/RX pins
- Separate transmit & receive FIFO buffer (16x8) to reduce CPU interrupts
- Support up to 115.2K baud-rate
- Programmable baud rate generator
- Standard asynchronous communication bits – Stat/Stop/Parity
- Independent masking of transmit FIFO, receive FIFO, receiving timeout and error condition Interrupts
- False start-bit detection
- Line break generation and detection
- Fully programmable serial interface characteristics
- 5/6/7/8 data length
- Even, odd, stick and none parity generation and detection
- 1/2 stop-bit generation

1.3.18 Timer

- Directly connected to APB Bus
- Built-in 3 sets of 32-bit timer modules
- Free-running or periodic mode
- Maskable interrupts

1.3.19 Watchdog Timer (WDT)

- Directly connected to APB bus
- Watchdog function
- Built-in 32-bit programmable counter
- Generate interrupt or reset after counting down to zero (programmable)

1.3.20 Real Time Clock (RTC)

- Directly connected to APB bus
- Clock source is divided from 24MHz clock input
- 24-Hour timer mode with highest precision of tenth of a second
- Support Calendar function with correction logic for leap years
- Programmable alarm with interrupt generation
- Maskable interrupt
- No battery backup support

1.3.21 LPC Bus Interface

- Directly connected to APB bus interface
- Dual operation modes
  - Master mode: designed to update system BIOS, TPM or LPC keyboard controller (I/O, memory, firmware read write cycles)
  - Salve mode: designed for BMC functions (I/O read write cycles)
- Support Serial IRQ (reduce polling time)
- Support port 80H/81H (programmable address) snooping registers with interrupt option
- Support one Virtual UART and one Pass-through UART (16550) (SIRQ#)
- Compliant with IPMI version 2.0 KCS/BT mode

1.3.22 Hash & Crypto Engine

- Directly connected to APB bus
- Direct data access through internal memory bus
- Programmable AES/RC4 encryption or decryption mode with programmable context buffer location for fast context switching
- Support multiple message digest standards: MD5/SHA1/SHA224/SHA256, HMAC-MD5/HMAC-SHA1/HMAC-SHA224/HMAC-SHA256
- Support 4 types of engine trigger modes:
  - Encryption/decryption only
  - Message digest only
  - Encryption/decryption first, message digest second
  - Message digest first, encryption/decryption second
- Support AES crypto standard with the following modes:
  - Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), Counter (CTR)
  - Support Three Different Key Sizes : 128, 192 or 256 bits
  - Key expansion task is done by software
  - Encryption/Decryption Throughput: $>$ 100M bps

1.3.23 MDMA Engine

- Directly connected to AHB bus
- Support direct data transfer through internal memory bus
- Accelerate memory-to-memory data movement throughput by at least 4 times
- Support fast memory data fill operation (for ECC memory initialization)

1.3.24 Memory Integrity Check (MIC) Engine

- Directly connected to AHB bus
- Automatic memory integrity check by constant checksum scanning
- Directly access SDRAM memory through M-bus
- 4K bytes of memory per checksum unit
- Each checksum unit can be individually enabled or skipped
- Support Fletcher's Checksum algorithm
- Programmable scanning rate and scanning range control
- Slight extra memory bandwidth and capacity demand
- Generate an interrupt whenever detecting checksum errors

1.3.25 PWM Controller

- Support 4 PWM outputs
- Support both low-frequency and high-frequency PWM for fan speed control
- Duty cycle from 0 to 100% with 1/256 resolution
- Support low-frequency PWM pulse stretching for fan speed measurements
- Shared with GPIO pins

1.3.26 Fan Tachometer Controller

- Directly connected to APB bus
- Support up to 16 tachometer inputs
- Measurement schemes: rising edge, falling edge or both edges
- Support Interrupt trigger when over fan speed limitation setting
- Shared with DVO input pins

1.3.27 PECI Controller

- Directly connected to APB bus
- Intel PECI 2.0/1.1 compliant
- Support up to 4 CPU and 2 domains per CPU
- Need an external analog transmitter and receiver circuit
- Shared with GPIO pins

1.3.28 Software Specifications

- Manufacturer Test Program (under DOS)
- VBIOS
- Windows Driver
  - Support Windows 2000 (with WHQL)
  - Support Windows Server 2003 x86/x64 (with WHQL)
  - Support Windows Server 2008 x86/x64 (with WHQL)
  - Support Windows XP x86/x64 (without WHQL)
- Linux X Window Driver
  - Support XFree86 4.x.x
  - Support XORG 6.8.x, 6.9.x, 7.x
- FreeBSD X Window Driver
  - Support XORG 6.9
- Solaris 10 X86
  - Support XORG 6.8, 6.9, 7.x
- VGA BIOS flash utility (under DOS)

1.4 Feature Comparisons (AST2100/AST2050/AST1100)

The following table shows the major feature comparisons between AST2100, AST2050 and AST1100 product specification.

Feature	AST2100	AST2050	AST1100
VGA/2D Controller	Yes	Yes	Yes
Graphics Display Controller	Yes	No	No
BMC Controller	Yes	Yes	Yes
Storage Redirection	Yes	Yes	Yes
KVM Redirection	Yes	Yes	No
Process Technology	TSMC 0.13um	TSMC 0.13um	TSMC 0.13um
Package	27mmx27mm PBGA	19mmx19mm TFBGA	19mmx19mm TFBGA
Pin Count	487 Pins	355 Pins	355 Pins
Ball Pitch	1.0 mm	0.8 mm	0.8 mm
Typical Power Consumption	< 1.5W	< 1.25W	< 1W
ARM926 Embedded CPU	275MHz (max)	200MHz (max)	200MHz (max)
SDRAM Memory Bus Width	32/16 Bits	16 Bits	16 Bits
SDRAM Memory Types	DDR2/DDR	DDR2/DDR	DDR2/DDR
Maximum Memory Capacity	256MB	128MB	128MB
Maximum Memory Clock Frequency	266MHz	200MHz	200MHz
ECC Support	Yes	No	No
Memory Integrity Check Engine	Yes	Yes	Yes
MDMA Controller	Yes	Yes	Yes
Maximum Graphics Display Resolutions	1920x1200@60Hz	1600x1200@60Hz	1600x1200@60Hz
Maximum Video Clock Frequency	200MHz	165MHz	165MHz
USB 2.0 Controller	Yes	Yes	Yes
USB 1.1 Controller	Yes	No	No
Hash & Crypto Engine	Yes	Yes	Yes
I2C/SMBus Controller	Yes (x7)	Yes (x7)	Yes (x7)
PWM Outputs	Yes (x4)	Yes (x4)	Yes (x4)
Fan Tech	Yes (x16)	Yes (x16)	Yes (x16)
PECI 2.0/1.1	Yes	Yes	Yes
GPIO	64 (max)	46 (max)	46 (max)
UART	Yes (x2) (Flow control x2)	Yes (x2) (Flow control x1)	Yes (x2) (Flow control x1)
Virtual UART	Yes	Yes	Yes
Pass-through UART	Yes	Yes	Yes
LPC Bus Controller	Yes (Salve & Master)	Yes (Slave & Master)	Yes (Slave & Master)
Watchdog Timer	Yes	Yes	Yes
Timer	Yes	Yes	Yes
Real Time Clock (RTC)	Yes	Yes	Yes
Digital Video Output	Yes (24-Bit Single Edge or 12-Bit Dual Edge)	Yes (18-Bit Single Edge or 12-Bit Dual Edge)	Yes (18-Bit Single Edge or 12-Bit Dual Edge)
Digital Video Input	Yes (24-Bit Single Edge or 12-Bit Dual Edge)	Yes (18-Bit Single Edge or 12-Bit Dual Edge)	No
Ethernet MAC Module	Yes (x2)	Yes (x2)	Yes (x2)
Ethernet MAC Throughput	10/100/1000M bps	10/100M bps	10/100M bps
Ethernet MAC Interface	MII(x2) or RMII(x2) or GMII(x1)	MII (x1) or RMII (x2)	MII (x1) or RMII (x2)
Flash Memory Controller	SPI Flash or/and NOR Flash or/and NAND Flash	SPI Flash	SPI Flash

1.5 Feature Comparisons (AST2050/AST1100/AST2000)

The following table shows the major feature comparisons between AST2050, AST1100 and AST2000 product specification.

Feature	AST2050	AST1100	AST2000
VGA/2D Controller	Yes	Yes	Yes
BMC Controller	Yes	Yes	Yes
Storage Redirection	Yes	Yes	Yes
KVM Redirection	Yes	No	Yes
Process Technology	TSMC 0.13um	TSMC 0.13um	TSMC 0.18um
Package	19mmx19mm TFBGA	19mmx19mm TFBGA	27mmx27mm PBGA
Pin Count	355 Pins	355 Pins	388 Pins
Ball Pitch	0.8 mm	0.8 mm	1.0 mm
Typical Power Consumption	< 1.25W	< 1W	<2.0W
ARM926 Embedded CPU	200MHz (max)	200MHz (max)	200MHz (max)
SDRAM Memory Bus Width	16 Bits	16 Bits	32/16 Bits
SDRAM Memory Types	DDR2/DDR	DDR2/DDR	DDR
Maximum Memory Capacity	128MB	128MB	256MB
Maximum Memory Clock Frequency	200MHz	200MHz	166MHz
ECC Support	No	No	No
Memory Integrity Check Engine	Yes	Yes	No
MDMA Controller	Yes	Yes	No
Maximum Graphics Display Resolutions	1600x1200@60Hz	1600x1200@60Hz	1600x1200@60Hz
Maximum Video Clock Frequency	165MHz	165MHz	165MHz
Video Compression Format	YUV420 & YUV444	No	YUV420
Video Compression Resolution (max)	1600x1200@60Hz	No	1600x1200@60Hz
Video Compression Algorithm	JPEG & VQ	No	JPEG
USB 2.0 Controller	Yes	Yes	Yes (Device)
USB 1.1 Controller	No	No	Yes
Hash & Crypto Engine	Yes	Yes	Crypto Only
I2C/SMBus Controller	Yes (x7)	Yes (x7)	Yes (x7)
PWM Outputs	Yes (x4)	Yes (x4)	No
Fan Tech	Yes (x16)	Yes (x16)	No
PECI 2.0/1.1	Yes	Yes	No
GPIO	46 (max)	46 (max)	32 (max)
UART	Yes (x2) (Flow control x1)	Yes (x2) (Flow control x1)	Yes (x2) (Flow control x1)
Virtual UART	Yes	Yes	No
Pass-through UART	Yes	Yes	No
LPC Bus Controller	Yes (Slave & Master)	Yes (Slave & Master)	Yes (Slave)
Watchdog Timer	Yes	Yes	Yes
Timer	Yes	Yes	Yes
Real Time Clock (RTC)	Yes	Yes	Yes
Digital Video Output	Yes (18-Bit Single Edge or 12-Bit Dual Edge)	Yes (18-Bit Single Edge or 12-Bit Dual Edge)	Yes (24-Bit Single Edge)
Digital Video Input	Yes (18-Bit Single Edge or 12-Bit Dual Edge)	No	Yes (24-Bit Single Edge)
Ethernet MAC Module	Yes (x2)	Yes (x2)	Yes (x1)
Ethernet MAC Throughput	10/100M bps	10/100M bps	10/100M bps
Ethernet MAC Interface	MII (x1) or RMII (x2)	MII (x1) or RMII (x2)	MII (x1) or RMII (x1)
Flash Memory Controller	SPI Flash	SPI Flash	NOR Flash

1.6 Applications

Figure-2 illustrates the typical applications of the device in server applications.

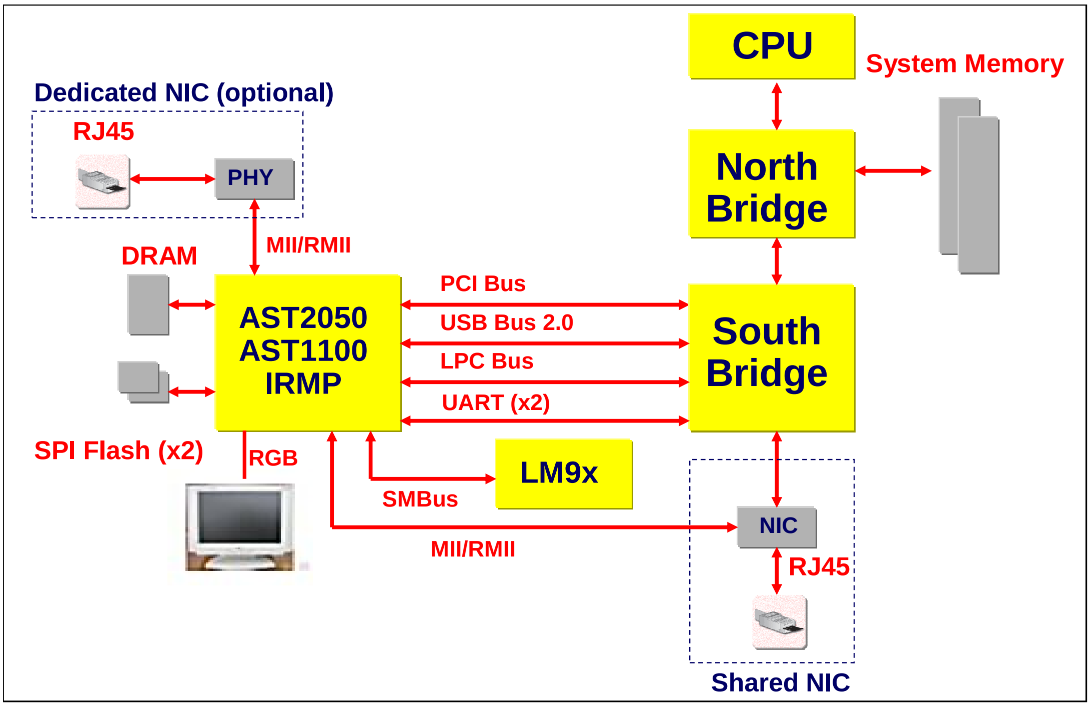


Figure 2: AST2050/AST1100 system applications

2 Functional Description

2.1 Embedded CPU

The embedded CPU core integrated in AST2050 / AST1100 is ARM926EJ with maximum running frequency 200MHz. The CPU core includes 16KB I-cache, 16KB D-Cache, Memory Management Unit (MMU), and two AHB system buses.

AST2050 / AST1100 provides a JTAG-compliant interface for code debugging in the initial development phase. The JTAG interface also supports CPU reset pin that can be directly controlled by In-Circuit-Emulator (ICE).

An integrated PLL-based clock generator generates the required CPU clock. There are several clock frequencies (200/166/133/100MHz) can be selected as the power-on CPU clock frequency by the setting of external trapping resistors. An integrated logic-based programmable frequency divider (1/1, 1/2, 1/4, 1/8, 1/16) can be used to slow down CPU frequency for power saving. The default value of the frequency divider is 1/1.

The embedded CPU can be reset under several conditions:

- Power-On reset (generated from external application circuit)
- Watchdog Timer reset (generated from Watchdog Timer)
- GPIO pin reset (optional)

The reset signal from PCI/LPC bus will not reset the embedded CPU at all.

AST2050 / AST1100 adopts dual AHB bus architecture that can significantly improve CPU performance. Simply put, one of the AHB bus is primarily designed to serve all the memory access from I-cache miss. The other AHB bus is primarily designed to serve all the memory access from D-cache miss. Peripheral read & write cycles will also go through one of the AHB buses. With the help of the two AHB bus bridges, CPU can directly access data through SDRAM memory controller, which can support much higher bandwidth than simply going through AHB bus and serve many requests simultaneously. Additionally, AHB bus congestions can be significantly reduced from this architecture.

2.2 SDRAM Memory Controller

AST2050 / AST1100 integrates a very flexible SDRAM Controller to support either DDR or DDR2 SDRAM memory chips. The I/O buffers of SDRAM memory interface are combo-type buffers that can support both SSTL-1 and SSTL-2 signaling interface depending on the related register setting. The SDRAM memory data bus width can only be 16 bits, programmed by internal registers. Except for targeting on very high performance applications, 16-bit data bus width should be able to meet most of the applications.

An integrated Delay-Lock-Loop (DLL) is designed to support high precision timing tuning for DRAM interface signals. With the help of the DLL, the output timing can be highly stable even over a wide range of semiconductor process, operating voltage or working temperature variations.

For DDR SDRAM memory configuration, AST2050 / AST1100 can support DDR400 and DDR333 SDRAM memory chips with 2.5V or 1.8V signaling voltage. For DDR2 SDRAM memory configuration, AST2050 / AST1100 can support DDR2 400 with 1.8V signaling voltage. All the related I/O buffers can be programmed with two driving strengths. On-die terminator option is available too.

2.3 VGA Controller

The integrated VGA controller is fully compliant with the well-known legacy VGA standard. One dedicated programmable PLL is designed to generate the required video clock for different graphics display resolution modes, ranging from 25MHz $\sim$ 165MHz. For power saving or any other purpose, ARM CPU can directly turn off this PLL. The maximum graphics display resolution mode that AST2050 / AST1100 can support is 1600x1200@60Hz. The corresponding video clock frequency is 165MHz.

An analog block with embedded triple Digital-to-Analog Converters (DAC) is integrated by AST2050 / AST1100 . One set of 1.2V reference voltage generator is equipped with the analog block. The maximum swing of the analog video output is set by 700mV (Vpp). For sensing the attachment of external monitors, an external analog comparator is required. The output of the comparator has to be connected to the input pin dedicated for monitor sense (GPIOD5). If the analog video output is split for two video outputs, each output can be equipped with one analog comparator, and the two output can be wired OR before connecter to the above dedicated pin.

AST2050 / AST1100 also supports hardware cursor overlay function with 64x64 cursor window size. The hardware cursor can be either 2 bit-per-pixel formats or RGB555 color format. VESA compliant DDC function is also available.

AST2050 / AST1100 also supports Digital Video Output (DVO) signals. The video output signals can be programmed as 18-bit single-edge or 12-bit dual-edge signaling mode. DVO interface is designed to co-work with an external TMDS transmitter for supporting DVI output. When selecting 12-bit dual-edge mode, the left 12 DVO pins can be configured as GPIO pins. When DVO output function is not necessary, all the 18 DVO pins can be configured as GPIO pins. Due to the limitation from finite pin resource, DVO pins are shared with Digital Video Input pins designed for capturing video from external RGB data stream. In a general case, this limitation should not be an issue.

2.4 2D Graphics Accelerator

AST2050 / AST1100 integrates a powerful-pipelined 2D Graphics Accelerator with maximum throughout at 64 bits data output per clock. The instruction set supported by 2D Graphics Accelerator is fully complaint with Microsoft Windows operating systems or Linux operating systems. Its clock source can be either from ARM CPU PLL or SDRAM PLL. In a general condition, selecting clock source from SDRAM PLL is recommended. The maximum running frequency is 200MHz. When 2D Graphics Accelerator stays in idle state, its clock can be totally stopped or automatically divided into 1/16 for power saving. Either one can be enabled by a simple register setting. When receiving new commands, 2D Graphics Accelerator will automatically resume its operation frequency to its full speed.

AST2050 / AST1100 supports the following display drivers according to the current product plan.

- **Windows XP/2K/03 32 Bits**
- **Windows XP/03 64 Bits (Win Server 03 with AMD64/EM64T)**
- **XFree86 32 Bits (RHEL v3/v4 with X86)**
- **XFree86 64 Bits (RHEL v3/v4 with AMD64/EM64T)**
- **SUSE 9 (XFree86 32/64 Bits)**
- **NetWare 6.5 (VESA mode)**

2.5 Video Compression Engine

AST2050 / AST1100 integrates a very powerful video compression engine providing a wide range of tradeoffs between the video quality level and the compression ratio. The tradeoff is highly depends on the available network bandwidth and the expected visual quality level. The compression algorithm is a proprietary one, specially developed to enhance not only video compression but also graphics compression. Actually, the later one is much difficult to handle and consumes a great deal of network bandwidth.

Basically, the compression algorithm adopted by AST2050 / AST1100 is a block-based compression scheme. The block size can be either 8 pixels by 8 pixels or 16 pixels by 16 pixels, depending on the selected video data format. Video Compression Engine will automatically monitor any scene change block by block. Only the blocks with scene change will be compressed again. This scheme will significantly reduce network bandwidth requirement.

AST2050 / AST1100 supports two video data formats for compression. One is YUV420 for lower network loading and fair video quality level, the other is YUV444 for higher network loading but much higher video quality level. This feature is very important especially for improving the readability of colorful texts with a small font size. When Video Compression Engine captures RGB video data stream from the embedded Graphics Display Controller or from an external digital video input, AST2050 / AST1100 will automatically compress RGB data stream into YUV420 or YUV444 data stream, depending on the corresponding register setting.

In order to facilitate the required video compression activities, a portion of SDRAM memory will be allocated as a dedicated buffer for video compression. The size of the buffer depends on the maximum graphics resolution to be supported. AST2050 / AST1100 supports up to 1920x1200x16bpp graphics resolution. AST2050 / AST1100 adopts a special video capture scheme (Quick Fetch) that can utilize the remaining memory bandwidth for video compression, therefore, even under 16-bit SDRAM memory configuration, AST2050 / AST1100 is still able to support both graphics display and video compression at the same time under such a high graphics resolution. Certainly, it will somewhat result in a lower frame rate.

2.6 10/100 Ethernet MAC

AST2050 / AST1100 integrates dual 10/100 Ethernet MAC modules, compliant with IEEE802.3 specification, in one chip. The major functions of the dual MAC modules can support include keyboard re-direction, video re-direction, mouse redirection, storage re-direction, and IPMI related communications. Beside, fail-over feature can be easily supported by taking advantage of this new feature as well. Two external PHY chips are required when both MAC modules are enabled.

The dual MAC support several types of interfaces to external PHY chips. They include 1 set of MII interfaces, or 2 sets of RMII interfaces.

2.7 USB 2.0 Virtual Hub Controller

AST2050 / AST1100 integrates one virtual hub controller complaint with USB specification revision 2.0. By way of adopting hub architecture, software porting efforts and compatibility issue can be significantly reduced. USB 2.0 PHY is also integrated in this new silicon. There are 7 downstream ports with configurable endpoints. AST2050 / AST1100 integrates an endpoint pool with totally 21 sets of endpoints (each of which can be programmed as Interrupt, Bulk or Isochronous endpoint) to be allocated by the 7 downstream ports individually. Each downstream port is always equipped with one control endpoint, and can additionally allocate 1 to 15 sets of endpoints from the endpoint pool.

One DMA engine is equipped with this hub controller for directly accessing SDRAM memory without suffering the potential AHB bus congestion. USB remote wake is also in the supporting list. The clock source is from an external 24MHz oscillator.

Since USB 2.0 Controller can support hub function with up to 7 downstream ports, it shall be able to easily emulate USB keyboard and USB mouse functions without utilizing the integrated USB1.1 Controller. This approach can save one USB port for the on-board chip set.

2.8 Static Memory Controller (SMC)

Static Memory Controller supports SPI flash memory with CPU boot-code fetching capability. There are 2 chip select pins available to be used to expand flash memory capacity. SPI NOR flash memory configuration is good for small foot print applications with smaller code size (no more than 4MB). Due to the limited memory bandwidth provided by SPI interface, the code boot time should be longer than other flash memory type.

2.9 Vector Interrupt Controller

ARM CPU is equipped with a Vector Interrupt Controller with maximum 32 input sources. Each interrupt source can be programmed to support rising/falling-edge trigger mode or high/low-level trigger mode. ARM CPU supports two priority levels of interrupt mode:

- Fast Interrupt Request (FIQ): for fast and low latency time interrupt handling
- Interrupt request(IRQ): for more general interrupts

Each interrupt source can be programmed to generate FIRQ or IRQ. It's highly recommend that only one of the interrupt sources is programmed to generate FIRQ, and all the other interrupt sources should be programmed to generate IRQ interrupt . Furthermore, the priority level of each interrupt source totally depends on the checking sequence of interrupt software handler. This kind of priority setting will let the priority level selection be more flexible but at the cost of performance penalty in some extent. Table 36 lists the arrangement of each interrupt source for Interrupt Controller.

2.10 UART Controller

AST2050 / AST1100 integrates two sets of UART Controllers, which is fully complaint with industry standard 16550, to provide serial communication capabilities. Each UART Controller is built-in 16x8 transmit FIFO, and 16x8 receive FIFO, both of which can be programmed to be enabled or disabled. One set is with full flow control signals; the other one is with TX/RX only. The clock source of both UART Controllers is from 24MHz or 24MHz/13 clock source. In order to make it clear, the mentioned UART Controller can be categorized as physical UART Controllers which are different from the two Virtual UART Controller integrated in LPC Bus Controllers.

2.11 Virtual UART Controller

AST2050 / AST1100 integrates a Virtual UART modules providing virtual serial communication capabilities between host CPU and ARM CPU. The virtual UART is equipped two sets of registers compatible with the industry defector standard - 16550 UART. One set is for host CPU; the other set is for ARM CPU. Host CPU and ARM CPU can communicate with each other like there is a physical UART link between them, but the related data transfer actually is just through pure register read/write transfers inside the chip. The base address for host CPU to access UART registers through LPC bus can be programmed by ARM CPU by the extended related registers.

AST2050 / AST1100 also integrates a Pass-through mode of UART1 or UART2. It creats a control path from the LPC bus, through AHB/APB, to UART1 or UART2. Host can directly access UART1 or

UART2 by LPC I/O cycles without any firmware help. It could be used to replace a COM port of Super I/O on host side. The base address for host CPU to access UART1 or UART2 registers through LPC bus can be programmed by ARM CPU by the extended related registers.

2.12 Timer

There are 3 sets of 32-bit decrement timers integrated in AST2050 / AST1100 . Each counter is equipped with two sets of match registers. Whenever any one of the match registers is equal to the current counter value, an interrupt will be triggered. These timers also can be programmed to generate an interrupt or not when a counter overflow occurs. All the three counter values can be read back at any time.

The clock source of these timers is from either APB bus clock source or 1MHz clock source which is divided from the embedded never-stopped 24MHz oscillator. Therefore, adopting 1MHz clock source is highly recommended.

2.13 Watchdog Timer

AST2050 / AST1100 integrates one set of 32-bit programmable Watchdog Timer to prevent system deadlock. In general, Watchdog Timer, when being enabled, must be repeatedly re-started by firmware code before time-out, otherwise Watchdog Timer will, depending on the related register settings, generate interrupt signal to interrupt ARM CPU or generate reset signal to reset system.

The clock source of Watchdog Timer can be from either APB bus clock source or 1MHz clock source. Adopting 1MHz clock source is highly recommended.

2.14 Hash and Crypto Engine (HACE)

AST2050 / AST1100 provides one powerful Hash & Crypto Engine to accelerate encryption, decryption and message digest functions. Crypto Engine, integrating a DMA engine, can directly access SDRAM memory without going through AHB bus.

AST2050 / AST1100 supports both AES and RC4 encryption and decryption standards. For highly sensitive data, AES is highly recommended; otherwise RC4 shall be a good choice. The computing power required for RC4 is much less than the one required for AES. Therefore, firmware design shall take the available computing power of client CPU into account in selecting encryption and decryption standard.

AST2050 / AST1100 supports versatile message digest standards including MD5, SHA1, SHA224, SHA256, HMAC-MD5, HMAC-SHA1, HMAC-SHA224 and HMAC-SHA256. Furthermore, AST2050 / AST1100 supports several cascaded trigger commands to reduce SDRAM memory bandwidth loading. Based on these cascaded trigger commands, data encryption/decryption can be directly followed by message digest without writing data into and read-back from SDRAM memory, and vice versa.

2.15 I2C/SMBus Serial Interface Controller

AST2050 / AST1100 integrates 7 sets of multifunction I2C/SMBus controllers, each of which can be programmed as a bus master controller or bus slave controller. There is a 256-byte FIFO pool that can be dynamically allocated for different I2C/SMBus controller. The maximum data rate can be up to 1Mbps. Two out of seven controllers are equipped with alert pins, which can be replace by GPIO pins when alter pins are not necessary.

In order to improve noise immunity, each input buffer of I2C/SMBus controller is designed to be Schmitt input buffer. Additionally, an optional de-bouncing sequential logic is available for each input buffer. External pull-up resistors are always required.

2.16 GPIO Controller

GPIO Controller integrated in AST2050 / AST1100 can be programmed to control the direction and output state (High, Low or Tri-State) of each pin individually, and read back the status of each input buffer at any time. Due to some of the GPIO pins might be programmed to be shared by some other function pins, GPIO Controller can support at most 46 GPIO pins.

In order to improve noise immunity, some of input buffers are designed to be Schmitt input type. Furthermore, each input buffer can be programmed to generate CPU interrupt, and the interrupt sensitivity mode of each GPIO pin can be individually programmed to be rising-edge, falling-edge, dual-edge, high-level, or low-level. Each GPIO pin can also be programmed to be able to ignore the reset signal from Watchdog Timer, i.e. being reset tolerance, very useful for some applications. At this reset tolerant mode, the default GPIO register values can only being reset by SOC power-on reset signal, but the interrupt related registers is still out of this exception.

2.17 PWM & Fan Tachometer Controller

AST2050 / AST1100 can support up to 4 sets of PWM outputs. Depending on different requirements from different fan model, PWM controller can be programmed to work at high-frequency or low-frequency fan speed control mode individually. The duty cycle of each PWM output can be from 0 to 100% with 1/256 incremental resolution. In order to improve the accuracy of fan speed measurement in low-frequency mode, PWM Controller can automatically stretch PWM output signal when fan speed measurement is under going. This scheme will significantly improve the accuracy of fan speed measurement.

The clock source of PWM controller is divided from external 24MHz input. When PWM outputs are not necessary, those PWM pins can be re-configured as GPIO pins.

2.18 PECI Controller

AST2050 / AST1100 integrates a PECI 2.0/1.1 compliant host controller, addressing up to 4 CPU and supporting 2 domains per CPU. In order to simplify silicon design, an external analog comparator circuit is required. The clock source of PECI Controller is from external 24MHz input. When PECI is not necessary, this pin can be re-configured as GPIO pin.

2.19 Real-Time Clock

AST2050 / AST1100 integrates Real-time Clock (RTC) which is a 24-hour increment timer with clock precision up to a second. It also supports calendar function with correction logic for leap years.

Moreover, RTC provides second, minute, hour, day, and clock alarm functions. For example, when turned on second alarm function, RTC will auto trigger an interrupt for each second. The function is useful for implementing a clock function.

The clock source of RTC is 1MHz clock divided from the embedded 24MHz oscillator. It's not necessary to include an external 32 KHz clock oscillator to enable RTC.

In order to simplify circuit design complexity, RTC isn't equipped with any power-cut layout. Powering by battery is not allowed for AST2050 / AST1100 . Turning-off AST2050 / AST1100 standby power will result in the loss of RTC timer data. If powering by battery is necessary, an external I2C/SMBus-based RTC chip is highly recommended. The extra BOM cost should be under US$ 0.5.

2.20 LPC Interface Controller

LPC Interface Controller is designed to support BMC related applications. The device can be programmed to work at two operation modes that include:

- Master Mode (for updating LPC-based system BIOS flash memory or access an external LPC-based TPM device)
- Slave Mode (for BMC and extension functions)

Due to LPC bus protocol doesn't support multiple-master mode, AST2050 / AST1100 can work at Master Mode only when host CPU is fully shutdown otherwise bus conflict will happen and result in system malfunction. Firmware code should always take the responsibility to make sure that master cycles only happen when host CPU is already shutdown. Additionally, external analog switch device is required to electrically isolate LPC bus signaling from the attached south bridge chip. AST2050 /

AST1100 supports I/O, memory and firmware read/write cycles for LPC bus interface. SIRQ is also in the supporting list.

LPC Interface Controller is equipped with one BMC controller supporting 3 sets of KCS channels, or 2 sets of KCS channels and one set of BT channel. The register-programming model is fully IPMI 2.0/1.1 compliant. The BMC register set of AST2050 / AST1100 is compatible with the one of AST2000.

Additionally, AST2050 / AST1100 also supports Port 80H and Port 81H write cycles monitoring. Any I/O cycle with I/O address 80H or 81H will be logged in the corresponding registers. The corresponding interrupt generation is available when properly enabled. Actually, AST2050 / AST1100 provides 2 programmable addresses for I/O write cycle monitoring. Firmware code can program any two specific addresses for monitoring. 80H and 81H are the default values of the two programmable address registers.

2.21 Memory Integrity Check (MIC) Engine

Memory Integrity Check (MIC) engine provides an alternative for developing highly reliable Out-Of-Band (OOB) management processor. Unlike the solution based on the ECC option requiring extra memory capacity and bandwidth, MIC supports a low-cost alternative that can in some extent maintain the integrity of memory content but only at the cost of slight extra memory bandwidth and capacity demand.

There is no need to add any extra SDRAM chips to support this function. Only a portion of SDRAM memory is allocated as a working memory for MIC Engine. The allocated working memory records the corresponding checksum value (32 bits) and control bits (2 bits) for each checksum unit which is 4K bytes of memory in AST2050 / AST1100 design. Fletcher's Checksum algorithm is adopted in this design.

MIC Engine can directly access SDRAM memory for integrity check but with the lowest request priority to SDRAM Memory Controller. This kind of arrangement can make sure that the impact on system performance is definitely insignificant. Checksum scanning rate and scanning range are programmable. Whenever a checksum error is detected, an MIC interrupt will be generated. The corresponding interrupt handler must take the whole responsibility to recover memory integrity.

3 Pin & I/O Related Specification

3.1 Pin Description

Abbreviation Definition:

Symbol	:	Description
#	:	Denotes active low signal
I	:	Input buffer
On	:	Output buffer with n mA driving capability
I/On	:	Input/Output bidirectional buffer with n mA output driving capability
IU	:	Input buffer with internal pull high resistor
ID	:	Input buffer with internal pull low resistor
IS	:	Schmitt type input buffer
IS/O	:	Schmitt type input buffer and output buffer
P	:	Power/Ground pin
A	:	Analog pin
VREF	:	Reference voltage
PCI33	:	Standard PCI 33MHz 3.3V protocol I/O buffer with 5V tolerance input buffer.
CMOS	:	3.3V CMOS protocol I/O buffer with 5V tolerance input buffer.
SSTL	:	SSTL-18/SSTL-2 compliant SDRAM buffer type

Note : IU with internal pull high is only used for input buffer used, it can not be used to drive external loads. The system design must use Standby power domain on all paths connected on these pins to prevent current leakage from the internal pull-up resistor.

PCI BUS 46 pins

Ball	Signal	I/O	Type	Description
C22	AD0	I/O	PCI33	**PCI AD bus** 33MHz 32-bit address and data bus. The AD bus can be reversed to AD[0:31] when adding a 3.3KΩ resistor pull to 3.3V on pin ROMA21.
C21	AD1			
C20	AD2			
C19	AD3			
D22	AD4			
D21	AD5			
D20	AD6			
D19	AD7			
E21	AD8			
E20	AD9			
E19	AD10			
F22	AD11			
F21	AD12			
F20	AD13			
F19	AD14			
G22	AD15			
J20	AD16			
J19	AD17			
K22	AD18			

to next page

from previous page

K21 K20 K19 L22 L21 M22 M21 M20 M19 N22 N21 N20 N19	AD19 AD20 AD21 AD22 AD23 AD24 AD25 AD26 AD27 AD28 AD29 AD30 AD31			
E22 G21 J21 L19	CBE0# CBE1# CBE2# CBE3#	I/O	PCI33	**PCI C/BE[3:0]#** PCI bus command and byte enables bus. The CBE bus can be reversed to CBE[0:3] when adding a 3.3KΩ resistor pull to 3.3V on pin ROMA21.
J22	FRAME#	I/O	PCI33	**PCI FRAME#** Frame is driven by the PCI master to indicate the beginning and duration of an access.
H20	TRDY#	I/O	PCI33	**PCI TRDY#** Target Ready indicates the target device to complete the current data phase of the transaction.
H19	IRDY#	I/O	PCI33	**PCI IRDY#** Initiator Ready indicates the bus master to complete the current data phase of the transaction.
H22	STOP#	I/O	PCI33	**PCI STOP#** Stop indicates the current target is requesting the master to stop the current transaction.
H21	DEVSEL#	I/O	PCI33	**PCI DEVSEL#** Device Select indicates the driving device has decoded its address as the target of the current access.
L20	IDSEL	I	PCI33	**PCI ID select** Initialization Device Select is used as a chip select during configuration read and write transactions.
G20	PAR	I/O	PCI33	**PCI parity check** Parity is even parity across AD[31:0] and C/BE[3:0]#. Parity generation is required by all PCI agents.
P22	PCICLK	I/O	PCI33	**PCI clock** 33MHz PCI bus clock input for master or slave mode. Or PCI bus clock output for master mode.
P21	BRST#	IS/O	PCI33	**PCI/LPC bus reset** This reset signal reset PCI/LPC bus controller, 2D engine, VGA device of the chip only. It can be output mode when function as PCI or LPC master mode.
B11	INTA#/ GPIOB0	O16 IS/O16	CMOS	**PCI INTA# output (default)** **GPIO group B bit 0**

LPC BUS 8 pins

Ball	Signal	I/O	Type	Description
B17 A17 D16 C16	LAD0 LAD1 LAD2 LAD3	I/O	PCI33	**LPC AD bus** 33MHz 4-bit address and data bus.
A16	LCLK	I/O	PCI33	**LPC clock input/output** 33MHz LPC bus clock input for master or slave mode. Or LPC bus clock output for master mode.
B16	LFRAME#	I/O	PCI33	**LPC FRAME#** LPC Frame is driven by the LPC master to indicate the beginning and duration of an access.
D15	LPCPD#	IS/O	PCI33	**LPC power down** This LPC power down signal indicates LPC bus to enter power down mode.
C15	LPCSIRQ	I/O	PCI33	**LPC serial IRQ**

DDR/DDR2 DRAM BUS 49 pins

Ball	Signal	I/O	Type	Description
AA12 Y12 W12 AB11 AA11 Y11 W11 Y10 Y15 AA15 AA14 Y14 W14 AB13 AA13 Y13	DQ0 DQ1 DQ2 DQ3 DQ4 DQ5 DQ6 DQ7 DQ8 DQ9 DQ10 DQ11 DQ12 DQ13 DQ14 DQ15	I/O	SSTL	**DRAM data bus** 16-bit DDR/DDR2 double data rate data bus.
W10 W15	DM0 DM1	O	SSTL	**DRAM byte mask bus** When activated during writes, the corresponding data groups are masked for each data byte lane. Same timing as DQ.
AB17 W18 Y18 AA18 W19 Y19 W20 Y20 AA20 AB20 W21 Y21 AA21	MA0 MA1 MA2 MA3 MA4 MA5 MA6 MA7 MA8 MA9 MA10 MA11 MA12	O	SSTL	**DRAM address bus** These signals are used to provide the row and column address to the DRAM.

to next page

from previous page

Y17 AA17 Y16	BA0 BA1 BA2	O	SSTL	**DRAM bank address** These signals are used to provide the bank address to the DRAM. BA2 can be used as MA13 at 4 banks mode.
W16	CS#	O	SSTL	**DRAM chip select pin**
AA16	RAS#	O	SSTL	**DRAM row address select pin** Used to define the DRAM row commands.
AB16	CAS#	O	SSTL	**DRAM column address select pin** Used to define the DRAM column commands.
W17	WE#	O	SSTL	**DRAM write enable pin** Used to define the DRAM write commands.
AB18	CKE	O	SSTL	**DRAM clock enable control pin** Used to control DRAM power down mode.
AA19	CK	O	SSTL	**DRAM clock pin** DRAM differential clock. The crossing of the positive edge of CK and the negative edge of its complement CK# are used to sample the command and control signals on the DRAM.
AB19	CK#	O	SSTL	**DRAM clock pin inversed** DRAM complement differential clock.
AA10 AB14	DQS0 DQS1	I/O	SSTL	**DRAM data bidirectional strobe pins** The crossing of the rising and falling edges of DQS[1:0] and DQS[1:0]# are used for capturing data during read and write transactions.
AB10 AB15	DQS0# DQS1#	I/O	SSTL	**DRAM data bidirectional strobe pins inversed**
AB21	ODT	O	SSTL	**DDR2 on-die termination enable control** Used to turn ON/OFF DDR2 on-die termination resistance. Leave it open when using DDR type SDRAM.
T18 AB12	VREFSSTL	-	VREF	**DRAM reference voltage** DRAM reference voltage input pin. The voltage is 0.5*MVDD.
W13	VSSRSSTL	-	P	**DRAM reference ground** DRAM reference voltage ground pin. Connected to GND.

Display Interface 24 pins

Ball	Signal	I/O	Type	Description
R3	VP0/ GPIOE0/ TACH0	ID/O8 ID/O8 ID	CMOS	**Video port bit 0** **GPIO group E bit 0 (default)** **Tachometer input pin 0**
R2	VP1/ GPIOE1/ TACH1	ID/O8 ID/O8 ID	CMOS	**Video port bit 1** **GPIO group E bit 1 (default)** **Tachometer input pin 1**
R1	VP2/ GPIOE2/ TACH2	ID/O8 ID/O8 ID	CMOS	**Video port bit 2** **GPIO group E bit 2 (default)** **Tachometer input pin 2**
T4	VP3/ GPIOE3/ TACH3	ID/O8 ID/O8 ID	CMOS	**Video port bit 3** **GPIO group E bit 3 (default)** **Tachometer input pin 3**

to next page

from previous page

T3	VP4/ GPIOE4/ TACH4	ID/O8 ID/O8 ID	CMOS	**Video port bit 4** **GPIO group E bit 4 (default)** **Tachometer input pin 4**
T2	VP5/ GPIOE5/ TACH5	ID/O8 ID/O8 ID	CMOS	**Video port bit 5** **GPIO group E bit 5 (default)** **Tachometer input pin 5**
U4	VP6/ GPIOE6/ TACH6	ID/O8 ID/O8 ID	CMOS	**Video port bit 6** **GPIO group E bit 6 (default)** **Tachometer input pin 6**
U3	VP7/ GPIOE7/ TACH7	ID/O8 ID/O8 ID	CMOS	**Video port bit 7** **GPIO group E bit 7 (default)** **Tachometer input pin 7**
V4	VP8/ GPIOF0/ TACH8	ID/O8 ID/O8 ID	CMOS	**Video port bit 8** **GPIO group F bit 0 (default)** **Tachometer input pin 8**
V3	VP9/ GPIOF1/ TACH9	ID/O8 ID/O8 ID	CMOS	**Video port bit 9** **GPIO group F bit 1 (default)** **Tachometer input pin 9**
V2	VP10/ GPIOF2/ TACH10	ID/O8 ID/O8 ID	CMOS	**Video port bit 10** **GPIO group F bit 2 (default)** **Tachometer input pin 10**
V1	VP11/ GPIOF3/ TACH11	ID/O8 ID/O8 ID	CMOS	**Video port bit 11** **GPIO group F bit 3 (default)** **Tachometer input pin 11**
W4	VP12/ GPIOF4/ TACH12	ID/O8 ID/O8 ID	CMOS	**Video port bit 12** **GPIO group F bit 4 (default)** **Tachometer input pin 12**
W3	VP13/ GPIOF5/ TACH13	ID/O8 ID/O8 ID	CMOS	**Video port bit 13** **GPIO group F bit 5 (default)** **Tachometer input pin 13**
W2	VP14/ GPIOF6/ TACH14	ID/O8 ID/O8 ID	CMOS	**Video port bit 14** **GPIO group F bit 6 (default)** **Tachometer input pin 14**
W1	VP15/ GPIOF7/ TACH15	ID/O8 ID/O8 ID	CMOS	**Video port bit 15** **GPIO group F bit 7 (default)** **Tachometer input pin 15**
Y4	VP16/ GPIOG0/	ID/O8 ID/O8	CMOS	**Video port bit 16** **GPIO group G bit 0 (default)**
Y3	VP17/ GPIOG1/	ID/O8 ID/O8	CMOS	**Video port bit 17** **GPIO group G bit 1 (default)**
U2	VPAHSYNC/ HSYNC/ GPIOH4	ID/O8 O8 ID/O8	CMOS	**Video port A horizontal sync** **VGA horizontal sync pin (default)** **GPIO group H bit 4**

to next page

from previous page

R4	VPAVSYNC/ VSYNC/ GPIOH5	ID/O16 O16 ID/O16	CMOS	**Video port A vertical sync** **VGA vertical sync pin (default)** **GPIO group H bit 5**
U1	VPADE/ GPIOH6	ID/O8 ID/O8	CMOS	**Video port A display enable signal** **GPIO group H bit 6 (default)**
T1	VPACLK/ GPIOH7	ID/O8 ID/O8	CMOS	**Video port A display enable signal** **GPIO group H bit 7 (default)**
B2	DDCADAT/ GPIOD6	IU/O8 IU/O8	CMOS	**Channel A DDC data pin (default)** **GPIO group D bit 6**
B1	DDCACLK/ GPIOD7	IU/O8 IU/O8	CMOS	**Channel A DDC clock pin (default)** **GPIO group D bit 7**
Note : VP0 ~ VP15 are default at input tri-state mode, so it can be GPI or TACH function.				

MII/RMII Dual Interface 18 pins

Ball	Signal	I/O	Type	Description
A4	MIITXD0/ RMIITXD0	O8 O8	CMOS	**MII 1 transmit data bus to PHY bit 0** **RMII 1 transmit data bus to PHY bit 0**
B4	MIITXD1/ RMIITXD1	O8 O8	CMOS	**MII 1 transmit data bus to PHY bit 1** **RMII 1 transmit data bus to PHY bit 1**
C4	MIITXD2/ RMII2TXD0/ GPIOE0	O8 O8 ID/O8	CMOS	**MII 1 transmit data bus to PHY bit 2** **RMII 2 transmit data bus to PHY bit 0** **GPIO group E bit 0**
D4	MIITXD3/ RMII2TXD1/ GPIOE1	O8 O8 ID/O8	CMOS	**MII 1 transmit data bus to PHY bit 3** **RMII 2 transmit data bus to PHY bit 1** **GPIO group E bit 0**
C5	MIITXEN/ RMIITXEN	O8 O8	CMOS	**MII 1 transmit enable** **RMII 1 transmit enable**
D5	MIITXER/ RMII2TXEN/ GPIOE4	O8 O8 ID/O8	CMOS	**MII 1 transmit error** **RMII 2 transmit enable** **GPIO group E bit 4**
A7	MIITXCK/ RMIIRCLK	ID ID	CMOS	**MII 1 transmit clock** **RMII 1 50MHz reference clock**
B7	MIIRXCK/ RMII2RCLK/ GPIOE5	ID ID ID/O8	CMOS	**MII 1 receive clock** **RMII 2 50MHz reference clock** **GPIO group E bit 5**
C6	MIIRXD0/ RMIIRXD0	ID ID	CMOS	**MII 1 receive data bus from PHY bit 0** **RMII 1 receive data bus from PHY bit 0**
D6	MIIRXD1/ RMIIRXD1	ID ID	CMOS	**MII 1 receive data bus from PHY bit 1** **RMII 1 receive data bus from PHY bit 1**
A5	MIIRXD2/ RMII2RXD0/ GPIOE2	ID ID ID/O8	CMOS	**MII 1 receive data bus from PHY bit 2** **RMII 2 receive data bus from PHY bit 0** **GPIO group E bit 2**

to next page

from previous page

B5	MIIRXD3/ RMII2RXD1/ GPIOE3	ID ID ID/O8	CMOS	**MII 1 receive data bus from PHY bit 3** **RMII 2 receive data bus from PHY bit 1** **GPIO group E bit 3**
D7	MIIRXDV/ RMIICRSDV	ID ID	CMOS	**MII 1 receive data valid** **RMII 1 receive carrier sense and data valid**
C7	MIIRXER/ RMIIRXER	ID ID	CMOS	**MII 1 receive data error** **RMII 1 receive data error**
B6	MIICRS/ RMII2CRSDV/ GPIOE6	ID ID ID/O8	CMOS	**MII 1 carrier sense** **RMII 2 receive carrier sense and data valid** **GPIO group E bit 6**
A6	MIICOL/ RMII2RXER/ GPIOE7	ID ID ID/O8	CMOS	**MII 1 collision** **RMII 2 receive data error** **GPIO group E bit 7**
A2	MIIMDIO	ID/O8	CMOS	**MAC management data input/output**
A3	MIIMDC	O8	CMOS	**MAC management data clock**

Note :
The default pin function is determined by hardware trapping bit[8:6]

UART Port 10 pins

Ball	Signal	I/O	Type	Description
W22	NCTS1	I	CMOS	**UART 1 clear to send modem status**
V19	NDCD1	I	CMOS	**UART 1 data carrier detect modem status**
V20	NDSR1	I	CMOS	**UART 1 data set ready modem status**
V22	NRI1	I	CMOS	**UART 1 ring indicator modem status**
U19	NDTR1	O8	CMOS	**UART 1 data terminate ready modem status**
V21	NRTS1	O8	CMOS	**UART 1 request to send modem status**
Y22	TXD1	O8	CMOS	**UART 1 transmit serial data output**
AA22	RXD1	I	CMOS	**UART 1 receive serial data input**
U21	TXD2	O8	CMOS	**UART 2 transmit serial data output**
U20	RXD2	I	CMOS	**UART 2 receive serial data input**

JTAG Port 6 pins

Ball	Signal	I/O	Type	Description
U22	TCK	IU	CMOS	**JTAG input clock** Clock input used to synchronize JTAG control and data transfer.
T22	RTCK	O12	CMOS	**JTAG output clock** JTAG TAP output clock for test only.
T21	TDI	IU	CMOS	**JTAG test data input** Serial data input to the JTAG TAP controller.
R19	TDO	O12	CMOS	**JTAG test data output** Serial data output from the JTAG TAP controller.
T19	TMS	IU	CMOS	**JTAG test mode select** Signal control input to the JTAG TAP controller is used to select the test logic state machine.

to next page

from previous page

T20	NTRST	IU	CMOS	**JTAG test reset** Asynchronous reset input to JTAG TAP controller. Must be low to ensure the TAP controller initialized to the test logic reset state.

Static Memory Port 29 pins

Ball	Signal	I/O	Type	Description
AB9	ROMCS0#	O8	CMOS	**Static memory chip select 0** SPI type flash chip select.
W7	ROMCS2#	O8	CMOS	**Static memory chip select 2** Default booting SPI type flash chip select.
Y2 Y1 AA4	ROMD0 ROMD1 ROMD2	ID/O8	CMOS	**ROMD0 : SPI flash CLK** **ROMD1 : SPI flash DO** **ROMD2 : SPI flash DI** These signals function as **Hardware Trapping** input when SRST# is **Low**. When SRST# is **High**, it functions as flash data bus or control signal depending on which type of flash is active. The trapping value of these bits doesn't affect hardware behavior. They are reserved for Software usage. Trapping value '0' : no external resistor required. Trapping value '1' : external pull-high resistor required.
AA9 Y9 W9 AB8 AA8 Y8 W8 AB7 AA5 Y5 W5 AB2 AB3 AB4 AA1 AA2 AA3 AA7 Y7 AB5 W6 Y6 AA6 AB6	ROMA0 ROMA1 ROMA2 ROMA3 ROMA4 ROMA5 ROMA6 ROMA7 ROMA8 ROMA9 ROMA10 ROMA11 ROMA12 ROMA13 ROMA14 ROMA15 ROMA16 ROMA17 ROMA18 ROMA19 ROMA20 ROMA21 ROMA22 ROMA23	ID/O8	CMOS	**ROMA[23:0] : Hardware trapping pins** These signals function as **Hardware Trapping** input when SRST# is **Low**. When SRST# is **High**, it functions as flash address bus or control signal depending on which type of flash is active. Trapping value '0' : no external resistor required. Trapping value '1' : external 3.3KΩ pull-high resistor required. The hardware trapping function for each bit defined as follows : ROMA[1:0] : ARM CPU boot code selection 00 : Reserved 01 : Reserved 10 : Boot from ROMCS2#, SPI flash memory 11 : Disable ARM CPU operation ROMA[3:2] : VGA memory size selection 00 : Select 8 MB VGA memory (default) 01 : Select 16 MB VGA memory 10 : Select 32 MB VGA memory 11 : Select 64 MB VGA memory VGA memory will share with SOC memory from SDRAM Controller. ROMA4 : Reserved, must always be 0 ROMA5 : Enable VGA BIOS ROM 0: No VGA BISO ROM (for on-board applications) 1: Enable VGA BIOS ROM (for add-on applications)

to next page

from previous page

				ROMA[8:6] : MAC interface mode selection 000: Reserved 001: Reserved 010: Reserved 011: Select MII(1) only 100: Select RMII(1) only 101: Reserved 110: Select RMII(1) and RMII(2) 111: Disable MAC ROMA[11:9] : H-PLL default clock frequency selection 000: Reserved 001: Reserved 010: Select 200 MHz (default) 011: Select 166 MHz 100: Select 133 MHz 101: Select 100 MHz 110: Reserved 111: Select 24 MHz (by enabling H-PLL bypass mode) ROMA[13:12] : CPU/AHB clock frequency ratio selection 00 : Select CPU:AHB = 1:1 01 : Select CPU:AHB = 2:1 (default) 10 : Select CPU:AHB = 4:1 11 : Select CPU:AHB = 3:1 ROMA14 : Bypass VGA DAC 0 : Normal DAC function (default) 1 : Bypass DAC Mode(for test mode only) ROMA15 : PCI Class Code selection 0 : Select the Class Code for video device 1 : Select the Class Code for VGA device ROMA16 : SOC boot up full speed mode 0 : ARM CPU will boot up at low speed mode (1/16 of full speed) 1 : ARM CPU will boot up at full speed mode **The said low speed mode is implemented by CPU throttling logic setting at 1/16 throttling ratio.** **And MPLL will be turned off and MCLK is source from 24MHz reference clock.** **When booting up at low speed mode, software must set this bit to 1 for normal speed operation, else it will always operate at low speed mode.** ROMA17 : PCI VGA Config Space Prefetch bit setting 0: PCI VGA Config prefetch bit return 0 (default) 1: PCI VGA Config prefetch bit return 1 This bit setting controls the PCI Config Space Prefetch bit return value. ROMA18 : Reserved, must always be 0 ROMA19 : Bypass all PLL 0 : No operation (default) 1 : Bypass all PLL (for test mode only) ROMA20 : Disable ARM CPU to M-bus bridge 0 : No operation (default) 1 : Disable ARM CPU to M-Bus bridge When this bit is set, ARM CPU can only access memory data through AHB bus. The direct path to M-bus will be disabled. This is for insurance policy only.

to next page

from previous page

				ROMA21 : PCI AD bus order swap mode 0 : Normal pin sequence 1 : Reversed pin sequence This option is designed to optimize the trace routing when PCB layout. AST2050 / AST1100 can be applied to either on-board or add-on applications. But the optimal pin sequence for the two kinds of application may not the same. So when this bit is set, the following pin shuffle will be applied immediately. AD[31] pin replaces AD[0] pin AD[30] pin replaces AD[1] pin .... AD[0] pin replace AD[31] pin CBE[3] pin replace CBE[0] pin CBE[2] pin replace CBE[1] pin CBE[1] pin replace CBE[2] pin CBE[0] pin replace CBE[3] pin ROMA22 : Enable test mode 0 : Enable normal mode (default) 1 : Enable test mode ROMA23 : Enable LPC dedicated reset pin function 0 : LPC reset is shared with PCI reset pin 1 : LPC reset is located at pin number B10

I2C/SMBUS/FML Port 14 pins

Ball	Signal	I/O	Type	Description
A15	SDA1	IS/O12	CMOS	**I2C/SMBUS 1 data pin, FML1 FLBMD pin** I2C/SMBUS controller 1 bidirectional data pin. FML controller 1 master data output pin. An external pull high resistor is required to connect to this pin.
B15	SCL1	IS/O12	CMOS	**I2C/SMBUS 1 clock pin, FML1 FLBMCK pin** I2C/SMBUS controller 1 bidirectional clock pin. FML controller 1 master clock output pin. An external pull high resistor is required to connect to this pin.
C14	SDA2	IS/O12	CMOS	**I2C/SMBUS 2 data pin, FML2 FLBMD pin** I2C/SMBUS controller 2 bidirectional data pin. FML controller 2 master data output pin. An external pull high resistor is required to connect to this pin.
D14	SCL2	IS/O12	CMOS	**I2C/SMBUS 2 clock pin, FML2 FLBMCK pin** I2C/SMBUS controller 2 bidirectional clock pin. FML controller 2 master clock output pin. An external pull high resistor is required to connect to this pin.
A14	SDA3	IS/O12	CMOS	**I2C/SMBUS 3 data pin** I2C/SMBUS controller 3 bidirectional data pin. An external pull high resistor is required to connect to this pin.
B14	SCL3	IS/O12	CMOS	**I2C/SMBUS 3 clock pin** I2C/SMBUS controller 3 bidirectional clock pin. An external pull high resistor is required to connect to this pin.
C13	SDA4	IS/O12	CMOS	**I2C/SMBUS 4 data pin** I2C/SMBUS controller 4 bidirectional data pin. An external pull high resistor is required to connect to this pin.

to next page

from previous page

D13	SCL4	IS/O12	CMOS	**I2C/SMBUS 4 clock pin** I2C/SMBUS controller 4 bidirectional clock pin. An external pull high resistor is required to connect to this pin.
A13	SDA5/ GPIOC6/	IS/O12 IS/O12	CMOS	**I2C/SMBUS 5 data pin** **GPIO group C bit 6 (default)**
B13	SCL5/ GPIOC7/	IS/O12 IS/O12	CMOS	**I2C/SMBUS 5 clock pin** **GPIO group C bit 7 (default)**
C12	SDA6/ GPIOH0	IS/O12 IS/O12	CMOS	**I2C/SMBUS 6 data pin, FML2 FLBSD pin** **GPIO group H bit 0 (default)**
D12	SCL6/ GPIOH1	IS/O12 IS/O12	CMOS	**I2C/SMBUS 6 clock pin, FML2 FLBINTCKEX pin** **GPIO group H bit 1 (default)**
A12	SDA7/ SALT2/ GPIOH2	IS/O12 IS/O12 IS/O12	CMOS	**I2C/SMBUS 7 data pin, FML1 FLBSD pin** **SMBus 2 ALT pin** **GPIO group H bit 2 (default)**
B12	SCL7/ SALT1/ GPIOH3	IS/O12 IS/O12 IS/O12	CMOS	**I2C/SMBUS 7 clock pin, FML1 FLBINTCKEX pin** **SMBus 1 ALT pin** **GPIO group H bit 3 (default)**

GPIO Port 15 pins

Ball	Signal	I/O	Type	Description
D11	GPIOA4/ PHYLINK	I/O16 I	CMOS	**GPIO group A bit 4 (default)** **PHY link status for MAC 1**
C11	GPIOA5/ PHYPD#	I/O16 O16	CMOS	**GPIO group A bit 5 (default)** **PHY power down control for MAC 1**
A11	GPIOB1/ FLBUSY#	IS/O16 IS	CMOS	**GPIO group B bit 1 (default)** **Flash memory busy status input**
D10	GPIOB2/ FLWP#	IS/O16 O16	CMOS	**GPIO group B bit 2 (default)** **Flash memory write protect**
C10	GPIOB3/	IS/O8	CMOS	**GPIO group B bit 3 (default)**
B10	GPIOB4/ VBCS/ LRST#	IS/O8 O8 IS/O8	CMOS	**GPIO group B bit 4** **VGA BIOS SPI Flash ROM chip select output** **Dedicated LPC reset input/output**
A10	GPIOB5/ VBCK	IS/O16 O16	CMOS	**GPIO group B bit 5** **VGA BIOS SPI Flash ROM clock output**
D9	GPIOB6/ VBDO/ WDTRST	IS/O8 O8 O8	CMOS	**GPIO group B bit 6** **VGA BIOS SPI Flash ROM data output** **Watchdog reset output (Not work at A0 version)** Please reference Section 27 for detailed usage.
C9	GPIOB7/ VBDI/ EXTRST#	IS/O8 IS IS	CMOS	**GPIO group B bit 7** **VGA BIOS SPI Flash ROM data input** **External SOC reset input (Not work at A0 version)**
B9	GPIOC0/ PECII	I/O8 I	CMOS	**GPIO group C bit 0 (default)** **PECI input pin**

to next page

from previous page

A9	GPIOC1/ PECIO	I/O8 O8	CMOS	**GPIO group C bit 1 (default)** **PECI output pin**
D8	GPIOC2/ PWM1	I/O8 O8	CMOS	**GPIO group C bit 2 (default)** **PWM fan control pin 1**
C8	GPIOC3/ PWM2	I/O8 O8	CMOS	**GPIO group C bit 3 (default)** **PWM fan control pin 2**
B8	GPIOC4/ PWM3	I/O8 O8	CMOS	**GPIO group C bit 4 (default)** **PWM fan control pin 3**
A8	GPIOC5/ PWM4	I/O8 O8	CMOS	**GPIO group C bit 5 (default)** **PWM fan control pin 4**

Note :
The default pin function of GPIOB4 $\sim$ GPIOB7 are determined by hardware trapping bit[5]

Miscellaneous 3 pins

Ball	Signal	I/O	Type	Description
R21	ENTEST	IS	CMOS	**Enable test mode** Enable chip to enter test mode. Tie the ENTEST to ground in normal operation mode.
R20	SRST#	IS	CMOS	**ARM SOC system reset pin** Keep SRST# Low after power-on and power stable for a period of time (minimum 10ms). It will reset the SOC part function, no effect to the PCI VGA functions.
R22	CLKIN	I	CMOS	**External reference clock input pin** Connect this pin to a 24MHz oscillator source.

USB 2.0 Port 11 pins

Ball	Signal	I/O	Type	Description
A21	USB_DN	I/O	A	**USB 2.0 $D-$ signal from USB cable**
B22	USB_DP	I/O	A	**USB 2.0 $D+$ signal from USB cable**
B21	USBRPU	-	A	**Connected to an external 1.5KΩ pull-up resistor**
C17	USBVRES	-	A	**Connected an external 8.2KΩ resistor to USBVSSA for band-gap reference circuit.**
C18	USBVSDL	-	P	**USB digital ground**
A20	USBVSSA1	-	P	**USB analog ground**
A18	USBVSSA2	-	P	**USB analog ground**
D18	USBV12L	-	P	**1.26V USB digital power**
B20	USBV33A1	-	P	**3.3V USB analog power**
B18	USBV33A2	-	P	**3.3V USB analog power**
A19	USBV33D	-	P	**3.3V USB digital power**

PLL Power 21 pins

Ball	Signal	I/O	Type	Description
H2	PLL1PV12	-	P	**1.26V PLL I/O power**
H1	PLL1PO33	-	P	**3.3V PLL I/O power**

to next page

from previous page

K4	PLL2PV12	-	P	**1.26V VPLL I/O power**
K3	PLL2PO33	-	P	**3.3V VPLL I/O power**
N3	PLL2PVSS	-	P	**PLL I/O ground**
J2	V1PLLDV12	-	P	**1.26V V1PLL digital power**
J1	V1PLLDVSS	-	P	**V1PLL ground**
J4	V1PLLAV12	-	P	**1.26V V1PLL analog power**
J3	V1PLLAVSS	-	P	**V1PLL ground**
L2	MPLLDV12	-	P	**1.26V MPLL digital power**
L1	MPLLDVSS	-	P	**MPLL ground**
K2	MPLLAV33	-	P	**3.3V MPLL analog power**
K1	MPLLAVSS	-	P	**MPLL analog ground**
L4	MPLLAV33G	-	P	**3.3V MPLL analog power**
L3	MPLLAVSSG	-	P	**MPLL analog ground**
N1	HPLLDV12	-	P	**1.26V HPLL digital power**
N2	HPLLDVSS	-	P	**HPLL digital ground**
M4	HPLLAV33	-	P	**3.3V HPLL analog power**
M3	HPLLAVSS	-	P	**HPLL analog ground**
M2	HPLLAV33G	-	P	**3.3V HPLL analog power**
M1	HPLLAVSSG	-	P	**HPLL analog ground**

DAC 19 pins

Ball	Signal	I/O	Type	Description
C3	DACDV12	-	P	**1.26V DAC digital power**
C2	DACDVSS	-	P	**DAC digital ground**
H3	DACDHV33	-	P	**3.3V DAC digital power**
H4	DACDHVSS	-	P	**DAC digital ground**
G1	DACAV33	-	P	**3.3V DAC analog power**
G2	DACAVSS	-	P	**DAC analog ground**
F1	DACAV33R	-	P	**3.3V DAC analog power**
F2	DACAVSSR	-	P	**DAC analog ground**
E3	DACAV33G	-	P	**3.3V DAC analog power**
E2	DACAVSSG	-	P	**DAC analog ground**
D3	DACAV33B	-	P	**3.3V DAC analog power**
D2	DACAVSSB	-	P	**DAC analog ground**
E1	DACR	O	A	**DAC R component output**
D1	DACG	O	A	**DAC G component output**
C1	DACB	O	A	**DAC B component output**
G4	DACRSET	-	A	**DAC reference resistor** An external resistor 2.43KΩ connecting DACRSET pin to DACAVSS adjusts the magnitude of DAC full-scale output current.
F3	DACCOMP	-	A	**DAC compensation pin** This pin should be connected through a 0.01uF ceramic capacitor, parallel with a 10uF tantalum capacitor, to DACAV33 externally.

to next page

from previous page

G3	DACVREFIN	-	A	**DAC reference voltage input** It is suggested to place 0.1uF ceramic capacitor between this pin and DACAVSS pin externally.
F4	DACVREF	-	A	**DAC reference voltage output** This output delivers 1.2048V reference voltage from cell. It is normally connected to DACVREFIN pin.

Power 82 pins

Ball	Signal	I/O	Type	Description
E4	IV12D	-	P	**1.26V Core logic voltage VDD**
E5	IV12D	-	P	
E6	IV12D	-	P	
E11	IV12D	-	P	
E12	IV12D	-	P	
E13	IV12D	-	P	
M18	IV12D	-	P	
N18	IV12D	-	P	
P5	IV12D	-	P	
R5	IV12D	-	P	
T5	IV12D	-	P	
V14	IV12D	-	P	
V15	IV12D	-	P	
V16	IV12D	-	P	
U18	MVDD	-	P	**1.8V DDR2 SDRAM memory I/O VDD**
V11	MVDD	-	P	**2.6V DDR SDRAM memory I/O VDD**
V12	MVDD	-	P	
V13	MVDD	-	P	
V17	MVDD	-	P	
V18	MVDD	-	P	
E7	PV33D	-	P	**3.3V General purpose I/O VDD**
E8	PV33D	-	P	
E14	PV33D	-	P	
E15	PV33D	-	P	
E16	PV33D	-	P	
J18	PV33D	-	P	
K18	PV33D	-	P	
L5	PV33D	-	P	
L18	PV33D	-	P	
M5	PV33D	-	P	
U5	PV33D	-	P	
V5	PV33D	-	P	
V6	PV33D	-	P	
V7	PV33D	-	P	
V8	PV33D	-	P	

to next page

from previous page

A1	GND	-	P	**Digital ground**
A22	GND	-	P	
B3	GND	-	P	
B19	GND	-	P	
D17	GND	-	P	
E9	GND	-	P	
E10	GND	-	P	
E17	GND	-	P	
E18	GND	-	P	
F18	GND	-	P	
G18	GND	-	P	
G19	GND	-	P	
H5	GND	-	P	
J5	GND	-	P	
K5	GND	-	P	
K10	GND	-	P	
K11	GND	-	P	
K12	GND	-	P	
K13	GND	-	P	
L10	GND	-	P	
L11	GND	-	P	
L12	GND	-	P	
L13	GND	-	P	
M10	GND	-	P	
M11	GND	-	P	
M12	GND	-	P	
M13	GND	-	P	
N10	GND	-	P	
N11	GND	-	P	
N12	GND	-	P	
N13	GND	-	P	
P18	GND	-	P	
P19	GND	-	P	
P20	GND	-	P	
R18	GND	-	P	
V9	GND	-	P	
V10	GND	-	P	
AB1	GND	-	P	
AB22	GND	-	P	
F5	PLLVSS	-	P	**PLL substrate ground**
G5	PLLVSS	-	P	
N4	PLLVSS	-	P	
N5	PLLVSS	-	P	

to next page

from previous page

P1	PLLVSS	-	P	
P2	PLLVSS	-	P	
P3	PLLVSS	-	P	
P4	PLLVSS	-	P	

3.2 Ball Mapping Table

Signal	Ball	Signal	Ball	Signal	Ball	Signal	Ball
AD0	C22	AD1	C21	AD10	E19	AD11	F22
AD12	F21	AD13	F20	AD14	F19	AD15	G22
AD16	J20	AD17	J19	AD18	K22	AD19	K21
AD2	C20	AD20	K20	AD21	K19	AD22	L22
AD23	L21	AD24	M22	AD25	M21	AD26	M20
AD27	M19	AD28	N22	AD29	N21	AD3	C19
AD30	N20	AD31	N19	AD4	D22	AD5	D21
AD6	D20	AD7	D19	AD8	E21	AD9	E20
BA0	Y17	BA1	AA17	BA2	Y16	BRST#	P21
CAS#	AB16	CBE0#	E22	CBE1#	G21	CBE2#	J21
CBE3#	L19	CK	AA19	CK#	AB19	CKE	AB18
CLKIN	R22	CS#	W16	DACAV33	G1	DACAV33B	D3
DACAV33G	E3	DACAV33R	F1	DACAVSS	G2	DACAVSSB	D2
DACAVSSG	E2	DACAVSSR	F2	DACB	C1	DACCOMP	F3
DACDHV33	H3	DACDHVSS	H4	DACDV12	C3	DACDVSS	C2
DACG	D1	DACR	E1	DACRSET	G4	DACVREF	F4
DACVREFIN	G3	DDCACLK/ GPIOD7	B1	DDCADAT/ GPIOD6	B2	DEVSEL#	H21
DM0	W10	DM1	W15	DQ0	AA12	DQ1	Y12
DQ10	AA14	DQ11	Y14	DQ12	W14	DQ13	AB13
DQ14	AA13	DQ15	Y13	DQ2	W12	DQ3	AB11
DQ4	AA11	DQ5	Y11	DQ6	W11	DQ7	Y10
DQ8	Y15	DQ9	AA15	DQS0	AA10	DQS0#	AB10
DQS1	AB14	DQS1#	AB15	ENTEST	R21	FRAME#	J22
GND	A1	GND	A22	GND	B3	GND	B19
GND	D17	GND	E9	GND	E10	GND	E17
GND	E18	GND	F18	GND	G18	GND	G19
GND	H5	GND	J5	GND	K5	GND	K10
GND	K11	GND	K12	GND	K13	GND	L10
GND	L11	GND	L12	GND	L13	GND	M10
GND	M11	GND	M12	GND	M13	GND	N10
GND	N11	GND	N12	GND	N13	GND	P18
GND	P19	GND	P20	GND	R18	GND	V9
GND	V10	GND	AB1	GND	AB22	GPIOA4/ PHYLINK	D11

to next page

from previous page

Signal	Ball	Signal	Ball	Signal	Ball	Signal	Ball
GPIOA5/ PHYPD#	C11	GPIOB1/ FLBUSY#	A11	GPIOB2/ FLWP#	D10	GPIOB3	C10
GPIOB4/ VBCS/ LRST#	B10	GPIOB5/ VBCK	A10	GPIOB6/ VBDO/ WDTRST	D9	GPIOB7/ VBDI/ EXTRST#	C9
GPIOC0/ PECII	B9	GPIOC1/ PECIO	A9	GPIOC2/ PWM1	D8	GPIOC3/ PWM2	C8
GPIOC4/ PWM3	B8	GPIOC5/ PWM4	A8	HPLLAV33	M4	HPLLAV33G	M2
HPLLAVSS	M3	HPLLAVSSG	M1	HPLLDV12	N1	HPLLDVSS	N2
IDSEL	L20	INTA#/ GPIOB0	B11	IRDY#	H19	IV12D	E4
IV12D	E5	IV12D	E6	IV12D	E11	IV12D	E12
IV12D	E13	IV12D	M18	IV12D	N18	IV12D	P5
IV12D	R5	IV12D	T5	IV12D	V14	IV12D	V15
IV12D	V16	LAD0	B17	LAD1	A17	LAD2	D16
LAD3	C16	LCLK	A16	LFRAME#	B16	LPCPD#	D15
LPCSIRQ	C15	MA0	AB17	MA1	W18	MA10	W21
MA11	Y21	MA12	AA21	MA2	Y18	MA3	AA18
MA4	W19	MA5	Y19	MA6	W20	MA7	Y20
MA8	AA20	MA9	AB20	MIICOL/ RMII2RXER/ GPIOE7	A6	MIICRS/ RMII2CRSDV/ GPIOE6	B6
MIIMDC	A3	MIIMDIO	A2	MIIRXCK/ RMII2RCLK/ GPIOE5	B7	MIIRXD0/ RMIIRXD0	C6
MIIRXD1/ RMIIRXD1	D6	MIIRXD2/ RMII2RXD0/ GPIOE2	A5	MIIRXD3/ RMII2RXD1/ GPIOE3	B5	MIIRXDV/ RMIICRSDV	D7
MIIRXER/ RMIIRXER	C7	MIITXCK/ RMIIRCLK	A7	MIITXD0/ RMIITXD0	A4	MIITXD1/ RMIITXD1	B4
MIITXD2/ RMII2TXD0/ GPIOE0	C4	MIITXD3/ RMII2TXD1/ GPIOE1	D4	MIITXEN/ RMIITXEN	C5	MIITXER/ RMII2TXEN/ GPIOE4	D5
MPLLAV33	K2	MPLLAV33G	L4	MPLLAVSS	K1	MPLLAVSSG	L3
MPLLDV12	L2	MPLLDVSS	L1	MVDD	U18	MVDD	V11
MVDD	V12	MVDD	V13	MVDD	V17	MVDD	V18
NCTS1	W22	NDCD1	V19	NDSR1	V20	NDTR1	U19
NRI1	V22	NRTS1	V21	NTRST	T20	ODT	AB21
PAR	G20	PCICLK	P22	PLL1PO33	H1	PLL1PV12	H2
PLL2PO33	K3	PLL2PV12	K4	PLL2PVSS	N3	PLLVSS	F5
PLLVSS	G5	PLLVSS	N4	PLLVSS	N5	PLLVSS	P1
PLLVSS	P2	PLLVSS	P3	PLLVSS	P4	PV33D	E7
PV33D	E8	PV33D	E14	PV33D	E15	PV33D	E16

to next page

from previous page

Signal	Ball	Signal	Ball	Signal	Ball	Signal	Ball
PV33D	J18	PV33D	K18	PV33D	L5	PV33D	L18
PV33D	M5	PV33D	U5	PV33D	V5	PV33D	V6
PV33D	V7	PV33D	V8	RAS#	AA16	ROMA0	AA9
ROMA1	Y9	ROMA10	W5	ROMA11	AB2	ROMA12	AB3
ROMA13	AB4	ROMA14	AA1	ROMA15	AA2	ROMA16	AA3
ROMA17	AA7	ROMA18	Y7	ROMA19	AB5	ROMA2	W9
ROMA20	W6	ROMA21	Y6	ROMA22	AA6	ROMA23	AB6
ROMA3	AB8	ROMA4	AA8	ROMA5	Y8	ROMA6	W8
ROMA7	AB7	ROMA8	AA5	ROMA9	Y5	ROMCS0#	AB9
ROMCS2#	W7	ROMD0	Y2	ROMD1	Y1	ROMD2	AA4
RTCK	T22	RXD1	AA22	RXD2	U20	SCL1	B15
SCL2	D14	SCL3	B14	SCL4	D13	SCL5/ GPIOC7	B13
SCL6/ GPIOH1	D12	SCL7/ SALT1/ GPIOH3	B12	SDA1	A15	SDA2	C14
SDA3	A14	SDA4	C13	SDA5/ GPIOC6	A13	SDA6/ GPIOH0	C12
SDA7/ SALT2/ GPIOH2	A12	SRST#	R20	STOP#	H22	TCK	U22
TDI	T21	TDO	R19	TMS	T19	TRDY#	H20
TXD1	Y22	TXD2	U21	USB_DN	A21	USB_DP	B22
USBRPU	B21	USBV12L	D18	USBV33A1	B20	USBV33A2	B18
USBV33D	A19	USBVRES	C17	USBVSDL	C18	USBVSSA1	A20
USBVSSA2	A18	V1PLLAV12	J4	V1PLLAVSS	J3	V1PLLDV12	J2
V1PLLDVSS	J1	VP0/ GPIOE0/ TACH0	R3	VP1/ GPIOE1/ TACH1	R2	VP10/ GPIOF2/ TACH10	V2
VP11/ GPIOF3/ TACH11	V1	VP12/ GPIOF4/ TACH12	W4	VP13/ GPIOF5/ TACH13	W3	VP14/ GPIOF6/ TACH14	W2
VP15/ GPIOF7/ TACH15	W1	VP16/ GPIOG0	Y4	VP17/ GPIOG1	Y3	VP2/ GPIOE2/ TACH2	R1
VP3/ GPIOE3/ TACH3	T4	VP4/ GPIOE4/ TACH4	T3	VP5/ GPIOE5/ TACH5	T2	VP6/ GPIOE6/ TACH6	U4
VP7/ GPIOE7/ TACH7	U3	VP8/ GPIOF0/ TACH8	V4	VP9/ GPIOF1/ TACH9	V3	VPACLK/ GPIOH7	T1
VPADE/ GPIOH6	U1	VPAHSYNC/ HSYNC/ GPIOH4	U2	VPAVSYNC/ VSYNC/ GPIOH5	R4	VREFSSTL	T18
VREFSSTL	AB12	VSSRSSTL	W13	WE#	W17		

3.3 Ball Map

	1	2	3	4	5	6	7	8	9	10	11
A	GND	MIIMDIO	MIIMDC	MIITXD0/ RMIITXD0	MIIRXD2/ RMII2RXD0/ GPIOE2	MIICOL/ RMII2RXER/ GPIOE7	MIITXCK/ RMIIRCLK	GPIOC5/ PWM4	GPIOC1/ PECIO	GPIOB5/ VBCK	GPIOB1/ FLBUSY#
B	DDCACLK/ GPIOD7/ OSCCLK	DDCADAT/ GPIOD6	GND	MIITXD1/ RMIITXD1	MIIRXD3/ RMII2RXD1/ GPIOE3	MIICRS/ RMII2CRSDV/ GPIOE6	MIIRXCK/ RMII2RCLK/ GPIOE5	GPIOC4/ PWM3	GPIOC0/ PECII	GPIOB4/ VBCS/ LRST#	INTA#/ GPIOB0
C	DACB	DACDVSS	DACDV12	MIITXD2/ RMII2TXD0/ GPIOE0	MIITXEN/ RMIITXEN	MIIRXD0/ RMIIRXD0	MIIRXER/ RMIIRXER	GPIOC3/ PWM2	GPIOB7/ VBDI/ EXTRST#	GPIOB3	GPIOA5/ PHYPD#
D	DACG	DACAVSSB	DACAV33B	MIITXD3/ RMII2TXD1/ GPIOE1	MIITXER/ RMII2TXEN/ GPIOE4	MIIRXD1/ RMIIRXD1	MIIRXDV/ RMIICRSDV	GPIOC2/ PWM1	GPIOB6/ VBDO/ WDTRST	GPIOB2/ FLWP#	GPIOA4/ PHYLINK
E	DACR	DACAVSSG	DACAV33G	IV12D	IV12D	IV12D	PV33D	PV33D	GND	GND	IV12D
F	DACAV33R	DACAVSSR	DACCOMP	DACVREF	PLLVSS						
G	DACAV33	DACAVSS	DACVREFIN	DACRSET	PLLVSS						
H	PLL1PO33	PLL1PV12	DACDHV33	DACDHVSS	GND						
J	V1PLLDVSS	V1PLLDV12	V1PLLAVSS	V1PLLAV12	GND						
K	MPLLAVSS	MPLLAV33	PLL2PO33	PLL2PV12	GND					GND	GND
L	MPLLDVSS	MPLLDV12	MPLLAVSS G	MPLLAV33G	PV33D					GND	GND
M	HPLLAVSSG	HPLLAV33G	HPLLAVSS	HPLLAV33	PV33D					GND	GND
N	HPLLDV12	HPLLDVSS	PLL2PVSS	PLLVSS	PLLVSS					GND	GND
P	PLLVSS	PLLVSS	PLLVSS	PLLVSS	IV12D						
R	VP2/ GPIOE2/ TACH2	VP1/ GPIOE1/ TACH1	VP0/ GPIOE0/ TACH0	VPAVSYNC/ VSYNC/ GPIOH5	IV12D						
T	VPACLK/ GPIOH7	VP5/ GPIOE5/ TACH5	VP4/ GPIOE4/ TACH4	VP3/ GPIOE3/ TACH3	IV12D						
U	VPADE/ GPIOH6	VPAHSYNC/ HSYNC/ GPIOH4	VP7/ GPIOE7/ TACH7	VP6/ GPIOE6/ TACH6	PV33D						
V	VP11/ GPIOF3/ TACH11	VP10/ GPIOF2/ TACH10	VP9/ GPIOF1/ TACH9	VP8/ GPIOF0/ TACH8	PV33D	PV33D	PV33D	PV33D	GND	GND	MVDD
W	VP15/ GPIOF7/ TACH15	VP14/ GPIOF6/ TACH14	VP13/ GPIOF5/ TACH13	VP12/ GPIOF4/ TACH12	ROMA10	ROMA20	ROMCS2#	ROMA6	ROMA2	DM0	DQ6
Y	ROMD1	ROMD0	VP17/ GPIOG1	VP16/ GPIOG0	ROMA9	ROMA21	ROMA18	ROMA5	ROMA1	DQ7	DQ5
AA	ROMA14	ROMA15	ROMA16	ROMD2	ROMA8	ROMA22	ROMA17	ROMA4	ROMA0	DQS0	DQ4
AB	GND	ROMA11	ROMA12	ROMA13	ROMA19	ROMA23	ROMA7	ROMA3	ROMCS0#	DQS0#	DQ3
	1	2	3	4	5	6	7	8	9	10	11

Figure 3: Ball Map – Left Side

	12	13	14	15	16	17	18	19	20	21	22
A	SDA7/ SALT2/ GPIOH2	SDA5/ GPIOC6	SDA3	SDA1	LCLK	LAD1	USBVSSA2	USBV33D	USBVSSA1	USB_DN	GND
B	SCL7/ SALT1/ GPIOH3	SCL5/ GPIOC7	SCL3	SCL1	LFRAME#	LAD0	USBV33A2	GND	USBV33A1	USBRPU	USB_DP
C	SDA6/ GPIOH0	SDA4	SDA2	LPCSIRQ	LAD3	USBVRES	USBVSDL	AD3	AD2	AD1	AD0
D	SCL6/ GPIOH1	SCL4	SCL2	LPCPD#	LAD2	GND	USBV12L	AD7	AD6	AD5	AD4
E	IV12D	IV12D	PV33D	PV33D	PV33D	GND	GND	AD10	AD9	AD8	CBE0#
F							GND	AD14	AD13	AD12	AD11
G							GND	GND	PAR	CBE1#	AD15
H							X	IRDY#	TRDY#	DEVSEL#	STOP#
J							PV33D	AD17	AD16	CBE2#	FRAME#
K	GND	GND					PV33D	AD21	AD20	AD19	AD18
L	GND	GND					PV33D	CBE3#	IDSEL	AD23	AD22
M	GND	GND					IV12D	AD27	AD26	AD25	AD24
N	GND	GND					IV12D	AD31	AD30	AD29	AD28
P							GND	GND	GND	BRST#	PCICLK
R							GND	TDO	SRST#	ENTEST	CLKIN
T							VREFSSTL	TMS	NTRST	TDI	RTCK
U							MVDD	NDTR1	RXD2	TXD2	TCK
V	MVDD	MVDD	IV12D	IV12D	IV12D	MVDD	MVDD	NDCD1	NDSR1	NRTS1	NRI1
W	DQ2	VSSRSSTL	DQ12	DM1	CS#	WE#	MA1	MA4	MA6	MA10	NCTS1
Y	DQ1	DQ15	DQ11	DQ8	BA2	BA0	MA2	MA5	MA7	MA11	TXD1
AA	DQ0	DQ14	DQ10	DQ9	RAS#	BA1	MA3	CK	MA8	MA12	RXD1
AB	VREFSSTL	DQ13	DQS1	DQS1#	CAS#	MA0	CKE	CK#	MA9	ODT	GND

Figure 4: Ball Map – Right Side

3.4 Video Input(AST2050 Only)/Display Output Port Mapping Table

3.4.1 Single Edge Data Mode : 18 Bits Interface

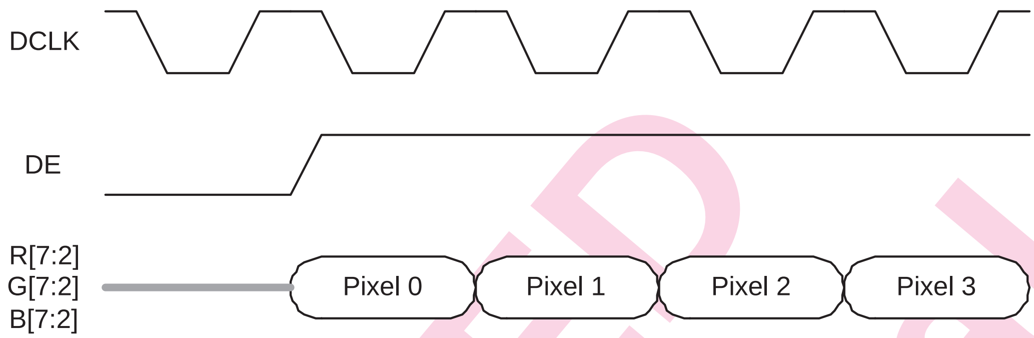


Figure 5: Single Edge Data Interface

Data	Video Port
Red[7:2]	VP[17:12]
Green[7:2]	VP[11:6]
Blue[7:2]	VP[5:0]

3.4.2 Dual Edge Data Mode : 12 Bits Interface

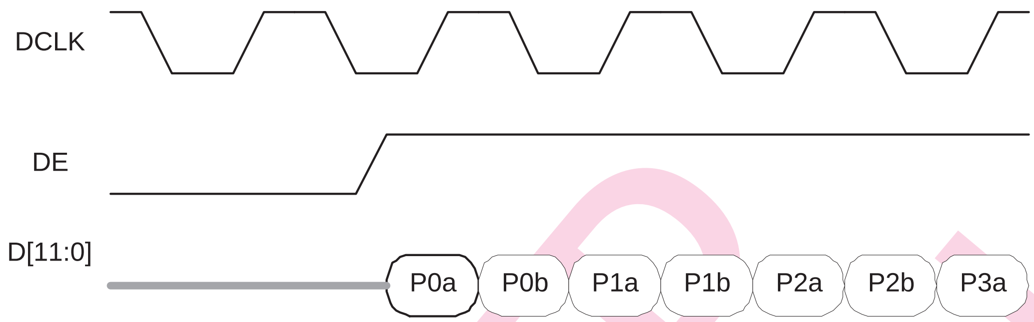


Figure 6: Dual Edge Data Interface

Video Input/Display Output Port

Pixel #	P0a (rise)	P0b (fall)	P1a (rise)	P1b (fall)	P2a (rise)	P2b (fall)
VP11	G0[3]	R0[7]	G1[3]	R1[7]	G2[3]	R2[7]
VP10	G0[2]	R0[6]	G1[2]	R1[6]	G2[2]	R2[6]
VP9	G0[1]	R0[5]	G1[1]	R1[5]	G2[1]	R2[5]
VP8	G0[0]	R0[4]	G1[0]	R1[4]	G2[0]	R2[4]
VP7	B0[7]	R0[3]	B1[7]	R1[3]	B2[7]	R2[3]
VP6	B0[6]	R0[2]	B1[6]	R1[2]	B2[6]	R2[2]
VP5	B0[5]	R0[1]	B1[5]	R1[1]	B2[5]	R2[1]
VP4	B0[4]	R0[0]	B1[4]	R1[0]	B2[4]	R2[0]
VP3	B0[3]	G0[7]	B1[3]	G1[7]	B2[3]	G2[7]
VP2	B0[2]	G0[6]	B1[2]	G1[6]	B2[2]	G2[6]
VP1	B0[1]	G0[5]	B1[1]	G1[5]	B2[1]	G2[5]
VP0	B0[0]	G0[4]	B1[0]	G1[4]	B2[0]	G2[4]

3.5 GPIO Summary

All GPIOs can support input interrupt with sensitive high/low level trigger, rising/falling edge trigger and input debouncing filter. The minimum input pulse widith for edge trigger must larger than 2 PCLK cycle.

GPIO	Ball	Pin Name	Driving	Input Buffer	Internal Resistor
GPIOA4	D11	GPIOA4/PHYLINK	16mA	TTL	NA
GPIOA5	C11	GPIOA5/PHYPD#	16mA	TTL	NA
GPIOB0	B11	INTA#/GPIOB0	16mA	Schmitt	NA
GPIOB1	A11	GPIOB1/FLBUSY#	16mA	Schmitt	NA
GPIOB2	D10	GPIOB2/FLWP#	16mA	Schmitt	NA
GPIOB3	C10	GPIOB3	8mA	Schmitt	NA
GPIOB4	B10	GPIOB4/VBCS/ LRST#	8mA	Schmitt	NA
GPIOB5	A10	GPIOB5/VBCK	16mA	Schmitt	NA
GPIOB6	D9	GPIOB6/VBDO/WDTRST	8mA	Schmitt	NA
GPIOB7	C9	GPIOB7/VBDI/EXTRST#	8mA	Schmitt	NA
GPIOC0	B9	GPIOC0/PECII	8mA	TTL	NA
GPIOC1	A9	GPIOC1/PECIO	8mA	TTL	NA
GPIOC2	D8	GPIOC2/PWM1	8mA	TTL	NA
GPIOC3	C8	GPIOC3/PWM2	8mA	TTL	NA
GPIOC4	B8	GPIOC4/PWM3	8mA	TTL	NA
GPIOC5	A8	GPIOC5/PWM4	8mA	TTL	NA
GPIOC6	A13	SDA5/GPIOC6	12mA	Schmitt	NA
GPIOC7	B13	SCL5/GPIOC7	12mA	Schmitt	NA
GPIOD6	B2	DDCADAT/GPIOD6	8mA	TTL	Pull-Up
GPIOD7	B1	DDCACLK/GPIOD7	8mA	TTL	Pull-Up
GPIOE0(1)	R3	VP0/GPIOE0/TACH0	8mA	TTL	Pull-Down
GPIOE1(1)	R2	VP1/GPIOE1/TACH1	8mA	TTL	Pull-Down
GPIOE2(1)	R1	VP2/GPIOE2/TACH2	8mA	TTL	Pull-Down
GPIOE3(1)	T4	VP3/GPIOE3/TACH3	8mA	TTL	Pull-Down
GPIOE4(1)	T3	VP4/GPIOE4/TACH4	8mA	TTL	Pull-Down
GPIOE5(1)	T2	VP5/GPIOE5/TACH5	8mA	TTL	Pull-Down
GPIOE6(1)	U4	VP6/GPIOE6/TACH6	8mA	TTL	Pull-Down
GPIOE7(1)	U3	VP7/GPIOE7/TACH7	8mA	TTL	Pull-Down
GPIOE0(2)	C4	MIITXD2/RMII2TXD0/GPIOE0	8mA	TTL	Pull-Down
GPIOE1(2)	D4	MIITXD3/RMII2TXD1/GPIOE1	8mA	TTL	Pull-Down
GPIOE2(2)	A5	MIIRXD2/RMII2RXD0/GPIOE2	8mA	TTL	Pull-Down
GPIOE3(2)	B5	MIIRXD3/RMII2RXD1/GPIOE3	8mA	TTL	Pull-Down
GPIOE4(2)	D5	MIITXER/RMII2TXEN/GPIOE4	8mA	TTL	Pull-Down
GPIOE5(2)	B7	MIIRXCK/RMII2RCLK/GPIOE5	8mA	TTL	Pull-Down
GPIOE6(2)	B6	MIICRS/RMII2CRSDV/GPIOE6	8mA	TTL	Pull-Down
GPIOE7(2)	A6	MIICOL/RMII2RXER/GPIOE7	8mA	TTL	Pull-Down
GPIOF0	V4	VP8/GPIOF0/TACH8	8mA	TTL	Pull-Down
GPIOF1	V3	VP9/GPIOF1/TACH9	8mA	TTL	Pull-Down

to next page

from previous page

GPIOF2	V2	VP10/GPIOF2/TACH10	8mA	TTL	Pull-Down
GPIOF3	V1	VP11/GPIOF3/TACH11	8mA	TTL	Pull-Down
GPIOF4	W4	VP12/GPIOF4/TACH12	8mA	TTL	Pull-Down
GPIOF5	W3	VP13/GPIOF5/TACH13	8mA	TTL	Pull-Down
GPIOF6	W2	VP14/GPIOF6/TACH14	8mA	TTL	Pull-Down
GPIOF7	W1	VP15/GPIOF7/TACH15	8mA	TTL	Pull-Down
GPIOG0	Y4	VP16/GPIOG0	8mA	TTL	Pull-Down
GPIOG1	Y3	VP17/GPIOG1	8mA	TTL	Pull-Down
GPIOH0	C12	SDA6/GPIOH0	12mA	Schmitt	NA
GPIOH1	D12	SCL6/GPIOH1	12mA	Schmitt	NA
GPIOH2	A12	SDA7/SALT2/GPIOH2	12mA	Schmitt	NA
GPIOH3	B12	SCL7/SALT1/GPIOH3	12mA	Schmitt	NA
GPIOH4	U2	VPAHSYNC/HSYNC/GPIOH4	8mA	TTL	Pull-Down
GPIOH5	R4	VPAVSYNC/VSYNC/GPIOH5	16mA	TTL	Pull-Down
GPIOH6	U1	VPADE/GPIOH6	8mA	TTL	Pull-Down
GPIOH7	T1	VPACLK/GPIOH7	8mA	TTL	Pull-Down

Note: GPIOE has 2 groups, only 1 group can be choose at a time. Using SCUR74[27] to select.

4 Electrical Specifications

4.1 Absolute Maximum Ratings

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Core, DAC, PLL and USB power	IV12D DACDV12 HPLLDV12 MPLLDV12 PLL1PV12 PLL2PV12 USBV12L V1PLLAV12 V1PLLDV12	GND-0.3		1.44	V
DDR power	MVDD	GND-0.3		3	V
DDR2 power	MVDD	GND-0.3		2.16	V
I/O, DAC, PLL and USB power	DACAV33 DACAV33B DACAV33G DACAV33R DACDHV33 HPLLAV33 HPLLAV33G MPLLAV33 MPLLAV33G PLL1PO33 PLL2PO33 USBV33A1 USBV33A2 USBV33D PV33D	GND-0.3		3.96	V
Storage temperature	TSTG	-40		125	^oC

4.2 Recommended Operating Conditions

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Core, DAC, PLL and USB power	IV12D DACDV12 HPLLDV12 MPLLDV12 PLL1PV12 PLL2PV12 USBV12L V1PLLAV12 V1PLLDV12	1.2	1.26	1.32	V
DDR power	MVDD	2.5	2.6	2.7	V
DDR2 power	MVDD	1.71	1.8	1.89	V

to next page

from previous page

I/O, DAC, PLL and USB power	DACAV33 DACAV33B DACAV33G DACAV33R DACDHV33 HPLLAV33 HPLLAV33G MPLLAV33 MPLLAV33G PLL1PO33 PLL2PO33 USBV33A1 USBV33A2 USBV33D PV33D	3.135	3.3	3.465	V
Ambient operation temperature	TA	0		70	^oC

4.3 Operating Powers

The power are measured based on the following configurations:

1. 64MB DDR2 SDRAM: Qimonda HYB18TC512160BF-3S x 1
2. 8MB SPI Flash: ST 25PB64V6P x 1
3. CPU frequency: 200 MHz
4. DRAM frequency: DDR2-400 (200 MHz)

1.2V power	Average	max. 500 mA	AST2050 / AST1100 only
	Peak	max. 700 mA	
1.8V power	Average	max. 400 mA	include DRAM
	Peak	max. 700 mA	
3.3V power	Average	max. 200 mA	AST2050 / AST1100 only
	Peak	max. 300 mA	

4.4 I/O DC Electrical Specification

CMOS and PCI I/O Parameters

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Input Low Voltage	V_{IL}	-0.3		0.8	V
Input High Voltage	V_{IH}	2.0		5.5	V
Output Low Voltage @ I_{OL}(min)	V_{OL}			0.4	V
Output High Voltage @ I_{OH}(min)	V_{OH}	2.4			V
Threshold Point	V_T	1.17	1.23	1.26	V
Schmitt Trig. Low to High Threshold	V_T+	1.51	1.59	1.65	V
Schmitt Trig. High to Low Threshold	V_T-	0.92	0.98	1.01	V
Input Leakage Current @ V_I=3.3V or 0V	I_L			$\pm$10	μA

to next page

from previous page

Tri-state Output Leakage Current @ V_O=3.3V or 0V	I_{OZ}			±10	μA
Pull-up Resistor[1]	R_{PU}	44	61	92	KΩ
Pull-down Resistor	R_{PD}	33	51	89	KΩ
Low Level Output Current @ V_{OL}=0.4V 8mA	I_{OL}	9.3	15.2	21.3	mA
Low Level Output Current @ V_{OL}=0.4V 12mA	I_{OL}	12.5	20.3	28.5	mA
Low Level Output Current @ V_{OL}=0.4V 16mA	I_{OL}	18.7	30.5	42.6	mA
High Level Output Current @ V_{OH}=2.4V 8mA	I_{OH}	9.57	20.0	34.3	mA
High Level Output Current @ V_{OH}=2.4V 12mA	I_{OH}	13.3	28.0	48.0	mA
High Level Output Current @ V_{OH}=2.4V 16mA	I_{OH}	20.1	42.3	72.5	mA

DDR SSTL2 I/O Parameters

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
SSTL2 supply power voltage	MVDD	2.5	2.6	2.7	V
Reference Voltage	V_{REF}	MVDD/2 - 0.1	MVDD/2	MVDD/2 + 0.1	V
Termination Voltage	V_{TT}	V_{REF}-0.04	V_{REF}	V_{REF}+0.04	V
Input High Voltage	V_{IH}	V_{REF}+0.15		MVDD+0.3	V
Input Low Voltage	V_{IL}	-0.3		V_{REF}-0.15	V
Output minimum source DC current @ V_{OH}=1.74V	I_{OH}	$\lvert -8.1 \rvert$ ClassI $\lvert -16.2 \rvert$ ClassII			mA
Output minimum sink DC current @ V_{OL}=0.56V	I_{OL}	8.1 ClassI 16.2 ClassII			mA

DDR2 SSTL18 I/O Parameters

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
SSTL18 supply power voltage	MVDD	1.71	1.8	1.89	V
Reference Voltage	V_{REF}	0.833	0.9	0.969	V
Termination Voltage	V_{TT}	V_{REF}-0.04	V_{REF}	V_{REF}+0.04	V
Input High Voltage	V_{IH}	V_{REF}+0.125		MVDD+0.3	V
Input Low Voltage	V_{IL}	-0.3		V_{REF}-0.125	V
Output minimum source DC current @ V_{OH}=MVDD-0.28V	I_{OH}	$\lvert -13.4 \rvert$			mA
Output minimum sink DC current @ V_{OL}=0.28V	I_{OL}	13.4			mA

[1] This pull-up only acts on the input path, it will not function normally on the output path, so if need pull-up function externally, add pull-up resistor externally.

DAC I/O Parameters

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
DACDV12 supply DC current	$\lvert I \rvert$	0.2	0.227	0.27	mA
DACDHV33 supply DC current	$\lvert I \rvert$		1		uA
DACAV33 supply DC current	$\lvert I \rvert$	3.3	3.6	3.8	mA
DACAV33R supply DC current	$\lvert I \rvert$		36	38.5	mA
DACAV33G supply DC current	$\lvert I \rvert$		36	38.5	mA
DACAV33B supply DC current	$\lvert I \rvert$		36	38.5	mA
DACVREF output voltage			1.2048		V

4.5 Power Up Sequence

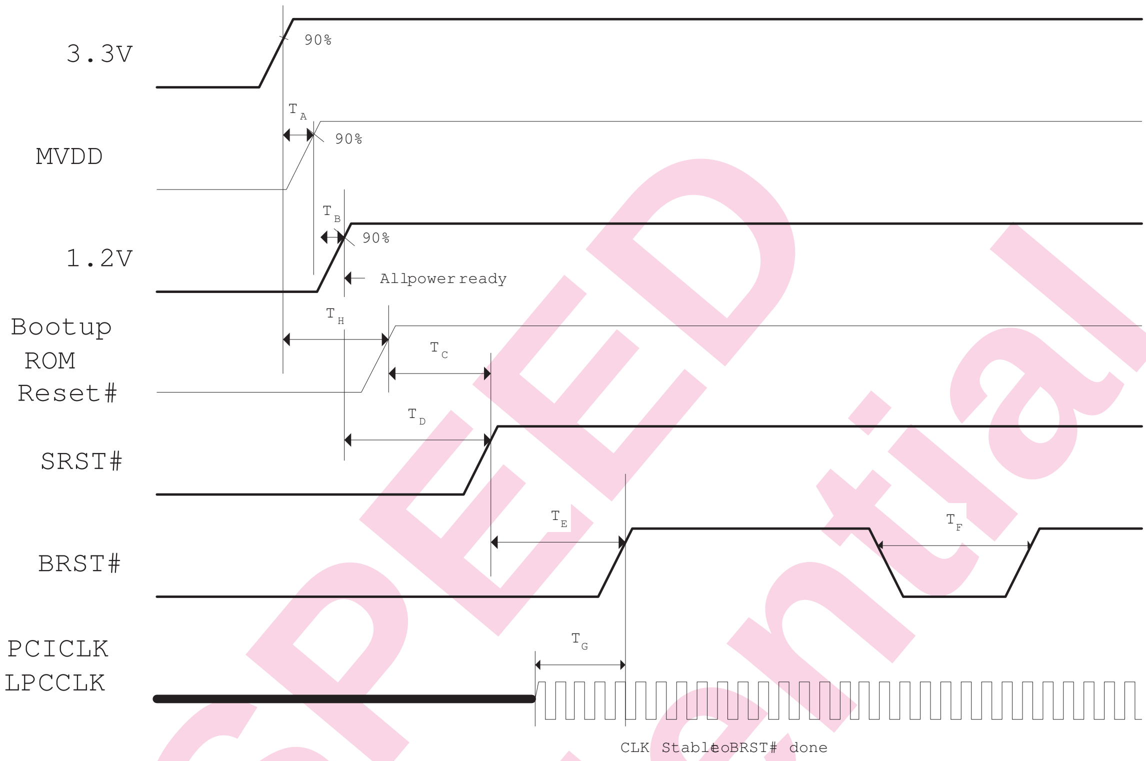


Figure 7: Power-up sequence

Power-up Sequence Parameters

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
3.3V to MVDD delay	T_A	0			us
MVDD to 1.2V delay	T_B	0			us
Flash memory reset to power-on reset delay	T_C	10			ms
Power stable to power-on reset finish, wait PLL ready	T_D	10			ms
Power-on Reset to PCI/LPC Reset finish, wait ID write by SOC OK	T_E	100			ms
Minimum valid Reset pulse width	T_F	1			us
Clock stable to Reset finish	T_G	1			us
3.3V to Flash memory reset delay	T_H	1			ms

There is another timing sequence must be care.
Software booting sequence : AST2050 / AST1100 firmware must boot finished before system BIOS booting. This is because system BIOS sometimes need to talk to AST2050 / AST1100 firmware, it is necessary for AST2050 / AST1100 firmware to boot-up finished first. This timing must be considered when doing system design.

4.6 Thermal Specification

4.6.1 Terminology

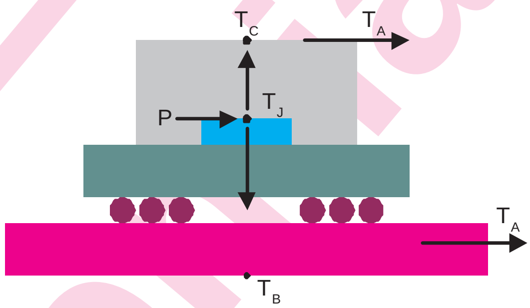
The major thermal dissipation paths can be illustrated as following:

- T_J : the maximum junction temperature
- T_A : the ambient or environment temperature
- T_C : the maximum compound surface temperature
- T_B : the maximum surface temperature of PCB bottom
- P : total input power

The thermal parameters can be defined as following figure:

1. Junction to ambient thermal resistance, $Theta_{JA}$

$$\theta_{JA} = \frac{T_J - T_A}{P}$$



Thermal Dissipation of PBGA Package

2. Junction to case thermal resistance, $Theta_{JC}$

$$\theta_{JC} = \frac{T_J - T_C}{P}$$

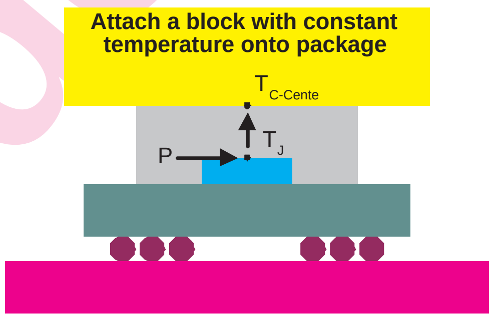


Figure 8: Thermal Terminology

4.6.2 Testing Conditions

Package Conditions	
Package Type	Thin Fine pitch Ball Grid Array Package
Ball Count	355
Package Dimension (L x W)	19 x 19 mm
Mold Thickness	1.2 mm (include solder ball)
Chip Size	5.8 x 5.6 x 0.254 mm
Die Attached Thickness	0.0254 mm
Solder Ball Number underneath Die	20
Via Number underneath Die	41
Ball Pitch	0.8 mm
Number of Cu Layer-Substrate	4 layers
PCB Conditions	
PCB Dimensions (L x W)	4 x 4.5 inches
PCB Thickness	1.6 mm
Number of Cu Layer-PCB	4 layers (2S2P)
Environment Conditions	
Maximum Junction temperature (C)	125
Maximum Ambient temperature (C)	70
Input Power (watt)	1
Control Condition	Air Flow = 0, 1, 2 m/s

4.6.3 Thermal Data

$\theta_{JA}(^oC/W)$			$\theta_{JC}(^oC/W)$
0 m/s	1 m/s	2 m/s	
23.6	19.9	18.9	6.2

4.6.4 Power Dissipation Capability

Air Flow (m/s)	0	1	2
Power Dissipation (watt)	2.3	2.8	2.9

4.7 AC Timing Specification

4.7.1 PCI Interface

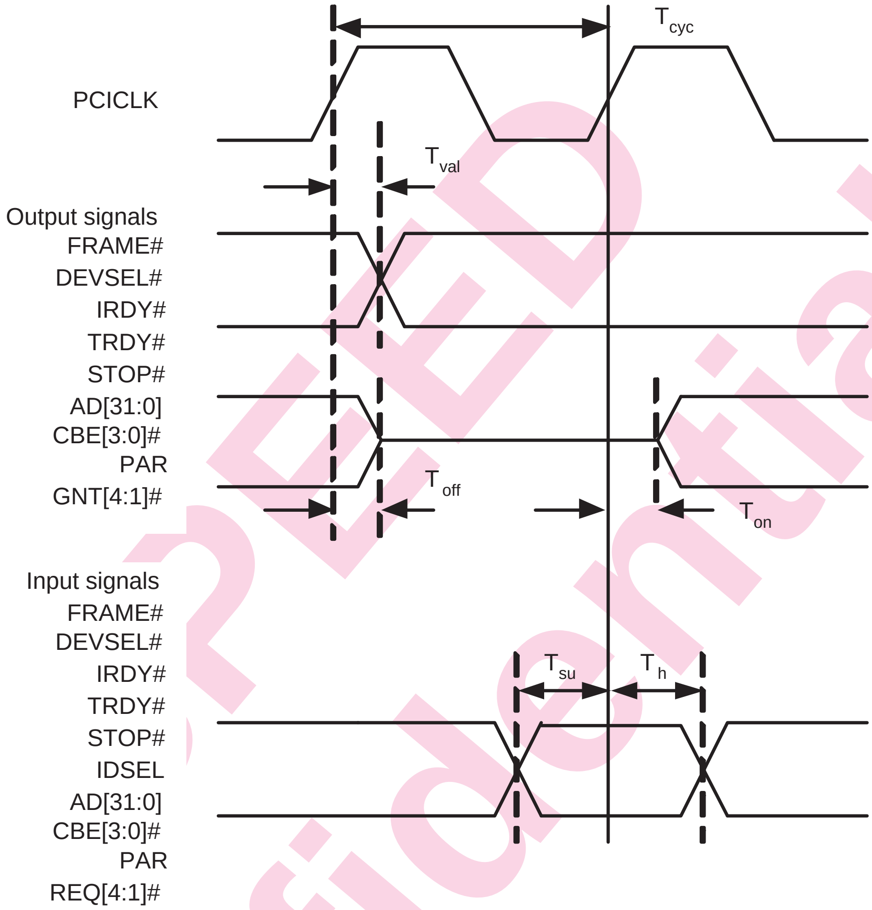


Figure 9: PCI Timing Waveform

PCI 33MHz

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Clock cycle time	T_{cyc}		30		ns
Output valid time	T_{val}	2		11	ns
Input setup time	T_{su}	7			ns
Input hold time	T_h	0			ns
Output float to active time	T_{on}	2			ns
Output active to float time	T_{off}			28	ns

4.7.2 MII/RMII Interface

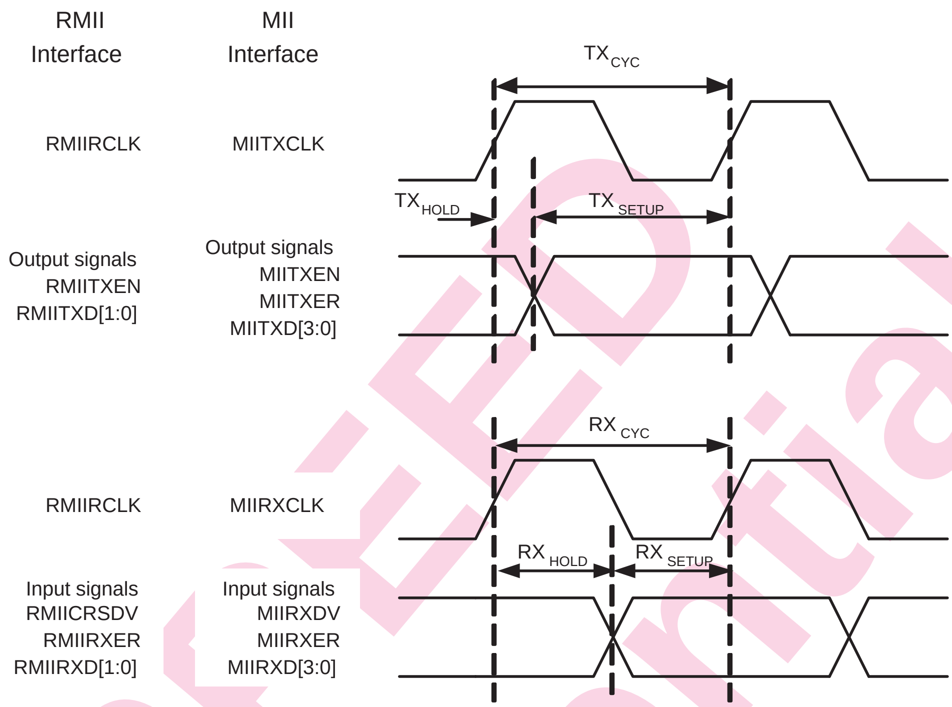


Figure 10: MII/RMII Timing Waveform

MII Timing

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Transmit Clock cycle time	TX_{CYC}		40		ns
Data output hold time	TX_{HOLD}	2.5			ns
Data output setup time	TX_{SETUP}	30			ns
Receive Clock cycle time	RX_{CYC}		40		ns
Data input hold time	RX_{HOLD}	0			ns
Data input setup time	RX_{SETUP}	5			ns

RMII Timing

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Transmit Clock cycle time	TX_{CYC}		20		ns
Data output hold time	TX_{HOLD}	2.5			ns
Data output setup time	TX_{SETUP}	10			ns
Receive Clock cycle time	RX_{CYC}		20		ns
Data input hold time	RX_{HOLD}	0			ns
Data input setup time	RX_{SETUP}	5			ns

4.7.3 DDR/DDR2 Interface

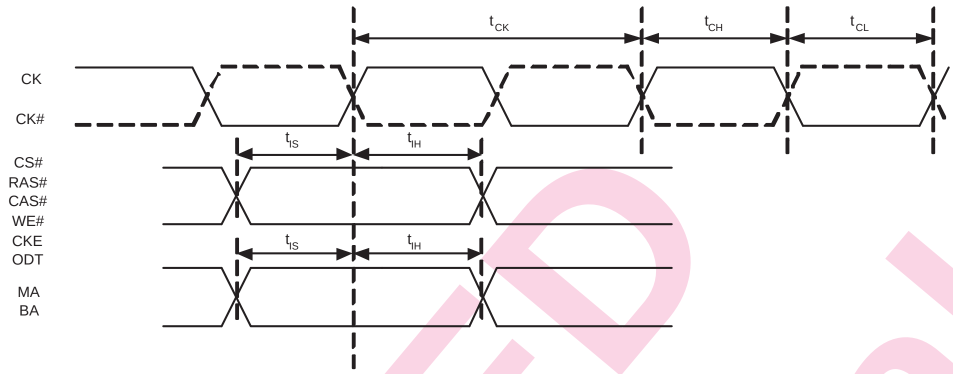


Figure 11: DDR/DDR2 Control Waveform

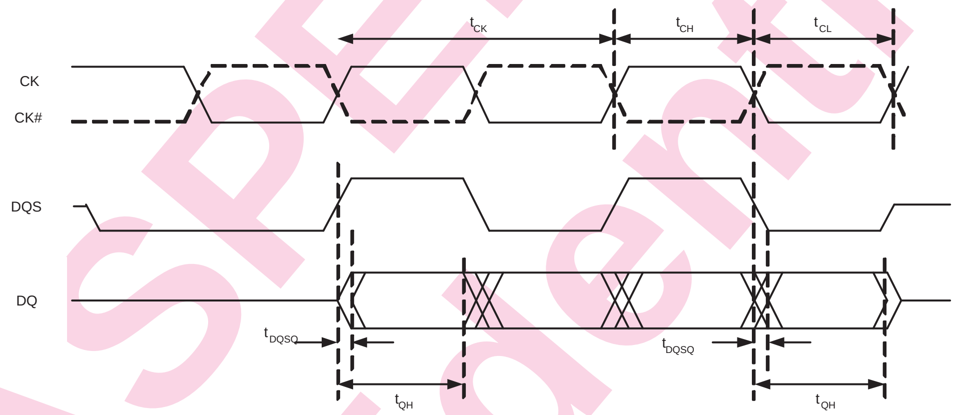


Figure 12: DDR/DDR2 Read Waveform

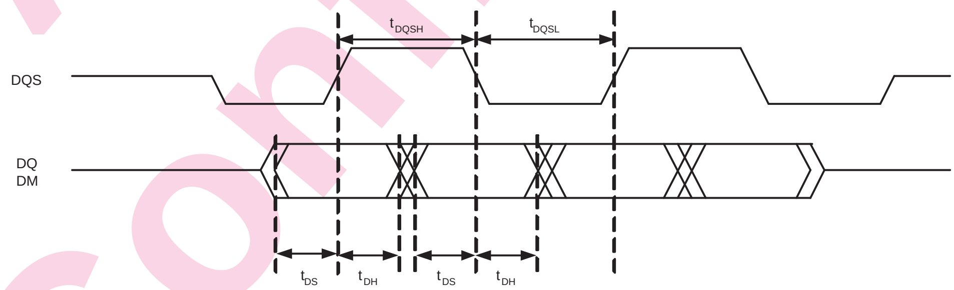


Figure 13: DDR/DDR2 Write Waveform

Embedded DLLs for CK/CK# output, DQS output and DQS input fine tune purpose. The DLL adjustment resolution is smaller than 1% of the memory clock cycle. All the timing related to the 3 signals can be fine tuned to get the best timing margin. Including t_{IS}, t_{IH}, t_{DS} and t_{DH}.

DDR333 Timing

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Clock cycle time	t_{CK}		6		ns
CK high level width	t_{CH}	0.45	0.5	0.55	tCK

to next page

from previous page

CL low level width	t_{CL}	0.45	0.5	0.55	tCK
Control input setup time	t_{IS}	0.8			ns
Control input hold time	t_{IH}	0.8			ns
DQ output hold time from DQS	t_{QH}	2.0			ns
DQS to DQ skew	t_{DQSQ}			0.45	ns
DQS high pulse width	t_{DQSH}	0.35			tCK
DQS low pulse width	t_{DQSL}	0.35			tCK
DQ,DQM setup time	t_{DS}	0.45			ns
DQ,DQM hold time	t_{DH}	0.45			ns

DDR400 Timing

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Clock cycle time	t_{CK}		5		ns
CK high level width	t_{CH}	0.45	0.5	0.55	tCK
CL low level width	t_{CL}	0.45	0.5	0.55	tCK
Control input setup time	t_{IS}	0.7			ns
Control input hold time	t_{IH}	0.7			ns
DQ output hold time from DQS	t_{QH}	1.75			ns
DQS to DQ skew	t_{DQSQ}			0.4	ns
DQS high pulse width	t_{DQSH}	0.35			tCK
DQS low pulse width	t_{DQSL}	0.35			tCK
DQ,DQM setup time	t_{DS}	0.4			ns
DQ,DQM hold time	t_{DH}	0.4			ns

DDR2-400 Timing

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Clock cycle time	t_{CK}		5		ns
CK high level width	t_{CH}	0.45	0.5	0.55	tCK
CL low level width	t_{CL}	0.45	0.5	0.55	tCK
Control input setup time	t_{IS}	0.6			ns
Control input hold time	t_{IH}	0.6			ns
DQ output hold time from DQS	t_{QH}	1.8			ns
DQS to DQ skew	t_{DQSQ}			0.35	ns
DQS high pulse width	t_{DQSH}	0.35			tCK
DQS low pulse width	t_{DQSL}	0.35			tCK
DQ,DQM setup time	t_{DS}	0.4			ns
DQ,DQM hold time	t_{DH}	0.4			ns

4.7.4 Video Interface: Single Edge

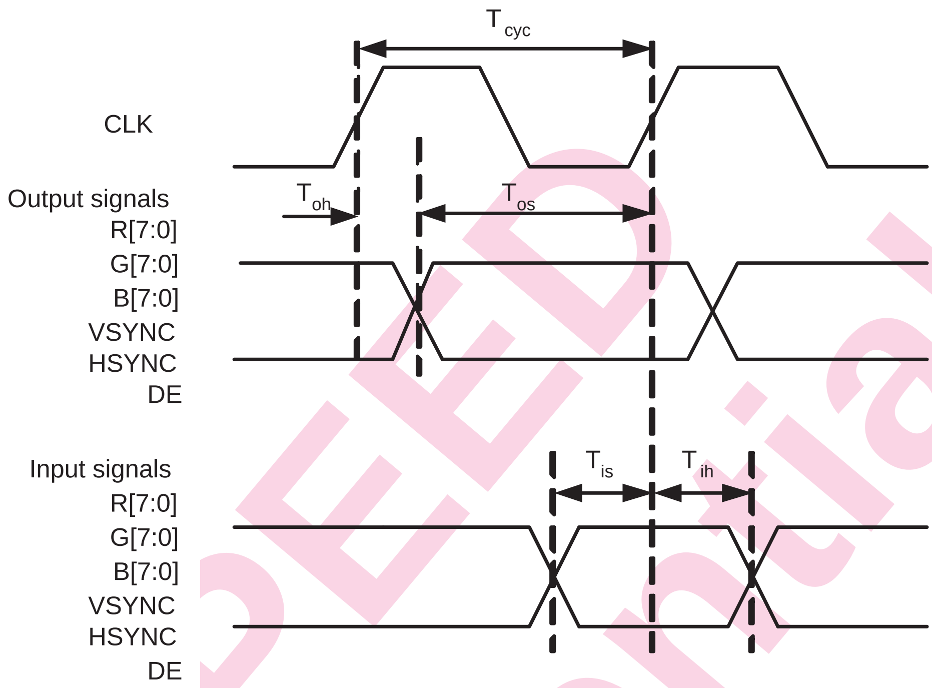


Figure 14: Video SDR Timing Waveform

Video SDR Timing

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Clock cycle time	T_{cyc}	6		40	ns
Data output setup time	T_{os}	1			ns
Data output hold time	T_{oh}	1			ns
Data input setup time	T_{is}	1			ns
Data input hold time	T_{ih}	1			ns

4.7.5 Video Interface: Dual Edge

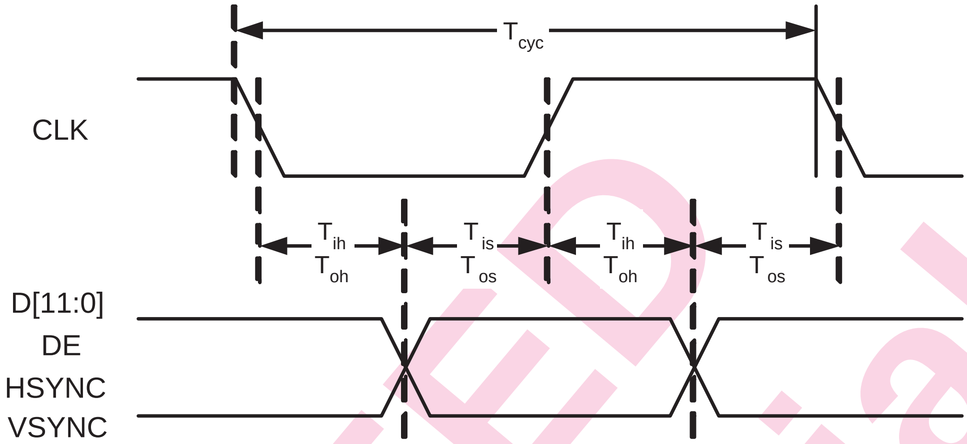


Figure 15: Video DDR Timing Waveform

Video DDR Timing

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Clock cycle time	T_{cyc}	6		40	ns
Data output setup time	T_{os}	1			ns
Data output hold time	T_{oh}	1			ns
Data input setup time	T_{is}	1			ns
Data input hold time	T_{ih}	1			ns

4.7.6 SPI Interface

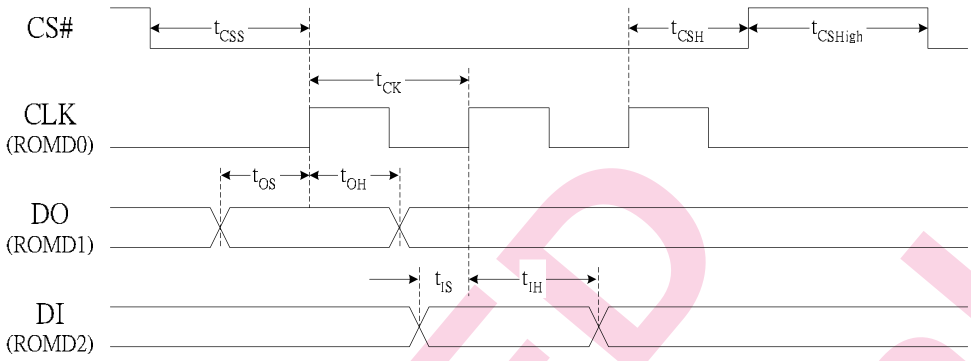


Figure 16: SPI Flash Timing Waveform

CPU = 200 MHz, AHB = 100 MHz (t_{AHB} = 10 ns)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
SPI Clock period (programmable)	t_{CH}	$2t_{AHB}$		$16t_{AHB}$	ns
CS# Setup time	t_{CSS}		$0.5t_{CK} + 2t_{AHB}$		ns
CS# Hold time	t_{CSH}	$0.5t_{CK} + t_{AHB}$			ns
CS# Inactive time (programmable)	t_{CSHigh}	$2t_{AHB}$		$18t_{AHB}$	ns
Data output to flash Setup time	t_{OS}		$0.5t_{CK}$		ns
Data output to flash Hold time	t_{OH}		$0.5t_{CK}$		ns
Data input from flash Setup time	t_{IS}	t_{AHB}			ns
Data input from flash Hold time	t_{IH}	$0.5t_{CK}$			ns

5 Package Information

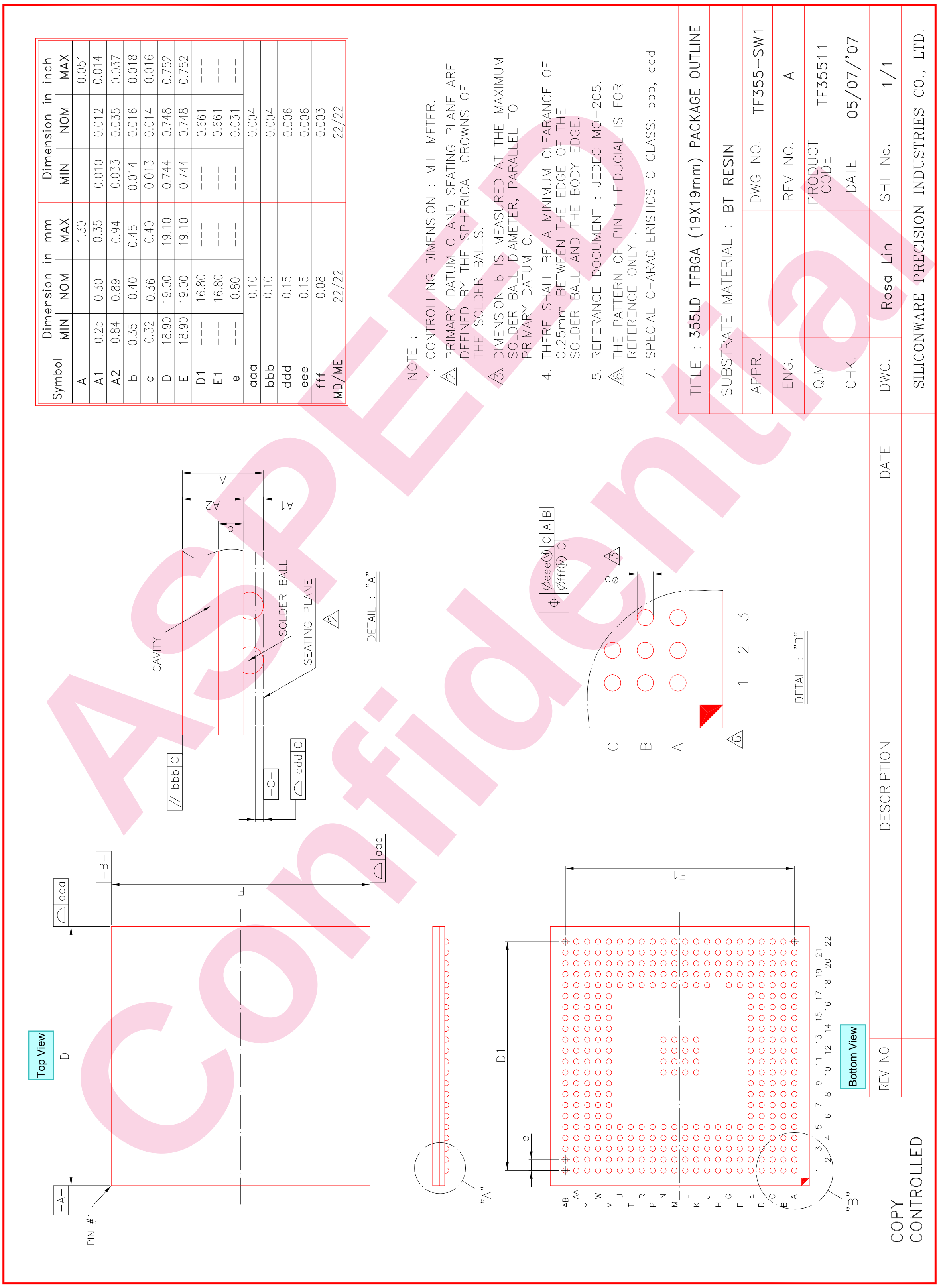


Figure 17: IC Package

6 XOR Tree

6.1 Brief Description

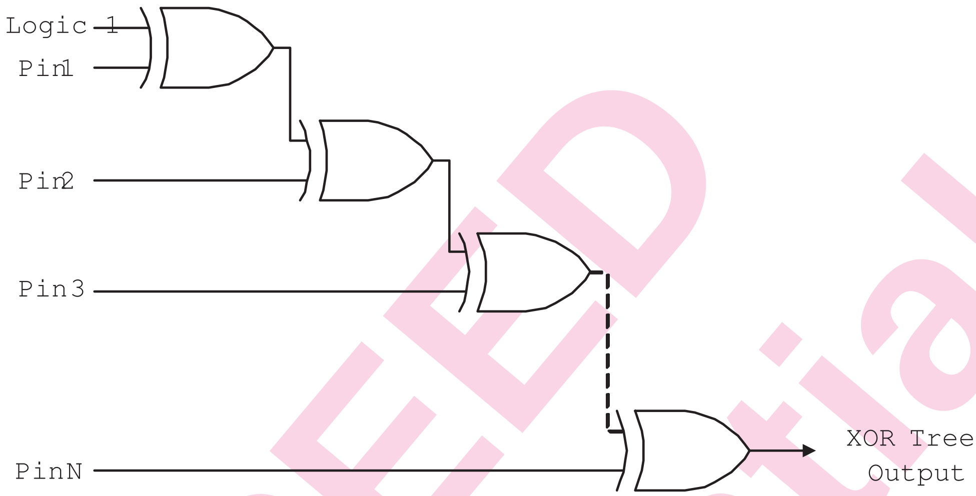


Figure 18: XOR Tree

The above picture is an example of XOR tree. Pin "XOR Tree Output" is an output pin and all of others are input pins. Start by driving all input pins with 0, and then shift input from pin 1 with value 1. The XOR tree output pin will be toggled as well as shift in operation is in progress. The XOR tree function can be shown as a truth table shown below.

XOR Tree Truth Table

Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	XOR Tree Output
0	0	0	0	0	0
1	0	0	0	0	1
1	1	0	0	0	0
1	1	1	0	0	1
1	1	1	1	0	0
1	1	1	1	1	1

6.2 XOR Tree Test Mode

AST2050 / AST1100 will enter XOR tree test mode by forcing ENTEST to logic '1', VPADE to logic '1' and VPAVSYNC to logic '0'. VPACLK and VPAHSYNC are the DDR IO input type selection. This setting can overwrite JTAG instructions. All of logic pins are force to input mode except XOR tree output pin. All analog pins and power pins are not included in XOR tree test function.

6.2.1 Force Pins of XOR Tree Test Mode

Force these pins can make AST2050 / AST1100 entering XOR tree test mode. The minimum setup time is 1us. After delaying 1us, AST2050 / AST1100 has been set to XOR tree test mode properly.

1	ENTEST	Logic 1
2	VPADE	Logic 1
3	VPAVSYNC	Logic 0

VPAHSYNC	VPACLK	DDR Combo IO Type
0	0	1.8V SSTL18
0	1	2.5V SSTL2
1	0	1.8V MDDR
1	1	3.3V LVTTL

6.2.2 XOR Tree Pin Order Table

No.	Signal	Ball	No.	Signal	Ball	No.	Signal	Ball
1	VP0	R3	72	WE#	W17	143	AD9	E20
2	VP1	R2	73	BA0	Y17	144	AD8	E21
3	VP2	R1	74	BA1	AA17	145	CBE0#	E22
4	VP3	T4	75	MA0	AB17	146	AD7	D19
5	VP4	T3	76	MA1	W18	147	AD6	D20
6	VP5	T2	77	MA2	Y18	148	AD5	D21
7	VP6	U4	78	MA3	AA18	149	AD4	D22
8	VP7	U3	79	CKE	AB18	150	AD3	C19
9	VP8	V4	80	MA4	W19	151	AD2	C20
10	VP9	V3	81	CK	AA19	152	AD1	C21
11	VP10	V2	82	MA5	Y19	153	AD0	C22
12	VP11	V1	83	MA6	W20	154	LAD0	B17
13	VP12	W4	84	MA7	Y20	155	LAD1	A17
14	VP13	W3	85	MA8	AA20	156	LAD2	D16
15	VP14	W2	86	MA9	AB20	157	LAD3	C16
16	VP15	W1	87	MA10	W21	158	LFRAME#	B16
17	VP16	Y4	88	MA11	Y21	159	LCLK	A16
18	VP17	Y3	89	MA12	AA21	160	LPCPD#	D15
19	ROMD0	Y2	90	ODT	AB21	161	LPCSIRQ	C15
20	ROMD1	Y1	91	RXD1	AA22	162	SCL1	B15
21	ROMD2	AA4	92	TXD1	Y22	163	SDA1	A15
22	ROMA16	AA3	93	NCTS1	W22	164	SCL2	D14
23	ROMA15	AA2	94	NDCD1	V19	165	SDA2	C14
24	ROMA14	AA1	95	NDSR1	V20	166	SCL3	B14
25	ROMA13	AB4	96	NRTS1	V21	167	SDA3	A14
26	ROMA12	AB3	97	NRI1	V22	168	SCL4	D13
27	ROMA11	AB2	98	NDTR1	U19	169	SDA4	C13
28	ROMA10	W5	99	RXD2	U20	170	SCL5	B13
29	ROMA9	Y5	100	TXD2	U21	171	SDA5	A13
30	ROMA8	AA5	101	TCK	U22	172	SCL6	D12
31	ROMA19	AB5	102	RTCK	T22	173	SDA6	C12
32	ROMA20	W6	103	TDI	T21	174	SCL7	B12
33	ROMA21	Y6	104	NTRST	T20	175	SDA7	A12
34	ROMA22	AA6	105	TMS	T19	176	GPIOA4	D11

to next page

from previous page

No.	Signal	Ball	No.	Signal	Ball	No.	Signal	Ball
35	ROMA23	AB6	106	TDO	R19	177	GPIOA5	C11
36	ROMCS2#	W7	107	SRST#	R20	178	INTA#	B11
37	ROMA18	Y7	108	CLKIN	R22	179	GPIOB1	A11
38	ROMA17	AA7	109	BRST#	P21	180	GPIOB2	D10
39	ROMA7	AB7	110	PCICLK	P22	181	GPIOB3	C10
40	ROMA6	W8	111	AD31	N19	182	GPIOB4	B10
41	ROMA5	Y8	112	AD30	N20	183	GPIOB5	A10
42	ROMA4	AA8	113	AD29	N21	184	GPIOB6	D9
43	ROMA3	AB8	114	AD28	N22	185	GPIOB7	C9
44	ROMA2	W9	115	AD27	M19	186	GPIOC0	B9
45	ROMA1	Y9	116	AD26	M20	187	GPIOC1	A9
46	ROMA0	AA9	117	AD25	M21	188	GPIOC2	D8
47	ROMCS0#	AB9	118	AD24	M22	189	GPIOC3	C8
48	DM0	W10	119	CBE3#	L19	190	GPIOC4	B8
49	DQ7	Y10	120	IDSEL	L20	191	GPIOC5	A8
50	DQS0	AA10	121	AD23	L21	192	MIITXCK	A7
51	DQ6	W11	122	AD22	L22	193	MIIRXCK	B7
52	DQ5	Y11	123	AD21	K19	194	MIIRXER	C7
53	DQ4	AA11	124	AD20	K20	195	MIIRXDV	D7
54	DQ3	AB11	125	AD19	K21	196	MIICOL	A6
55	DQ2	W12	126	AD18	K22	197	MIICRS	B6
56	DQ1	Y12	127	AD17	J19	198	MIIRXD0	C6
57	DQ0	AA12	128	AD16	J20	199	MIIRXD1	D6
58	DQ15	Y13	129	CBE2#	J21	200	MIIRXD2	A5
59	DQ14	AA13	130	FRAME#	J22	201	MIIRXD3	B5
60	DQ13	AB13	131	IRDY#	H19	202	MIITXEN	C5
61	DQ12	W14	132	TRDY#	H20	203	MIITXER	D5
62	DQ11	Y14	133	DEVSEL#	H21	204	MIITXD0	A4
63	DQ10	AA14	134	STOP#	H22	205	MIITXD1	B4
64	DQS1	AB14	135	PAR	G20	206	MIITXD2	C4
65	DQ9	AA15	136	CBE1#	G21	207	MIITXD3	D4
66	DQ8	Y15	137	AD15	G22	208	MIIMDC	A3
67	DM1	W15	138	AD14	F19	209	MIIMDIO	A2
68	CS#	W16	139	AD13	F20	210	DDCADAT	B2
69	BA2	Y16	140	AD12	F21	Out	DDCACLK	B1
70	RAS#	AA16	141	AD11	F22			
71	CAS#	AB16	142	AD10	E19			

Notice:

- XOR tree pin 48 to pin 90 are DDR IOs.
- Logic ’1’ of XOR tree pin 1 to 47 and pin 91 to 210 are 3.3V.

- Logic '1' of XOR tree output pin is 3.3V.
- Logic '0' of all XOR tree pins are 0V.
- For corresponding pin location, please reference Section 3.2.

6.2.3 List for Differential Pins

DQS0/DQS0#, DQS1/DQS1# and CK/CK# are DDR/DDR2 differential function pins. When doing XOR tree test, the input voltage for negative phase pins DQS0#(AB10), DQS1#(AB15) and CK#(AB19) must keep at MVDD/2 or the inverse level of positive phase pins.

6.2.4 List for Pins not in XOR Tree

All of these pins are analog signals. They did not be included in XOR tree test mode.

1	DACB	NC	7	DACVREFIN	NC
2	DACCOMP	NC	8	USBRPU	NC
3	DACG	NC	9	USBVRES	NC
4	DACR	NC	10	USB_DN	NC
5	DACRSET	NC	11	USB_DP	NC
6	DACVREF	NC	12	VREFSSTL	NC

Notice: NC means don't care.

All of these pins are powers. They did not be included in XOR tree test mode.

1	DACAV33	3.3V	24	MPLLAVSSG	GND
2	DACAV33B	3.3V	25	MPLLDV12	1.2V
3	DACAV33G	3.3V	26	MPLLDVSS	GND
4	DACAV33R	3.3V	27	MVDD	1.8V/2.5V
5	DACAVSS	GND	28	PLL1PO33	3.3V
6	DACAVSSB	GND	29	PLL1PV12	1.2V
7	DACAVSSG	GND	30	PLL2PO33	3.3V
8	DACAVSSR	GND	31	PLL2PV12	1.2V
9	DACDHV33	3.3V	32	PLL2PVSS	GND
10	DACDHVSS	GND	33	PV33D	3.3V
11	DACDV12	1.2V	34	USBV12L	1.2V
12	DACDVSS	GND	35	USBV33A1	3.3V
13	GND	GND	36	USBV33A2	3.3V
14	HPLLAV33	3.3V	37	USBV33D	3.3V
15	HPLLAV33G	3.3V	38	USBVSDL	GND
16	HPLLAVSS	GND	39	USBVSSA1	GND
17	HPLLAVSSG	GND	40	USBVSSA2	GND
18	HPLLDV12	1.2V	41	V1PLLAV12	1.2V
19	HPLLDVSS	GND	42	V1PLLAVSS	GND
20	IV12D	1.2V	43	V1PLLDV12	1.2V
21	MPLLAV33	3.3V	44	V1PLLDVSS	GND
22	MPLLAV33G	3.3V	45	VSSRSSTL	GND
23	MPLLAVSS	GND			

7 Multi-function Pins Mapping and Control

The following table defines the working function of all multi-function pins. The control priority is from "Function 1" (Highest) to "Function 4" (Lowest).

For the pins with 2 functions in it, it means an input mode.

Ball	Default	Function 1	Control 1	Function 2	Control 2	Function 3	Control 3	Function 4	Control 4
D11	GPIOA4	PHYLINK	SCU74[25]=1	GPIOA4	Others				
C11	GPIOA5	PHYPD#	SCU74[25]=1	GPIOA5	Others				
B11	INTA# (GPIOB0:IN)	INTA# (GPIOB0:IN)	SCU78[4]=0	GPIOB0	Others				
A11	GPIOB1	FLBUSY#	SCU74[2]=1	GPIOB1	Others				
D10	GPIOB2	FLWP#	SCU74[2]=1	GPIOB2	Others				
B10	N/A	VBCS	Trap[5]=1	LRST#	Trap[23]=1	GPIOB4	Others		
A10	N/A	VBCK	Trap[5]=1	GPIOB5	Others				
D9	N/A	VBDO	Trap[5]=1	WDTRST	SCU78[3]=1	GPIOB6	Others		
C9	N/A	VBDI	Trap[5]=1	EXTRST# (GPIOB7:IN)	GPIO04[15]=0	GPIOB7	Others		
B9	GPIOC0	PECII	SCU74[7]=1	GPIOC0	Others				
A9	GPIOC1	PECIO	SCU74[7]=1	GPIOC1	Others				
D8	GPIOC2	PWM1	SCU74[8]=1	GPIOC2	Others				
C8	GPIOC3	PWM2	SCU74[9]=1	GPIOC3	Others				
B8	GPIOC4	PWM3	SCU74[10]=1	GPIOC4	Others				
A8	GPIOC5	PWM4	SCU74[11]=1	GPIOC5	Others				
U2	HSYNC	HSYNC	SCU74[15]=1	VPAHSYNC	SCU74[16]=1	GPIOH4	Others		
R4	VSYNC	VSYNC	SCU74[15]=1	VPAVSYNC	SCU74[16]=1	GPIOH5	Others		
U1	GPIOH6	VPADE	SCU74[16]=1	GPIOH6	Others				
T1	GPIOH7	VPACLK	SCU74[16]=1	GPIOH7	Others				
B2	DDCADAT	DDCADAT	SCU74[18]=1	GPIOD6	Others				
B1	DDCACLK	OSCCLK	SCU2C[1]=1	DDCACLK	SCU74[18]=1	GPIOD7	Others		

to next page

from previous page

Ball	Default	Function 1	Control 1	Function 2	Control 2	Function 3	Control 3	Function 4	Control 4
A13	GPIOC6	MII2DIO	SCU74[20]=1	SDA5	SCU74[12]=1	GPIOC6	Others		
B13	GPIOC7	MII2DC	SCU74[20]=1	SCL5	SCU74[12]=1	GPIOC7	Others		
C12	GPIOH0	SDA6	SCU74[13]=1	GPIOH0	Others				
D12	GPIOH1	SCL6	SCU74[13]=1	GPIOH1	Others				
A12	GPIOH2	SDA7 (SALT2)	SCU74[14]=1	GPIOH2	Others				
B12	GPIOH3	SCL7 (SALT1)	SCU74[14]=1	GPIOH3	Others				
R3	GPIOE0	VP0	SCU74[22]=1	TACH0 (GPIOE0:IN)	GPIO24[0]=0	GPIOE0	Others		
R2	GPIOE1	VP1	SCU74[22]=1	TACH1 (GPIOE1:IN)	GPIO24[1]=0	GPIOE1	Others		
R1	GPIOE2	VP2	SCU74[22]=1	TACH2 (GPIOE2:IN)	GPIO24[2]=0	GPIOE2	Others		
T4	GPIOE3	VP3	SCU74[22]=1	TACH3 (GPIOE3:IN)	GPIO24[3]=0	GPIOE3	Others		
T3	GPIOE4	VP4	SCU74[22]=1	TACH4 (GPIOE4:IN)	GPIO24[4]=0	GPIOE4	Others		
T2	GPIOE5	VP5	SCU74[22]=1	TACH5 (GPIOE5:IN)	GPIO24[5]=0	GPIOE5	Others		
U4	GPIOE6	VP6	SCU74[22]=1	TACH6 (GPIOE6:IN)	GPIO24[6]=0	GPIOE6	Others		
U3	GPIOE7	VP7	SCU74[22]=1	TACH7 (GPIOE7:IN)	GPIO24[7]=0	GPIOE7	Others		
V4	GPIOF0	VP8	SCU74[22]=1	TACH8 (GPIOF0:IN)	GPIO24[8]=0	GPIOF0	Others		
V3	GPIOF1	VP9	SCU74[22]=1	TACH9 (GPIOF1:IN)	GPIO24[9]=0	GPIOF1	Others		

to next page

from previous page

Ball	Default	Function 1	Control 1	Function 2	Control 2	Function 3	Control 3	Function 4	Control 4
V2	GPIOF2	VP10	SCU74[22]=1	TACH10 (GPIOF2:IN)	GPIO24[10]=0	GPIOF2	Others		
V1	GPIOF3	VP11	SCU74[22]=1	TACH11 (GPIOF3:IN)	GPIO24[11]=0	GPIOF3	Others		
W4	GPIOF4	VP12	SCU74[23]=1	TACH12 (GPIOF4:IN)	GPIO24[12]=0	GPIOF4	Others		
W3	GPIOF5	VP13	SCU74[23]=1	TACH13 (GPIOF5:IN)	GPIO24[13]=0	GPIOF5	Others		
W2	GPIOF6	VP14	SCU74[23]=1	TACH14 (GPIOF6:IN)	GPIO24[14]=0	GPIOF6	Others		
W1	GPIOF7	VP15	SCU74[23]=1	TACH15 (GPIOF7:IN)	GPIO24[15]=0	GPIOF7	Others		
Y4	GPIOG0	VP16	SCU74[23]=1	GPIOG0	Others				
Y3	GPIOG1	VP17	SCU74[23]=1	GPIOG1	Others				
B6	N/A	MIICRS	Trap[8:6]=1,3	RMII2CRSDV	Trap[8:6]=6	GPIOE6	SCU74[27]=1		
A6	N/A	MIICOL	Trap[8:6]=1,3	RMII2RXER	Trap[8:6]=6	GPIOE7	SCU74[27]=1		
A4	N/A	MIITXD0	Trap[8:6]=1,3	RMIITXD0	Trap[8:6]=6,4				
B4	N/A	MIITXD1	Trap[8:6]=1,3	RMIITXD1	Trap[8:6]=6,4				
C4	N/A	MIITXD2	Trap[8:6]=1,3	RMII2TXD0	Trap[8:6]=6	GPIOE0	SCU74[27]=1		
D4	N/A	MIITXD3	Trap[8:6]=1,3	RMII2TXD1	Trap[8:6]=6	GPIOE1	SCU74[27]=1		
D5	N/A	MIITXER	Trap[8:6]=1,3	RMII2TXEN	Trap[8:6]=6	GPIOE4	SCU74[27]=1		
C5	N/A	MIITXEN	Trap[8:6]=1,3	RMIITXEN	Trap[8:6]=6,4				
C6	N/A	MIIRXD0	Trap[8:6]=1,3	RMIIRXD0	Trap[8:6]=6,4				
D6	N/A	MIIRXD1	Trap[8:6]=1,3	RMIIRXD1	Trap[8:6]=6,4				
A5	N/A	MIIRXD2	Trap[8:6]=1,3	RMII2RXD0	Trap[8:6]=6	GPIOE2	SCU74[27]=1		
B5	N/A	MIIRXD3	Trap[8:6]=1,3	RMII2RXD1	Trap[8:6]=6	GPIOE3	SCU74[27]=1		
C7	N/A	MIIRXER	Trap[8:6]=1,3	RMIIRXER	Trap[8:6]=6,4				
D7	N/A	MIIRXDV	Trap[8:6]=1,3	RMIICRSDV	Trap[8:6]=6,4				

to next page

from previous page

Ball	Default	Function 1	Control 1	Function 2	Control 2	Function 3	Control 3	Function 4	Control 4
A7	N/A	MIITXCK	Trap[8:6]=1,3	RMIIRCLK	Trap[8:6]=6,4				
B7	N/A	MIIRXCK	Trap[8:6]=1,3	RMII2RCLK	Trap[8:6]=6	GPIOE5	SCU74[27]=1		

Note: ":IN" denotes Input direction. For these pins working at input mode, so another input function also can be applied.

7.1 Multi-function Pins Selection Guide

The following descripts more detailed about the selection of single or group pins of multi-function pins.

- The following pin pairs are group selected.
  - PHYLINK and PHYPD#
  - FLBUSY# and FLWP#
  - VBCS, VBCK, VBDO and VBDI
  - PECII and PECIO
  - HSYNC and VSYNC
  - VPAHSYNC and VPAVSYNC
  - VPADE and VPACLK
  - DDCACLK and DDCADAT
  - MII2DC and MII2DIO
  - SCL5 and SDA5
  - SCL6 and SDA6
  - SCL7 and SDA7
  - VP[11:0]
  - VP[17:12]
  - MII function pins
  - RMII function pins
  - RMII2 function pins
  - GPIOE (group 1) vs. GPIOE (group 2) only 1 group can be selected as GPIO function at a time
  - GPIOE (group 2)
- The following pin pair groups can be defined individually.
  - (GPIOB4, LRST#) or (GPIOB6, WDTRST) or (GPIOB7, EXTRST#)
  - (GPIOC2, PWM1) or (GPIOC3, PWM2) or (GPIOC4, PWM3) or (GPIOC5, PWM4)
  - (GPIOD2, FLACK) or (GPIOD3, HCLKO) or (GPIOD5, VGASENSE)
  - (GPIOE0(1), TACH0) or (GPIOE1(1), TACH1) or (GPIOE2(1), TACH2) or (GPIOE3(1), TACH3) or (GPIOE4(1), TACH4) or (GPIOE5(1), TACH5) or (GPIOE6(1), TACH6) or (GPIOE7(1), TACH7)
  - (GPIOF0, TACH8) or (GPIOF1, TACH9) or (GPIOF2, TACH10) or (GPIOF3, TACH11) or (GPIOF4, TACH12) or (GPIOF5, TACH13) or (GPIOF6, TACH14) or (GPIOF7, TACH15)

8 Clock and Reset Tree Architecture

8.1 Clock Information

Clock Name	Frequency	Description
CPUCLK	200MHz (max)	Generated from a dedicated PLL (H-PLL)
HCLK	100MHz (max)	Generated from a dedicated PLL (H-PLL)
PCLK[1]	100MHz (max)	Generated from a dedicated PLL (H-PLL)
ECLK	200MHz (max)	Generated from a dedicated PLL (M-PLL or H-PLL)
MCLK	200MHz (max)	Generated from a dedicated PLL (M-PLL or H-PLL)
YCLK	200MHz (max)	From MCLK
GCLK	200MHz (max)	From MCLK
V1CLK	165MHz (max)	Generated from a dedicated PLL (V1-PLL) or from DVI clock input
DCLK	165MHz (max)	Generated from a dedicated PLL (V1-PLL)
BCLK	33MHz	From the external PCI bus clock
LCLK	33MHz	From the external LPC bus clock
USB2CLK	30MHz	Generated from USB2.0 PHY
CLK12M	12MHz	Divided from the external 24MHz clock source
CLK1M	1MHz	Divided from the external 24MHz clock source
CLK32K	32KHz	Divided from the external 24MHz clock source
PECICLK	2MHz (max)	Divided from the external 24MHz clock source
PWMCLK	24MHz (max)	Divided from the external 24MHz clock source
TACHCLK	6MHz (max)	Divided from the external 24MHz clock source
MIITXCK	25MHz	From the external Ethernet PHY
MIIRXCK	25MHz	From the external Ethernet PHY
MII2TXCK	25MHz	From the external Ethernet PHY
MII2RXCK	25MHz	From the external Ethernet PHY
RMIIRCLK	50MHz	From the external 50MHz clock source
RMII2RCLK	50MHz	From the external 50MHz clock source

[1]There is a limitation on the PCLK frequency allowed.
a. For rev. A1 chip, when LPC master APB2LPC bridge function is used: LCLK $\geq$ PCLK. Rev. A2 had fixed this issue.
b. When LPC slave channel #3 BT mode is used: PCLK $>$ 0.5*LCLK.

8.2 Clock and Reset Tree Mapping Table

Module	Reset Tree	Clock Tree					
AHB Controller	HRST_N	HCLK					
AHB to M-Bus Bridge	AHB_RST_N	HCLK	MCLK				
ARM CPU	HRST_N	CPUCLK	MCLK				
APB 1 Bridge	AHB_RST_N	HCLK	PCLK				
DRAM Controller	MMC_RST_N	PCLK	MCLK				
Crypto Engine	AES_RST_N	PCLK	MCLK	YCLK			
PCI to AHB Bridge	A2P_RST_N	HCLK	BCLK				
AHB to PCI Bridge	A2P_RST_N	HCLK	BCLK				
PCI Slave	PCI_RST_N	BCLK					
VGA	PCI_RST_N	BCLK	MCLK	DCLK			
VGA SPI BIOS	PCI_RST_N	BCLK					
2D Engine	G2D_RST_N	BCLK	MCLK	GCLK			
Video Engine	VCE_RST_N	MCLK	ECLK	HCLK	V1CLK	REFCLK	
MDMA	DMA_RST_N	HCLK	MCLK				
MAC1	MAC1_RST_N	HCLK	MIITXCK	MIIRXCK	RMIIRCLK	GMIICKIN	
MAC1 M-Bus Bridge	MAC1_RST_N	HCLK	MCLK				
MAC2	MAC2_RST_N	HCLK	MII2TXCK	MII2RXCK	RMII2RCLK		
MAC2 M-Bus Bridge	MAC2_RST_N	HCLK	MCLK				
USB2.0 Virtual Hub	UB2_RST_N	HCLK	MCLK	USB2CLK			
MIC	MIC_RST_N	HCLK	MCLK				
Interrupt Controller	HRST_N	HCLK					
Static Memory	HRST_N	HCLK					
APB2 Bridge	AHB_RST_N	HCLK	PCLK				
Timer	HRST_N	PCLK	CLK1M				
UART1	HRST_N	PCLK	UARTCLK				
UART2	HRST_N	PCLK	UARTCLK				
Watchdog	HRST_N	PCLK	CLK1M				
RTC	PWRSTNin	PCLK	CLK32K				
I2C	I2C_RST_N	PCLK	MCLK				
LPC	LPC_RST_N LPC_LRST_N	PCLK	LCLK	HCLK			
GPIO	HRST_N PWRSTNin	PCLK					
PWM	PWM_RST_N	PCLK	PWMCLK	PWMCLKM	PWMCLKN	TACHCLKM	TACHCLKN
PECI	PECI_RST_N	PCLK	PECICLK				

Figure 19: Clock and Reset Tree Mapping Table

8.3 Reset Tree Control Table

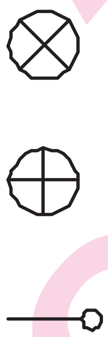
Module	Reset Tree	Global Reset			Register Control Reset : SCUR_04																
		SRST#	WDT	BRST#	DRAM	AHB	I2C	Crypto	LPC	Video	PCISlave	PWM	PECI	MAC1	MAC2	USB2.0	MDMA	MIC
AHB Controller	HRST_N	*	*															
AHB to M-Bus Bridge	AHB_RST_N	*	*			*												
ARM CPU	HRST_N	*	*															
APB 1 Bridge	AHB_RST_N	*	*			*												
DRAM Controller	MMC_RST_N	*			*													
Crypto Engine	AES_RST_N	*	*					*										
PCI to AHB Bridge	A2P_RST_N	*	*	*		*					*							
AHB to PCI Bridge	A2P_RST_N	*	*	*		*					*							
PCI Slave	PCI_RST_N			*							*							
VGA	PCI_RST_N			*							*							
VGA SPI BIOS	PCI_RST_N			*							*							
2D Engine	G2D_RST_N	*	*	*														
Video Engine	VCE_RST_N	*	*							*								
MDMA	DMA_RST_N	*	*														*	
MAC1	MAC1_RST_N	*	*											*				
MAC1 M-Bus Bridge	MAC1_RST_N	*	*											*				
MAC2	MAC2_RST_N	*	*												*			
MAC2 M-Bus Bridge	MAC2_RST_N	*	*												*			
USB2.0 Virtual Hub	UB2_RST_N	*	*													*		
MIC	MIC_RST_N	*	*															*
Interrupt Controller	HRST_N	*	*															
Static Memory	HRST_N	*	*															
APB2 Bridge	AHB_RST_N	*	*			*												
Timer	HRST_N	*	*															
UART1	HRST_N	*	*															
UART2	HRST_N	*	*															
Watchdog	HRST_N	*	*															
RTC	PWRSTNin	*																
I2C	I2C_RST_N	*	*				*											
LPC	LPC_RST_N	*	*						*									
	LPC_LRST_N			*														
GPIO	HRST_N	*	*															
	PWRSTNin	*																
PWM	PWM_RST_N	*										*						
PECI	PECI_RST_N	*	*										*					

Figure 20: Reset Tree Control Table

8.4 Symbol Description

Symbol	Description
CLKMASK	Clock gating off control
DEL2	Delay line of about 2ns
SCUxx	SCU register at offset xx
MCRxx	Memory controller register at offset xx
PECIxx	PECI controller register at offset xx
PTCRxx	PWM & Fan Tacho controller register at offset xx
VRxxx	Video register at offset xxx
VGACRxx	VGA CRT control register at offset xx
LHCRxx	LPC host controller register at offset xx
HWTrap	Hardware Trapping setting, equals SCU70
bootup_low_speed	Hardware Trapping bit[16]
dftmode	Mode when pin ENTEST = 1
bypass_clock	Mode when pin ENTEST = 1 or Hardware Trapping bit[22] = 1
hpll_freq_sel	Hardware Trapping bit[11:9]
hclk_ratio	Hardware Trapping bit[13:12]
pci_host_mode	Hardware Trapping bit[4]
UB11_StopClk	Stop clock control from USB1.1 controller
2D_Idle	2D Engine Idle
&	Logical AND operation
|	Logical OR operation
!	Logical NOT operation

Operation Definition



Conditional AND
Output is TRUE only when all input statements are TRUE

Conditional OR, summation
Output is TRUE when either one input statements are TRUE

Logical NOT

Figure 21: Tree Operation Symbol

8.5 Clock Tree Architecture

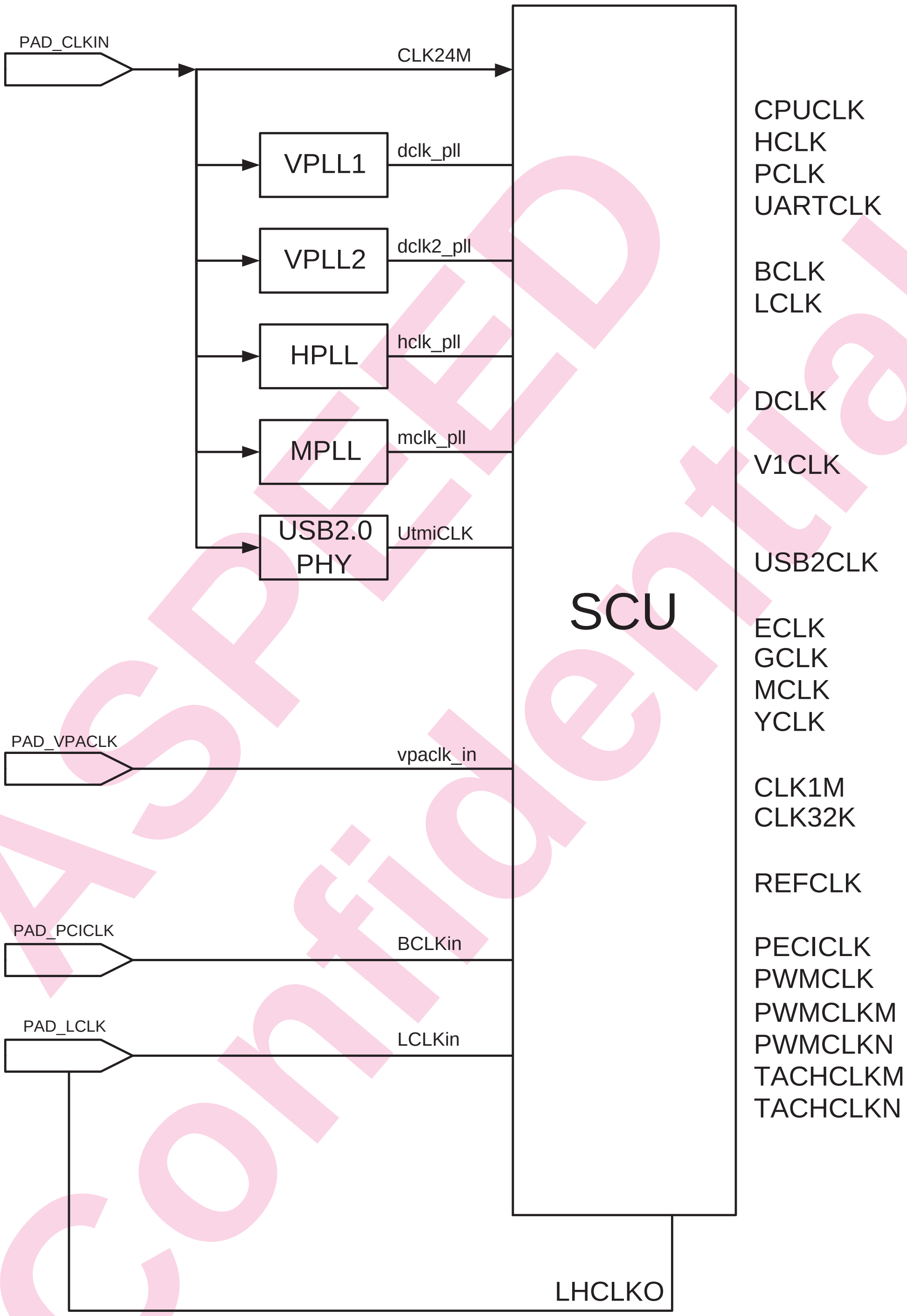


Figure 22: Clock Tree Global View

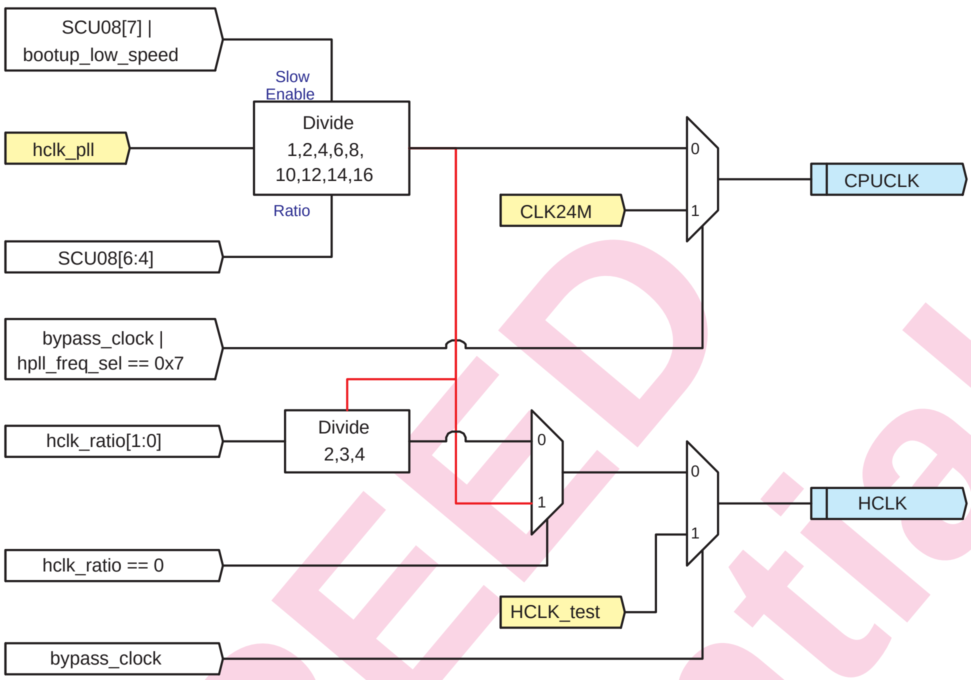


Figure 23: CPU & AHB Clock

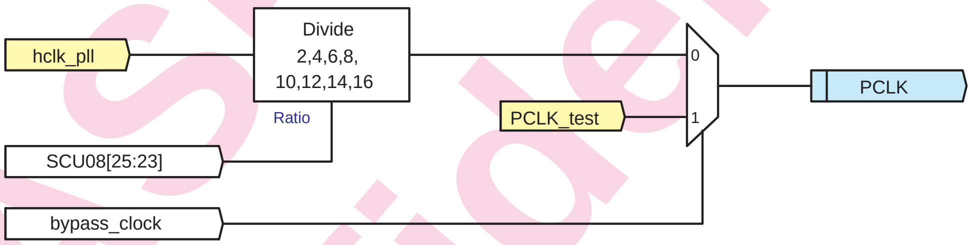


Figure 24: APB Clock

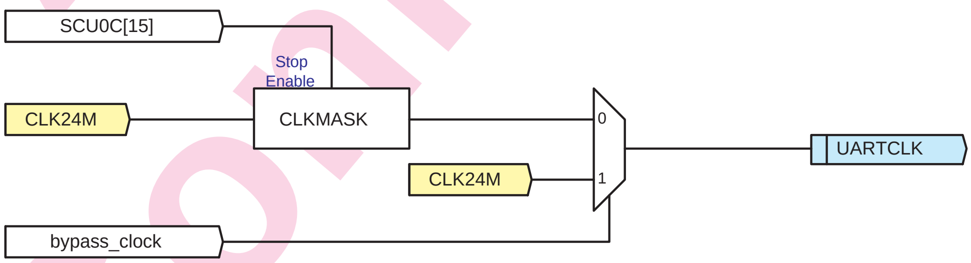


Figure 25: UART Clock

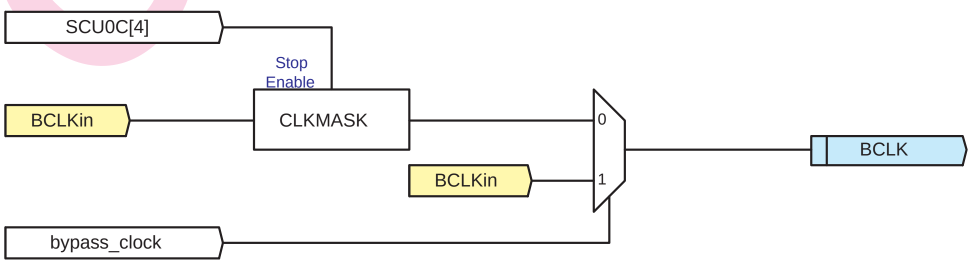


Figure 26: PCI Slave Clock

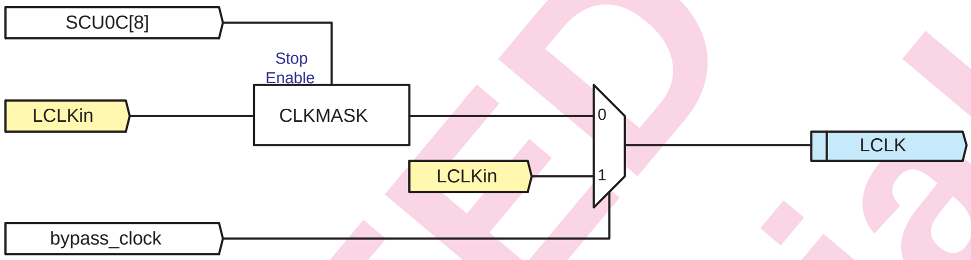


Figure 27: LPC Clock

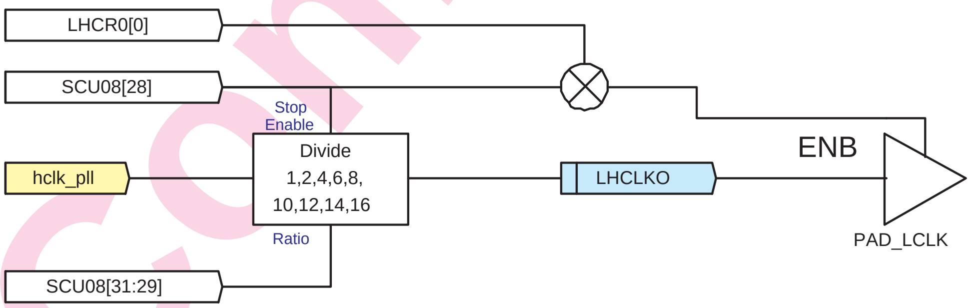


Figure 28: LPC Host Clock Output

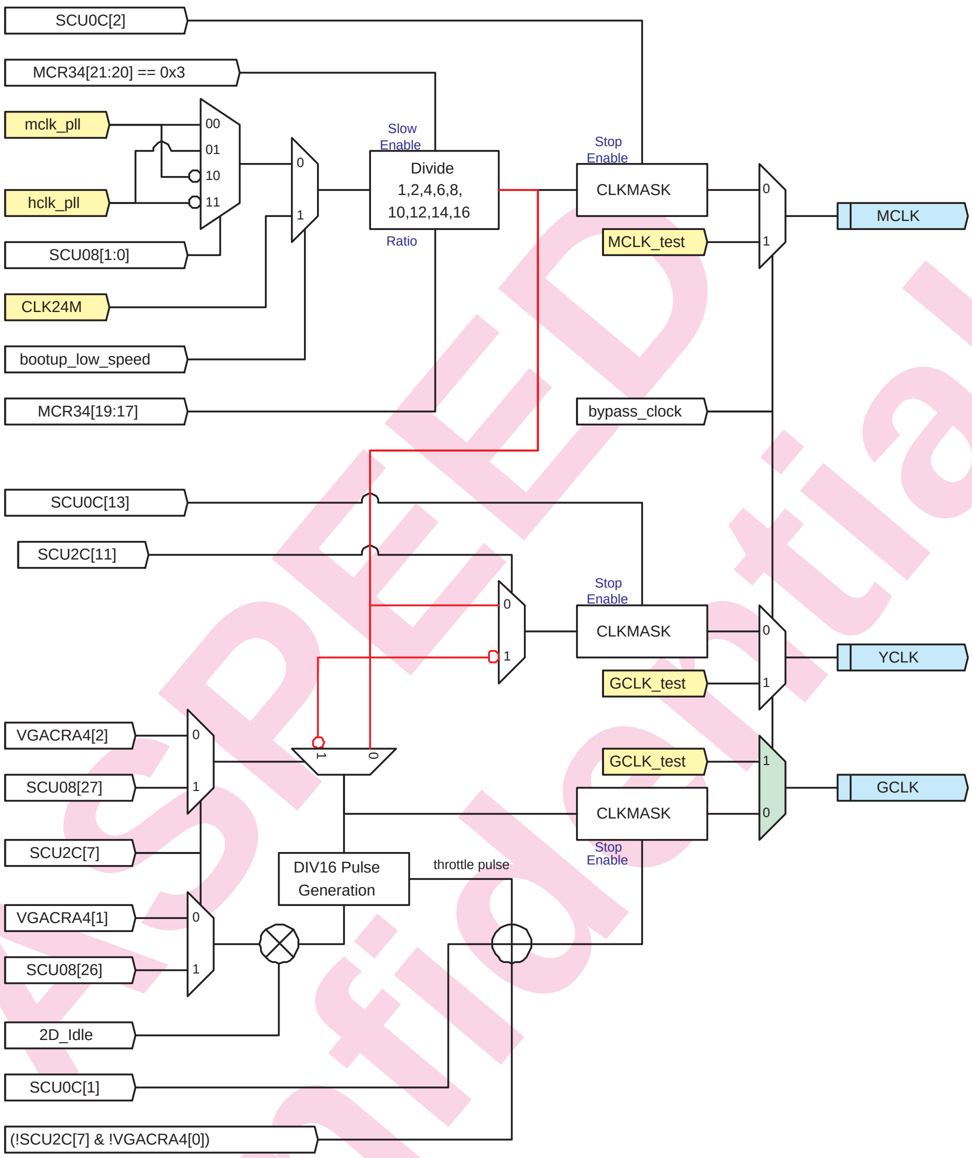


Figure 29: Memory Clock Group

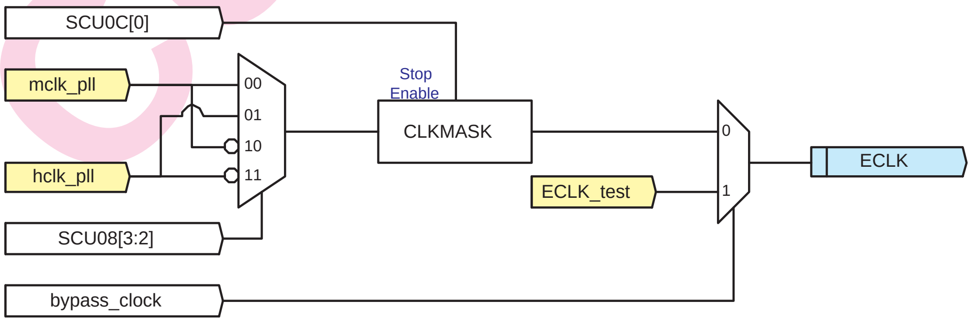


Figure 30: Video Engine Clock

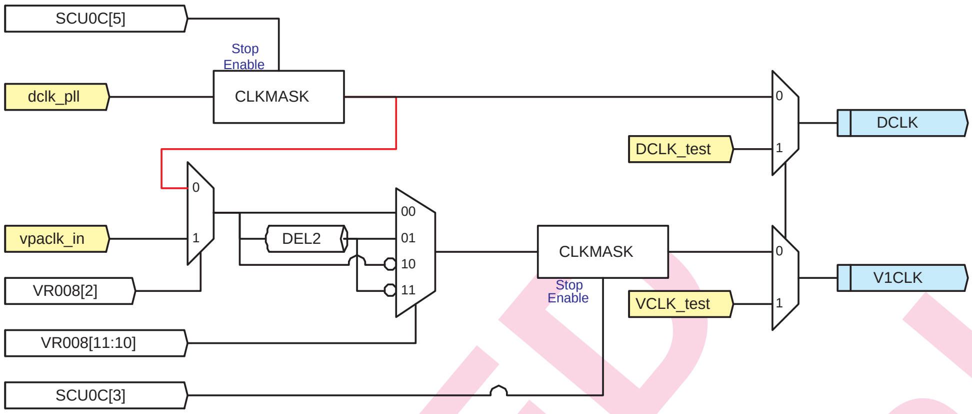


Figure 31: Video Clock

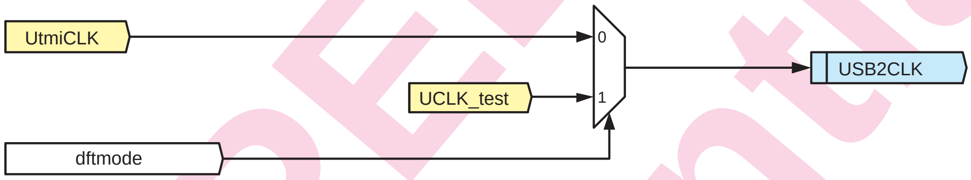


Figure 32: USB2.0 CLock

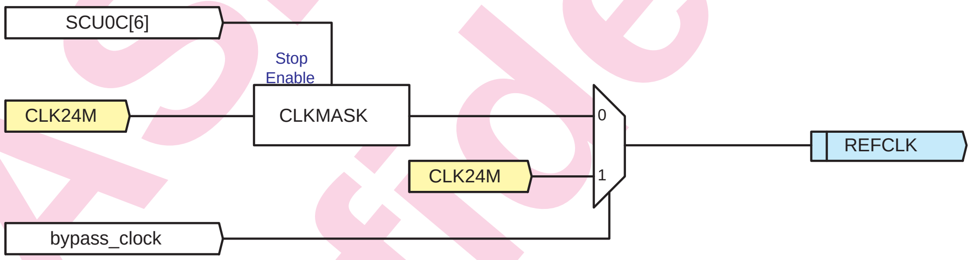


Figure 33: Reference Clock

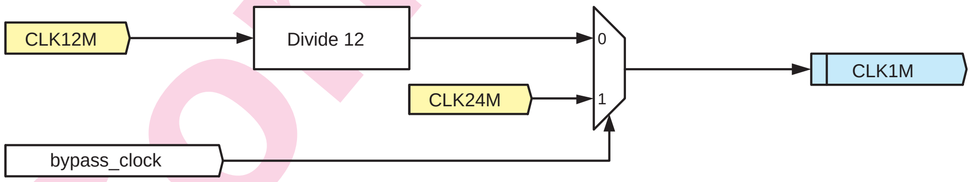


Figure 34: 1 MHz Clock

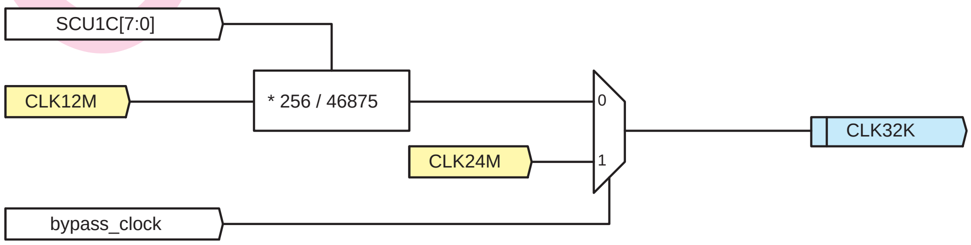


Figure 35: 32.768 KHz Clock

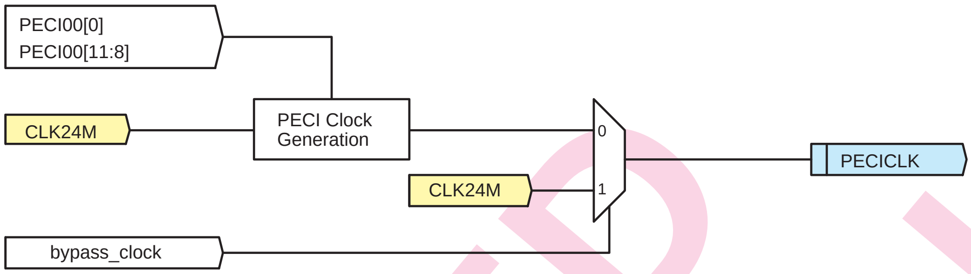


Figure 36: PECI Clock

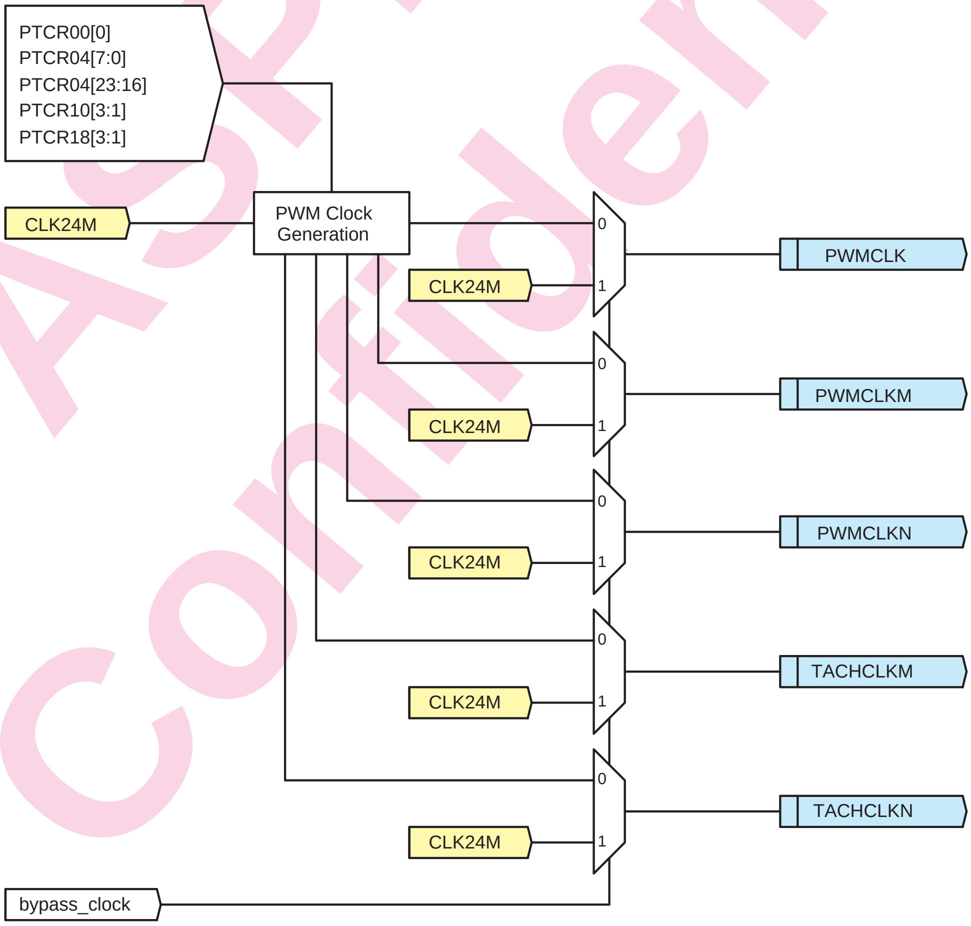


Figure 37: PWM Clock

8.6 Reset Tree Architecture

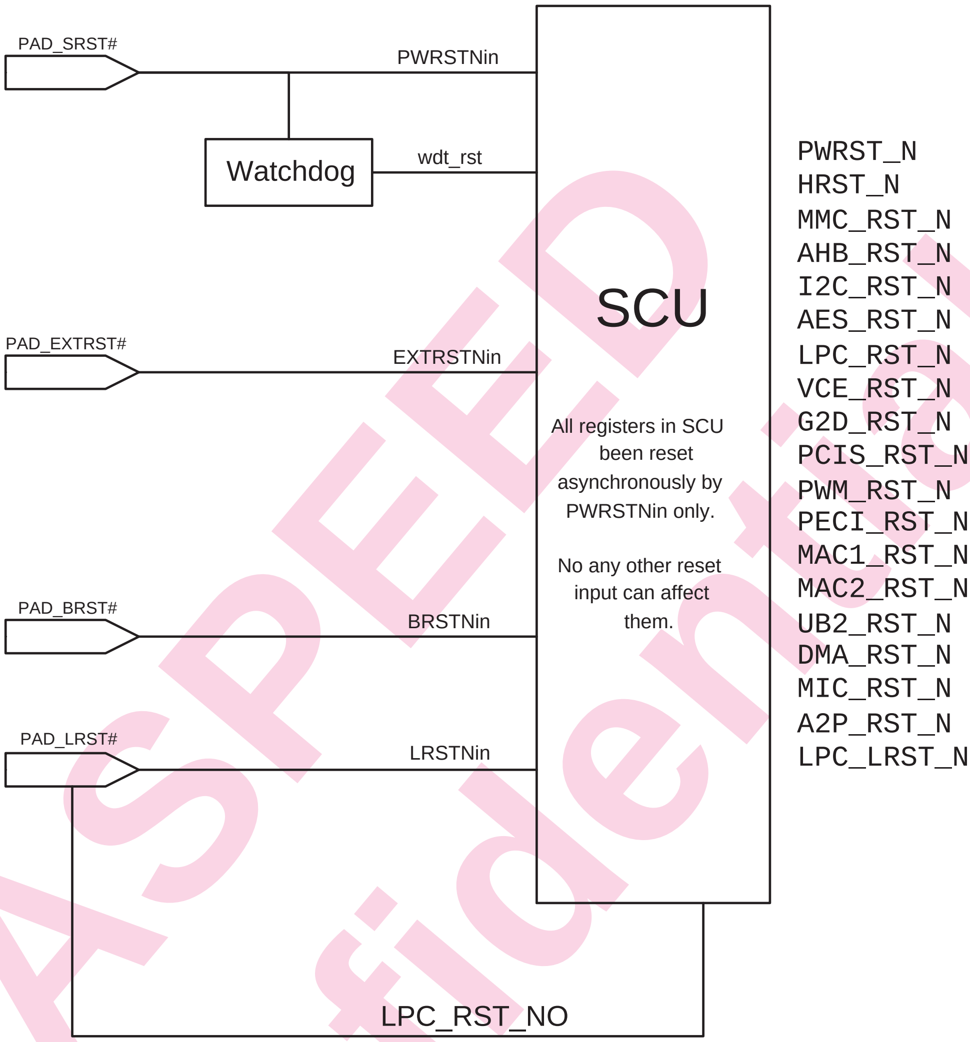


Figure 38: Reset Tree Global View

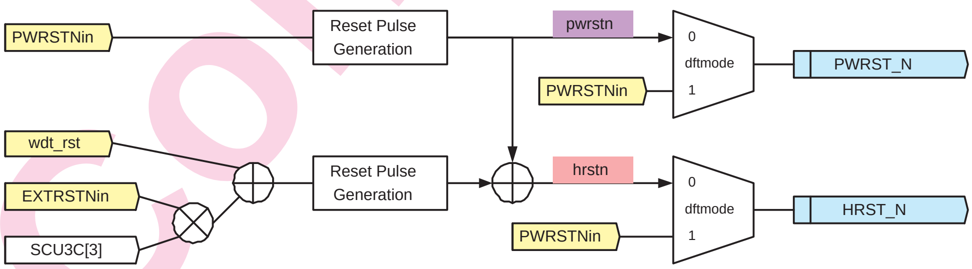


Figure 39: AHB Bus Reset

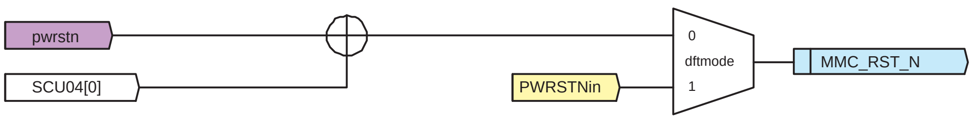


Figure 40: Memory Controller Reset

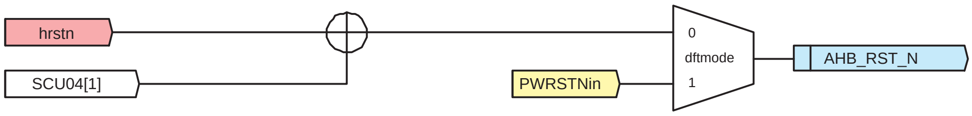


Figure 41: AHB Bus Bridge Reset

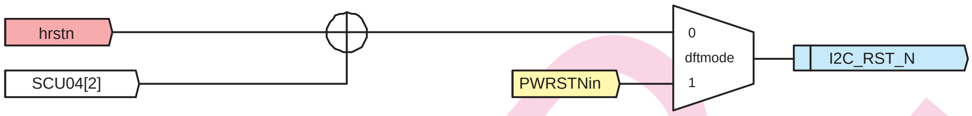


Figure 42: I2C Controller Reset

Figure 43: Crypto Engine Reset

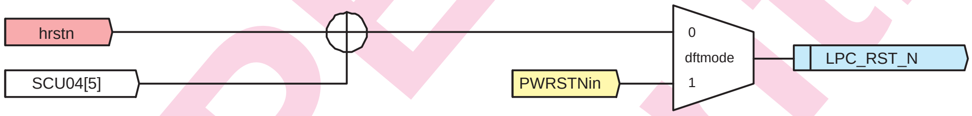


Figure 44: LPC Controller Reset

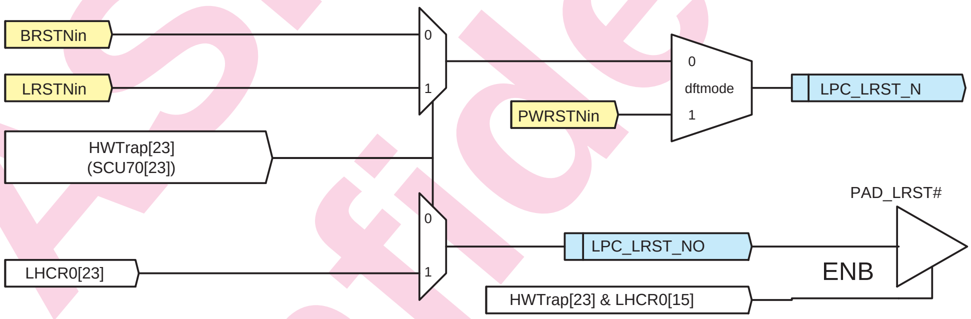


Figure 45: LPC Controller External Reset

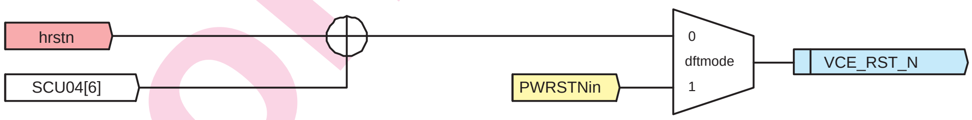


Figure 46: Video Engine Reset

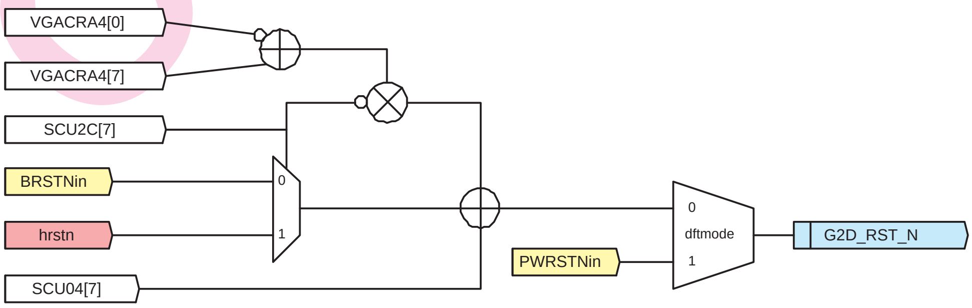


Figure 47: 2D Engine Reset

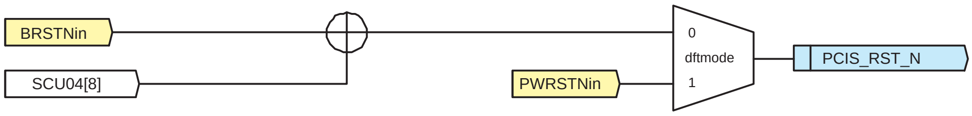


Figure 48: PCI Slave Reset

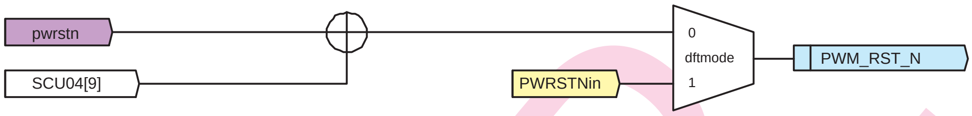


Figure 49: PWM Controller Reset

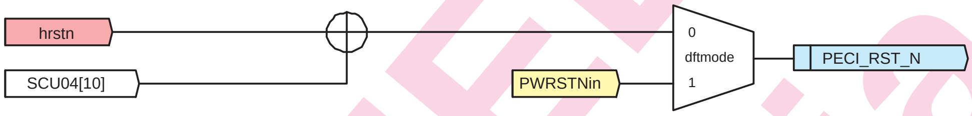


Figure 50: PECI Controller Reset

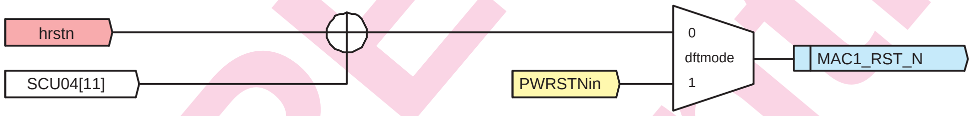


Figure 51: MAC1 Reset

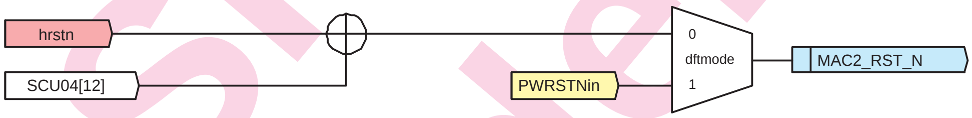


Figure 52: MAC2 Reset

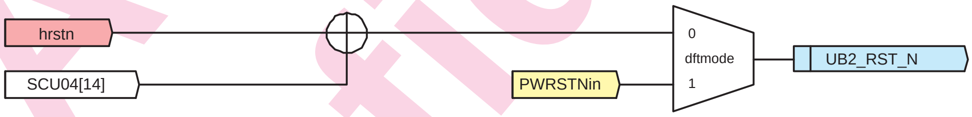


Figure 53: USB2.0 Controller Reset

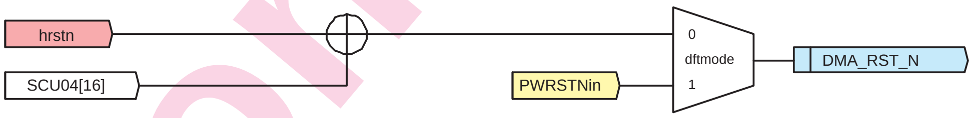


Figure 54: MDMA Engine Reset

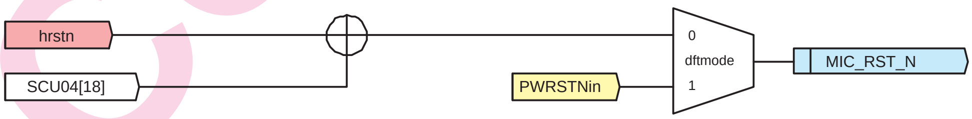


Figure 55: MIC Engine Reset

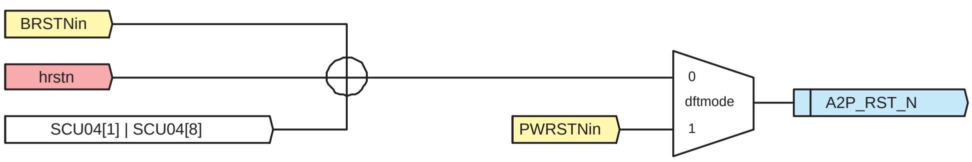


Figure 56: AHB to P-Bus Bridge Reset

Part II

ARM Interface

9 ARM Address Space Mapping

Address Range	Size (Byte)	Write Mode (Byte)	Read Mode (Byte)	IP Module
0000:0000-01FF:FFFF	32M	1/2/4	1/2/4	Static Memory (boot-up default)
0000:0000-0FFF:FFFF	256M	1/2/4	1/2/4	SDRAM (After Re-map)
1000:0000-15FF:FFFF	96M	1/2/4	1/2/4	Static Memory
1600:0000-17FF:FFFF	32M	4	1/2/4	Static Memory Controller (SMC)
1E60:0000-1E61:FFFF	128K	1/2/4	1/2/4	AHB Bus Controller (AHBC)
1E64:0000-1E65:FFFF	128K	4	1/2/4	Memory Integrity Check Controller (MIC)
1E66:0000-1E67:FFFF	128K	1/2/4	1/2/4	Fast Ethernet MAC Controller #1 (MAC1)
1E68:0000-1E69:FFFF	128K	1/2/4	1/2/4	Fast Ethernet MAC Controller #2 (MAC2)
1E6A:0000-1E6B:FFFF	128K	4	1/2/4	USB2.0 Controller
1E6C:0000-1E6D:FFFF	128K	1/2/4	1/2/4	Vector Interrupt Controller (VIC)
1E6E:0000-1E6E:0FFF	4K	4	1/2/4	SDRAM Controller (MMC)
1E6E:2000-1E6E:2FFF	4K	4	1/2/4	System Control Unit (SCU)
1E6E:3000-1E6E:3FFF	4K	4	1/2/4	Hash & Crypto Engine (HACE)
1E70:0000-1E71:FFFF	128K	1/2/4	1/2/4	Video Engine
1E72:0000-1E73:FFFF	128K	1/2/4	1/2/4	AHB to PCI (P-Bus) Bridge Controller (A2P)
1E74:0000-1E75:FFFF	128K	4	1/2/4	MDMA Controller
1E78:0000-1E78:0FFF	4K	4	1/2/4	GPIO Controller
1E78:1000-1E78:1FFF	4K	4	1/2/4	Real-Time Clock (RTC)
1E78:2000-1E78:2FFF	4K	4	1/2/4	Timer #1, #2, #3 Controller
1E78:3000-1E78:3FFF	4K	4	1/2/4	UART - #1
1E78:4000-1E78:4FFF	4K	4	1/2/4	UART - #2
1E78:5000-1E78:5FFF	4K	4	1/2/4	Watchdog Timer (WDT)
1E78:6000-1E78:6FFF	4K	4	1/2/4	PWM & Fan Tacho Controller
1E78:7000-1E78:7FFF	4K	4	1/2/4	Virtual UART
1E78:8000-1E78:8FFF	4K	4	1/2/4	Pass-through UART
1E78:9000-1E78:9FFF	4K	4	1/2/4	LPC Controller
1E78:A000-1E78:AFFF	4K	4	1/2/4	I2C/SMBus Controller
1E78:B000-1E78:BFFF	4K	4	1/2/4	PECI Controller
1E78:C000-1E78:CFFF	4K	4	1/2/4	PCI Arbiter
4000:0000-4FFF:FFFF	256M	1/2/4	1/2/4	SDRAM
5000:0000-5FFF:FFFF	256M	1/2/4	1/2/4	AHB Bus to LPC Bus Bridge
6000:0000-7FFF:FFFF	512M	1/2/4	1/2/4	PCI Host Memory #1
8000:0000-FFFF:FFFF	2G	1/2/4	1/2/4	PCI Host Memory #2

Note: **Program access the IP using un-supported access mode will get an un-predictable result.**

VGA Memory Space map to ARM Memory Space

The memory space is defined by the Hardware Trapping bit[3:2] at Page 217.

VGA Size	DRAM Size				
	16MB	32MB	64MB	128MB	256MB
8MB	4080:0000 ≀ 40FF:FFFF	4180:0000 ≀ 41FF:FFFF	4380:0000 ≀ 43FF:FFFF	4780:0000 ≀ 47FF:FFFF	4F80:0000 ≀ 4FFF:FFFF
16MB	–	4100:0000 ≀ 41FF:FFFF	4300:0000 ≀ 43FF:FFFF	4700:0000 ≀ 47FF:FFFF	4F00:0000 ≀ 4FFF:FFFF
32MB	–	–	4200:0000 ≀ 43FF:FFFF	4600:0000 ≀ 47FF:FFFF	4E00:0000 ≀ 4FFF:FFFF
64MB	–	–	–	4400:0000 ≀ 47FF:FFFF	4C00:0000 ≀ 4FFF:FFFF

10 Interrupt Source Table

INT#	Description	Attribute
0	Reserved	Reserved
1	MIC interrupt	Sensitive high level trigger
2	MAC1 interrupt	Sensitive high level trigger
3	MAC2 interrupt	Sensitive high level trigger
4	Crypto interrupt	Sensitive high level trigger
5	USB 2.0 interrupt	Sensitive high level trigger
6	MDMA interrupt	Sensitive high level trigger
7	Video Engine interrupt	Sensitive high level trigger
8	LPC interrupt	Sensitive high level trigger
9	UART1 alarm interrupt	Sensitive high level trigger
10	UART2 alarm interrupt	Sensitive high level trigger
11	Reserved	Reserved
12	I2C/SMBus interrupt	Sensitive high level trigger
13	Reserved	Reserved
14	Reserved	Reserved
15	PECI interrupt	Sensitive high level trigger
16	1'st counter interrupt of Timer	Rising-edge trigger
17	2'nd counter interrupt of Timer	Rising-edge trigger
18	3'rd counter interrupt of Timer	Rising-edge trigger
19	SMC interrupt	Sensitive high level trigger
20	GPIO interrupt	Sensitive high level trigger
21	SCU interrupt	Sensitive high level trigger
22	RTC second interrupt	Edge trigger and both edge
23	RTC day interrupt	Edge trigger and both edge
24	RTC hour interrupt	Edge trigger and both edge
25	RTC minute interrupt	Edge trigger and both edge
26	RTC alarm interrupt	Edge trigger and both edge
27	WDT alarm interrupt	Rising-edge trigger
28	Tachometer interrupt	Sensitive high level trigger
29	Reserved	Reserved
30	Reserved	Reserved
31	AHBC interrupt	Sensitive high level trigger

Table 36: Interrupt Source Table

11 Static Memory Controller

11.1 Overview

Static Memory Controller (SMC) implements 8 sets of 32-bit registers, which is listed below, to program the various static memory interfaces supported by AST2050 / AST1100 . Each register has its own specific offset value to derive its physical address location.

This is a superset of registers definition. For AST2050/AST1100 chip, only SPI flash type interface is supported.

Base address of SMC = 0x1600_0000
Physical address = (Base address of SMC) + Offset

SMC00: CE0 Segment AC Timing Register
SMC04: CE0 Control Register
SMC08: CE1 Control Register
SMC0C: CE2 Control Register
SMC10: Misc. Control Register
SMC14: NAND ECC Generation Control/Status register
SMC18: NAND ECC check value
SMC1C: NAND ECC check result

AST2050 / AST1100 supports three types of flash memory: NOR flash, NAND flash and SPI flash memory. Additionally, AST2050 / AST1100 also provides three chip select pins (CE0, CE1 and CE2) to control at most three flash memory devices, each of which can also be programmed to be any one of the three flash memory types. Moreover, each chip select pin is assigned to different non-overlapping address regions. The base addresses of the three chip select pins are as the followings:

Base address of CE0: 0x10000000
Base address of CE1: 0x10000000 + (Segment Size)
Base address of CE2: 0x10000000 + (Segment Size x 2)

Where, Segment Size is determined by SMC00 Bit [1:0]. Figure 57 shows the organization. Theoretically, all the three flash memory types can be working together simultaneously.
Only one of the three chip select pin can be assigned, by external trapping resisters, to support CPU boot code fetches (starting address 0x00000000). When selected, the addressing space of the assigned chip select pin will additionally include CPU boot code addressing space as well. The default flash memory type for each chip select pin is as the followings:

Default flash memory type of CE0: NOR flash
Default flash memory type of CE1: NAND flash
Default flash memory type of CE2: SPI flash

Therefore, NOR flash code boot has to connect on CE0, NAND flash code boot has to connect on CE1, and SPI flash code boot has to connect on CE2.
NOR flash memory type is for typical applications. NAND flash memory type is for applications with a large code size, but its software code needs to handle the potential bad block issue. SPI flash memory type is for applications with a small code size but critical footprint budget.

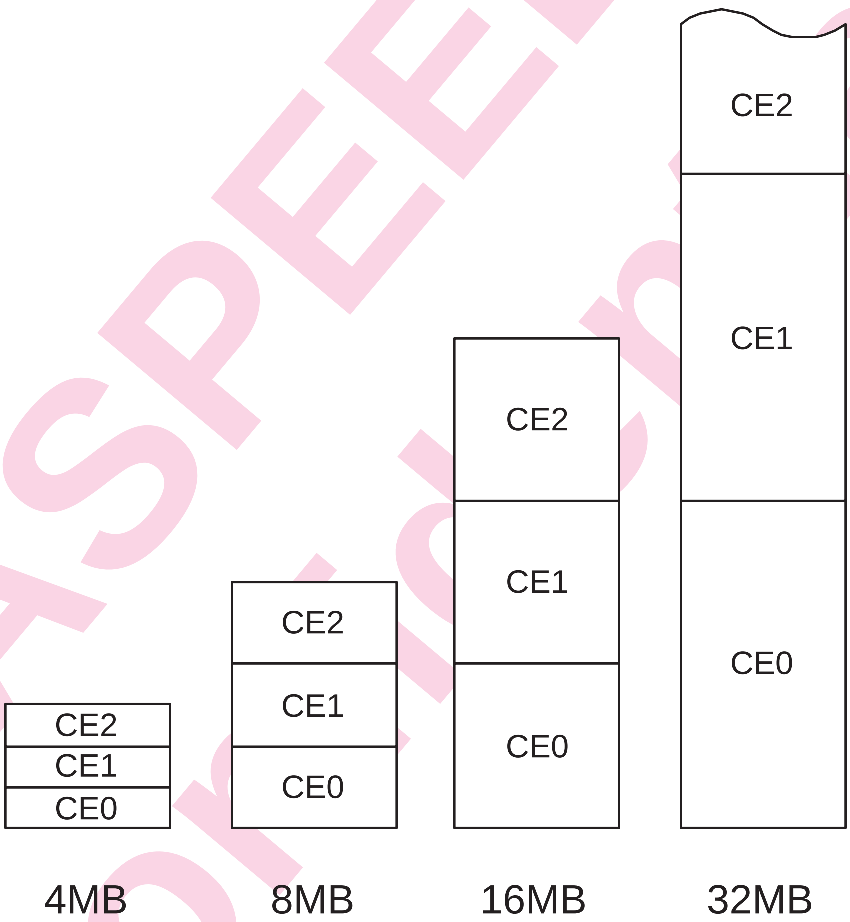


Figure 57: SMC Memory Space Organization

11.2 Timing Definition

- Figure 58: NOR type flash normal read timing
- Figure 59: NOR type flash normal write timing
- Figure 60: NOR type flash read/write cycle extended by external ACK
- Figure 61: NOR type flash setup time requirement of external ACK
- Figure 62: SPI type flash read/write timing
- Figure 63: SPI type flash read/write timing with dual input mode
- Figure 64: NAND type flash control timing

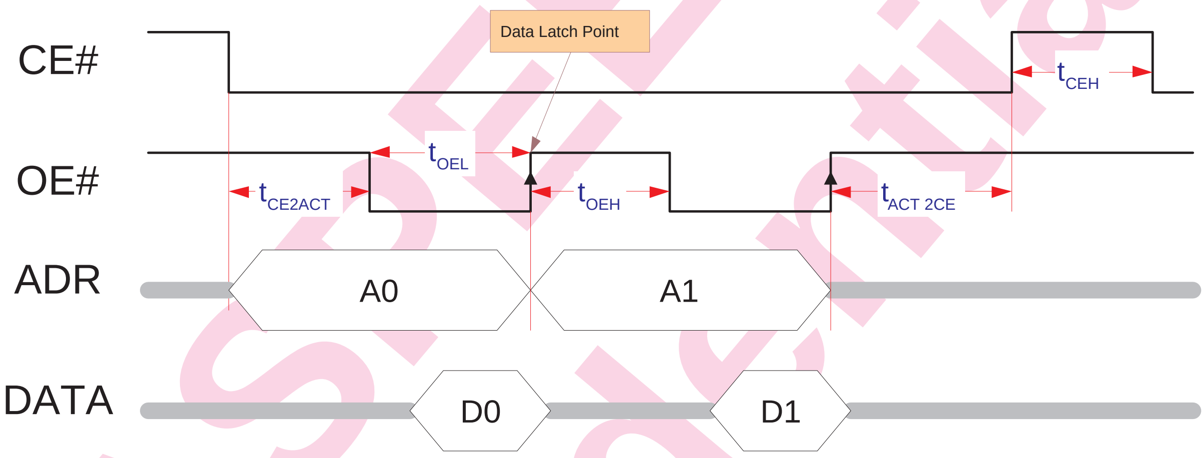


Figure 58: NOR Flash Read Timing

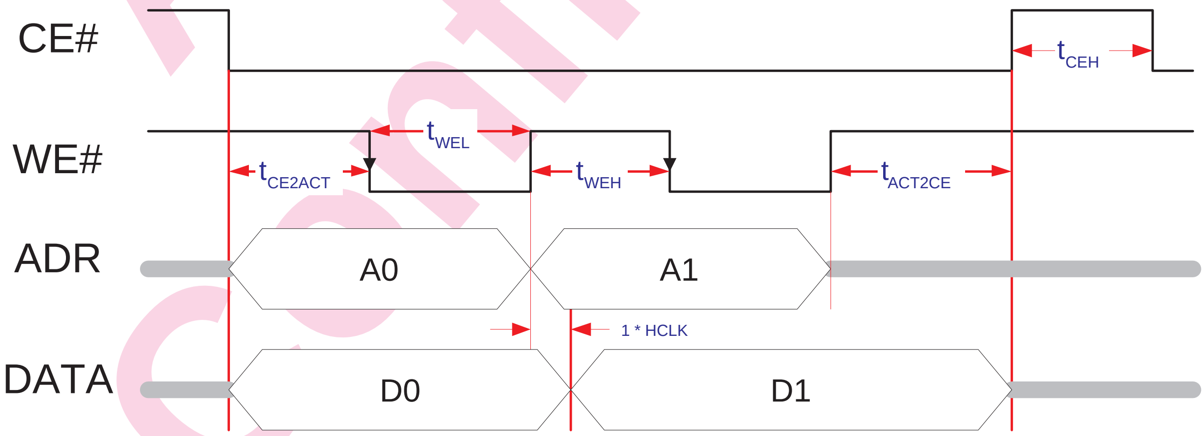


Figure 59: NOR Flash Write Timing

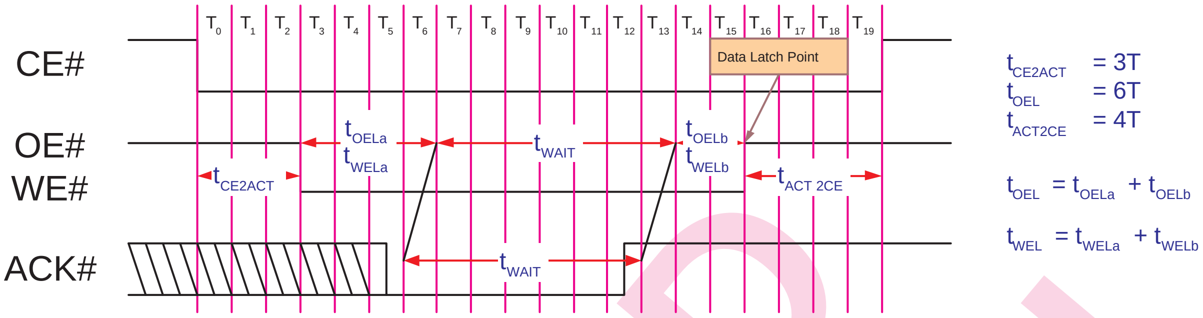


Figure 60: NOR Flash ACK Control Timing

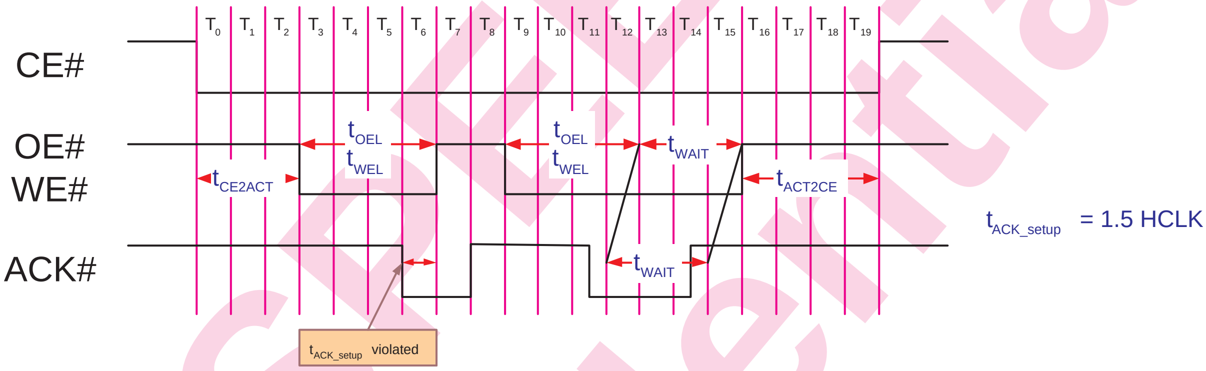


Figure 61: NOR Flash ACK Setup Timing

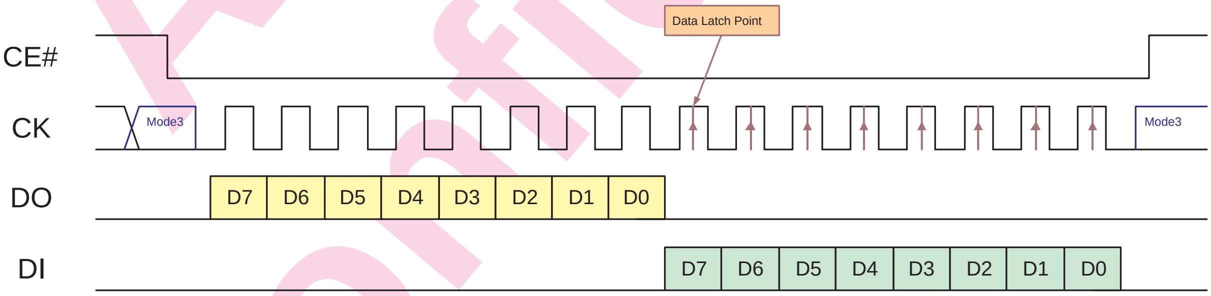


Figure 62: SPI Flash R/W Timing

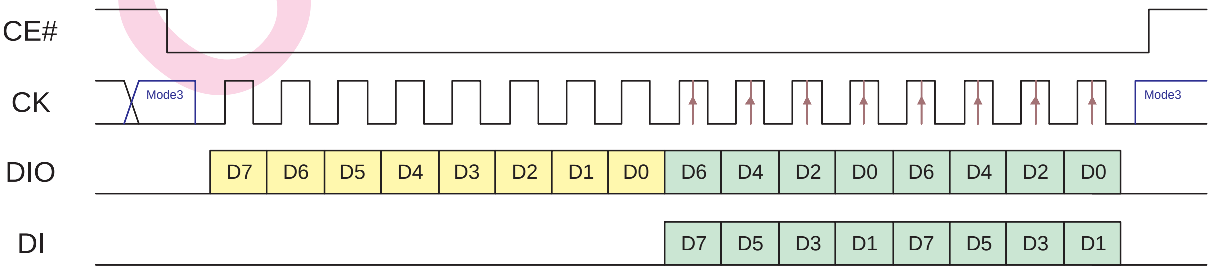


Figure 63: SPI Flash 2 Input R/W Timing

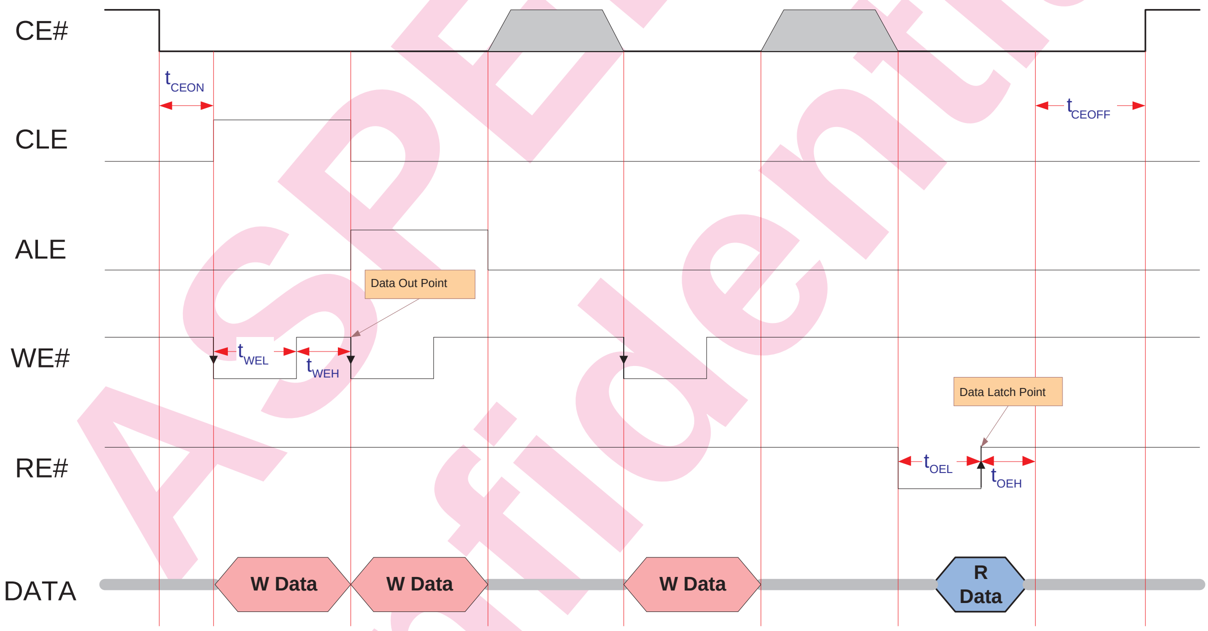


Figure 64: NAND Flash Timing

11.3 Registers : Base Address = 0x1600:0000

Offset: 00h		SMC00: CE0 Segment AC Timing Register — Init = 0x00000240
Bit	**R/W**	**Description**
31:13		**Reserved (0)**
12	RW	**Enable CE2 flash memory segment write** 0: CE2 segment is a read only segment 1: CE2 segment is a read and writable segment
11	RW	**Enable CE1 flash memory segment write** 0: CE1 segment is a read only segment 1: CE1 segment is a read and writable segment
10	RW	**Enable CE0 flash memory segment write** 0: CE0 segment is read only segment 1: CE0 segment is read and writable segment
9:8	RW	**CE2 flash type selection** 00: Select NOR flash type 01: Select NAND flash type 1x: Select SPI NOR flash type (default)
7:6	RW	**CE1 flash type selection** 00: Select NOR flash type 01: Select NAND flash type (default) 1x: Select SPI NOR flash type
5:4	RW	**CE0 flash type selection** 00: Select NOR flash type (default) 01: Select NAND flash type 1x: Select SPI NOR flash type
3:2	RW	**Reserved**
1:0	RW	**Segment size selection** 00: 32MB (default) 01: 16MB 10: 8MB 11: 4MB Segment size selection will determine the size of the addressing space for each chip select pin (CE0, CE1 and CE2). See Figure 57. The purpose of this register is to provide a possibility to assign a continuous addressing space for the three chip select pins, if required.

Offset: 04h Offset: 08h Offset: 0Ch		SMC04: CE0 Control Register — Init = 0 SMC08: CE1 Control Register — Init = 0 SMC0C: CE2 Control Register — Init = 0
Bit	**Attr.**	**Description**
Note : The definition of this register depends on the selected flash memory type. Therefore, it will depend on the setting of the register SMC00 [9:4].		
		NAND Flash Interface
31:28	RW	**WE# pulse width high (t-WEH)** 0000: 16T (1T = 1 HCLK clock) 0001: 15T 1111: 1T

to next page

from previous page

27:24	RW	**WE# pulse width low (t-WEL)** 0000: 16T (1T = 1 HCLK clock) 0001: 15T 1111: 1T
23:20	RW	**RE# pulse width high (t-REH)** 0000: 16T (1T = 1 HCLK clock) 0001: 15T 1111: 1T
19:16	RW	**RE# pulse width low (t-REL)** 0000: 16T (1T = 1 HCLK clock) 0001: 15T 1111: 1T
15:12	RW	**CE# active to command-start delay or command-end to CE# de-active delay (t-CESH)** 0000: 16T (1T = 1 HCLK clock) 0001: 15T 1111: 1T
11:10	RW	**WE# rising high edge to RE# falling low edge delay (t-WTR)** 00: 32T (1T = 1 HCLK clock) 01: 24T 10: 16T 11: 8T
9:4	RW	**Waiting time of Boot Mode read command busy (t-R)** 000000: around 63 us 000001: around 62 us .... 111111: around 1 us SMC integrates a timer with clock base 1MHz to count the waiting time, especially for read transfer time or write transfer time.
3	RW	**User mode row address cycle selection** 0: 3 Cycle, address from command data byte B0/B1/B2, other is don’t care 1: 2 Cycle, address from command data byte B0/B1, others are don’t care
2	RW	**User mode CE# active control mode** 0: CE# is active only when command is in progress 1: CE# is always active.
1	RW	**Random read capability (for boot mode only)** 0: No random read support 1: Support random read (Random read command must be 05h +2CA + E0h) Not all NAND flash memory devices can support random read. The page size of flash that support random read must be 2 KByte.

to next page

from previous page

0	RW	**Operation mode selection** **0: Select Boot Mode** (default) When selecting Boot Mode, SMC will automatically generate commands to access flash memory. Besides, NAND flash memory devices supporting Boot Mode must meet the following features: • Bit page organization: each page must have 2048 bytes • Page read command: 00h + 2CA + 3RA + 30h • Random read command: 05h + 2CA + E0h • Read transfer busy time must be less than 64us • The first block must be guaranteed to be valid and no ECC required **1: Select User Mode** When selecting User Mode, only the address bit [13:12] of AHB bus command will be decoded, automatically ignoring higher address bits, and the data of AHB bus command will be used as read/write data depending on the operation mode. The command decoding by address bit [13:12] is: 00: Read/Write data, depending on read/write operations. 01: CLE command write phase, CLE command is fetched from command data bit[7:0] (only write operation is accepted) 10: ALE command column address write phase, 2 bytes column address is fetched from command data bit[15:0] (only write operation is accepted) 11: ALE command row address write phase, 2 or 3 (defined at bit[3]) bytes row address is fetched from command data bit[23:0] (only write operation is accepted)
Note : 2CA means 2 bytes of Column Address. 3RA means 3 bytes of Row Address.		
SPI Flash Interface		
31:28	RW	**Reserved**
27:24	RW	**CE# Inactive pulse width** 0000: 16T (1T = 1 HCLK clock) 0001: 15T 1111: 1T
23:16	RW	**Command data** The content of this register is used as the data for Fast Read or Byte Write CMD phase.
15:13	RW	**Reserved**
12	RW	**Disable SPI flash read command merge** 0: Enable 1: Disable (with performance penalty) Set this bit will disable the SPI controller to merge continuous address reads. By default, continuous addresses reads will be merged to reduce the command overhead while read commands occur within 16 clocks.
11	RW	**Reserved**

to next page

from previous page

10:8	RW	**SPI clock frequency selection (t-CK)** 000: HCLK/16 (default) 001: HCLK/14 010: HCLK/12 011: HCLK/10 100: HCLK/8 101: HCLK/6 110: HCLK/4 111: HCLK/2
7:6	RW	**Dummy cycles before data for fast read command** 00: 0 Byte (default) 01: 1 Byte 10: 2 Byte 11: 3 Byte
5	RW	**MSB/LSB first control** 0: MSB First (default for boot code) 1: LSB First
4	RW	**Clock Mode_0/Mode_3 selection** 0: Select Mode 0 (The initial state of clock signal is 0) 1: Select Mode 3 (The initial sate of clock signal is 1)
3	RW	**Enable dual data input mode** 0: 1 bit data input each clock 1: 2 bits data input each clock When enabled this bit and the SPI flash memory device supports dual data input mode, the data rate will be doubled.
2	RW	**User Mode CE# Active Control** When in User Mode, SPI command cycle will be activated (CE# low) until set this bit to 1. Thats to say setting this bit to 1 will stop activating SPI interface after Read/Write operation finished or immediately if no Read/Write operation is in progress. But when different CE command is entered, SPI interface will be deactivated immediately.
1:0	RW	**Command Mode** 00: Normal Read (03h + Address + Read data [1/2/3/4 bytes]) 01: Fast Read (CMD + Address + Read data [1/2/3/4 bytes]) 10: Normal Write (CMD + Address + Write data [1/2/3/4 bytes]) 11: User Mode (Read/write data [1/2/3/4 bytes]) At this mode, address has no meaning, all address decoded in the segment address are valid, and data will be read/write to/from the LSB byte first of each 32 bits AHB command. This mode provides a flexible programming method for specific command type other than Read/Write command supported. CMD = 1 byte of data from bit[23:16] of this register Address = 3 bytes of data from address bus of AHB and output in the order from MSB byte to LSB byte Read/write data = 1∼4 bytes data from AHB data bus and output in the order from LSB byte to MSB byte Except User Mode, the address space can support up to 16 MBytes maximum.
For NOR Flash Interface		

to next page

from previous page

31:30	RW	**Timer value unit** 00: 0.5 us 01: 1.0 us 10: 2.0 us 11: 4.0 us This timer value is defined at SMC10 [31:24]
29:28	RW	**Operation mode** 0x: **Normal mode** 10: **t-WEL and t-OEL long mode**, the low pulse width of OE# and WE# is controlled by the timer value setting at SMC10 [31:24]. 11: **ACK control mode**. The OE# and WE# low pulse width can be stretched by an external ACK# input pin, if ACK# is pulled low 2 HCLK clock cycle before t-OEL or t-WEL timeout, then t-OEL and t-WEL timer will be stopped until ACK# is being release to high. Else if ACK# too long and expires the timeout setting then the ACK# input will be ignored and the bus cycle will continue to finish. At this time, there is a interrupt can be used to acknowledge this case.
27:24	RW	**CE# high pulse width for each AHB bus command (t-CEH)** 0000: No CE# high pulse width requirement 0001: > 2T (1T = 1HCLK) 0001: > 3T .... 1111: > 16T This timing defines the CE# high pulse minimum width requirement, if it is set to nonzero value, then CE# will raise high and keep at least the defined cycles for each AHB bus command. It still will have burst possibility if the AHB bus command is not a byte command.
23:20	RW	**OE#/WE# High to CE# High Delay (t-ACT2CE)**
19:16	RW	**WE# High Pulse Width (t-WEH)**
15:12	RW	**WE# Low Pulse Width (t-WEL)**
11:8	RW	**OE# High Pulse Width (t-OEH)**
7:4	RW	**OE# Low Pulse Width (t-OEL)**
3:0	RW	**CE# Low to OE#/WE# Low Delay (t-CE2ACT)** The following table can be applied to all the above timing settings of bit[23:0]. 0000: 16T (default) .... 1110: 2T 1111: 1T (1T = period of HCLK) These registers define the read/write timings for "NOR" flash read/write cycles. For read cycles, read data are latched at the rising edge of OE# signal. For write cycles, write data are latched at the rising edge of WE# signal.

Offset: 10h		SMC10: Misc. Control Register — Init = 0
Bit	**R/W**	**Description**
31:24	RW	**NOR timer value setting** The timer unit is defined at the NOR control register at SMC04, SMC08 and SMC0C.

to next page

from previous page

23	RW	**NOR ACK# control timeout interrupt status** 0 : Not timeout 1 : Timer timeout Write '1' to clear this bit value.
22	RW	**NOR ACK# control timeout interrupt enable** 0: Disable 1: Enable
21	RW	**NAND timer interrupt status** 0 : Not timeout 1 : Timer timeout Write '1' to clear this bit value.
20	RW	**NAND timer interrupt enable control** 0: Disable 1: Enable
19	RW	**NAND timer enable** 0 : Disable timer, and timer value will be reset 1 : Enable timer operation The timer is valid only at User Mode. The procedure of using timer: each step must be an independent command 1. Set Timer Value and clear Interrupt Status 2. Enable Timer and Interrupt Control 3. Wait Interrupt 4. Disable Timer 5. Clear Interrupt Status 6. go to step 2 for next timer usage
18:8	RW	**NAND Timer Value Setting** 0 : disable timer others : value * 4 us
7:6	RW	**NAND ECC Mode Selection** 00 : 256 bytes 01 : 512 bytes 10 : 1024 bytes 11 : 2048 bytes This mode selection control the shifting of ECC result to the exact bit size.
5	RW	**WP# output value** 0: Write function disabled 1: Write function enabled When WP# pin supported, this bit is used as the output value of WP# pin.
4	RW	**WP# pin supported** 0: Not support 1: Supported This bit controls the WP# pin output enable. And WP# pin must resistor pull low externally.
3	R	**R/B# pin input value** 0: R/B# pin is in busy state 1: R/B# pin is in normal state

to next page

from previous page

2	RW	**R/B# rising edge detect status** 0: No rising edge detected 1: Rising edge is detected When R/B# pin is not supported, this bit is always '0'. Write '1' to this bit will clear this register.
1	RW	**Enable R/B# status interrupt** 0: Disable R/B# status interrupt 1: Enable R/B# status interrupt (generate interrupt when R/B# pin rising edge is detected)
0	RW	**R/B# Pin Supported** 0: Not support 1: Supported At Boot Mode, read transfer time (t-R) will be determined by both R/B# status and the timer to count waiting time, depending on which one happening firstly. At User Mode, read transfer time will be controlled by R/B# status interrupt function.
Note : R/B# and WP# pins not only can be used for NAND flash, NOR flash also can use it.		

Offset: 14h — SMC14: NAND ECC Generation Control/Status register — Init = 0

Bit	R/W	Description
31:30		**Reserved (0)**
29	RW	**ECC Reset Enable** 0 : NOP 1 : Reset ECC buffer
28	RW	**ECC Generation Enable** Support maximum 2048 bytes SECDED type ECC code generation. 0 : Disable generation 1 : Enable generation for read/write command
27:0	R	**ECC Value** n:0 : Even number ECC value, bit[0,2,4,6], byte[0,2,4,6,...] 2n:n+1 : Odd number ECC value, bit[1,3,5,7], byte[1,3,5,7,...] For 2048 bytes ECC : bit 27:0 are valid, n = 14 For 1024 bytes ECC : bit 25:0 are valid, n = 13 For 512 bytes ECC : bit 23:0 are valid, n = 12 For 256 bytes ECC : bit 21:0 are valid, n = 11 For smaller size ECC generation, SW must discard the unused bits by themselves. This value can be reset by set bit[29] to '1'.

Offset: 18h — SMC18: NAND ECC check value — Init = 0

Bit	R/W	Description
31:28		**Reserved (0)**

to next page

from previous page

27:0	RW	**ECC Value** n:0 : Even number ECC value, bit[0,2,4,6], byte[0,2,4,6,...] 2n:n+1 : Odd number ECC value, bit[1,3,5,7], byte[1,3,5,7,...] For 2048 bytes ECC : bit 27:0 are valid, n = 14 For 1024 bytes ECC : bit 25:0 are valid, n = 13 For 512 bytes ECC : bit 23:0 are valid, n = 12 For 256 bytes ECC : bit 21:0 are valid, n = 11 For all bits not used, please fill with ’0’. This value will operates with Generated ECC Value to generate ECC Check Result. This is a hardware auxiliary ECC check function, SW fill this register with the ECC value stored in flash, and then can read the ECC check status immediately from next register. The ECC check function only useful for Flash Read.

Offset: 1Ch **SMC1C: NAND ECC check result** **Init = 0**

Bit	R/W	Description
31	R	**ECC Unrecoverable Error.** 0: No unrecoverable error occurs. 1: Indicates more than 1 bit error occurs, and can not be recovered.
30	R	**ECC Field Error, doesn’t need correct** 0: No error occurs. 1: Indicates ECC field 1 bit error, and no correct needed.
29	R	**ECC Recoverable Error, need correct** 0: No error occurs. 1: Indicate 1 bit error, and need SW to correct the bit position show in bit[13:0]
28	R	**ECC Check Pass** 0: Indicate error occurs 1: Indicate ECC check Pass, and no correct needed
27:16	R	**ECC Accumulate Counter** This counter value shows the data byte count number of ECC counted. When ECC reset control SMC14 [29] been set to ’1’, this counter will be cleared to 0.
15:14		**Reserved (0)**
13:3	R	**ECC Recoverable Error Byte Position** This value shows the error byte position if ECC error and need correct.
2:0	R	**ECC Recoverable Error Bit Position** This value shows the error bit position if ECC error and need correct.

12 AHB Bus Controller

12.1 Overview

Advanced High-performance Bus Controller (AHBC) supports a mechanism, including a priority arbiter, an address decoder and a data multiplexer, to control the overall operations of Advanced High-performance Bus (AHB), which is the main system bus for ARM CPU to communicate with the related peripherals. The priority arbiter, by round-robin arbitration scheme, assigns which bus master gets the right to access AHB for the moment. Each bus master has its own REQUEST/GRANT interface to the priority arbiter. The address decoder performs a centralized address decoding function.

AHBC also provide remapping mechanism to speed up the access time of program code.

AHBC totally implements 4 sets of 32-bit registers, which are listed below, to program the various functions supported by AHBC. Each register has its own specific offset value to derive its physical address location.

Base address of AHBC = 0x1E60_0000
Physical address = (Base address of AHBC) + Offset

AHBC00: Protection Key Register
AHBC80: Priority Control Register
AHBC88: interrupt Control Register
AHBC8C: Address Remapping Register

12.2 Features

- Directly connected to internal AHB bus
- AHB master and slave controller
- AHB bus multiplexer
- AHB slave address decoder
- AHB master controller with two-level arbitration (round-robin arbitration for each arbitration level)
- AHB memory address remapping control with register-write protection

12.3 Registers : Base Address = 0x1E60:0000

Offset: 00h		AHBC00: Protection Key Register Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Protect Key** Write 0xAEED_1A03: register 0x80∼0x8C is programmable. Write Others: register 0x80∼0x8C is not programmable. Read 1: means key opened. Read 0: means key locked.

Offset: 80h		AHBC80: Priority Control Register Init = 0
Bit	**R/W**	**Description**
31:16		**Reserved (0)**
15:0	RW	**Priority Level Selection** Bit n represents the level of master n on AHB 0: lower priority level 1: higher priority level AHBC can support up to 15 bus masters. The arbiter supports a two-level mechanism to arbitrate master requests. Each master can be programmed to a higher level or lower level. The following table shows the bit assignment of priority control. Bit[15] : Reserved Bit[14] : Reserved Bit[13] : Reserved Bit[12] : Reserved Bit[11] : Reserved Bit[10] : Reserved Bit[9] : Reserved Bit[8] : Reserved Bit[7] : Reserved Bit[6] : Reserved Bit[5] : LPC Master Bit[4] : PCI Host Bit[3] : CPU Data Bit[2] : CPU Instruction Bit[1] : P-Bus to AHB Bridge Bit[0] : Reserved

Offset: 88h		AHBC88: Interrupt Control Register Init = 0
Bit	**R/W**	**Description**
31:25		**Reserved (0)**
24	RW	**Interrupt status before mask** 0: interrupt never occur 1: interrupt occur Clear interrupt by write ’0’ to this register bit.
23:22		**Reserved (0)**

to next page

from previous page

21:20	R	**Response status** After decoder receives a non-existing address, the decoder responds the status to master. Those bits will represent the status bits. 00: OK response 01: ERROR response 1x: Reserved
19:17		**Reserved (0)**
16	R	**Enable Interrupt** 0: Disable interrupt 1: Enable interrupt Set this bit to '1', when bit[24] is 1, interrupt signal of AHB controller will interrupt CPU. When S/W wants to enable the interrupt, first clear bit[24] to avoid older interrupt status interrupt CPU.
15:0		**Reserved (0)**

Offset: 8Ch — AHBC8C: Address Remapping Register — Init = 0

Bit	R/W	Description
31:10		**Reserved (0)**
9:6	RW	**Reserved**
5	RW	**PCI Remap 1** 0: Disable mapping 1: Enable mapping This bit enables the mapping of physical address space range 0x8000_0000 ∼ 0xFFFF_FFFF to PCI Host controller. Remap mechanism provide to speed up access time of program code.
4	RW	**PCI Remap 0** 0: Disable mapping 1: Enable mapping This bit enables the mapping of physical address space range 0x6000_0000 ∼ 0x7FFF_FFFF to PCI Host controller. Remap mechanism provide to speed up access time of program code.
3:1	RW	**Reserved**
0	RW	**Boot Area Remap** This bit controls the physical address space range 0x0000_0000 ∼ 0x0FFF_FFFF mapping to what devices. 0: Mapping to Static memory 1: Mapping to SDRAM memory

13 Memory Integrity Check Controller

13.1 Overview

Memory Integrity Check Engine (MICE) implements 8 registers, which is listed below, to program the various functions supported by AST2050 / AST1100 . Each register has its own specific offset value to derive its physical address location.

Base address of MICE = 0x1E64_0000
Physical address = (Base address of MICE) + Offset

MIC00: Base Address of Control Buffer Register
MIC04: Base Address of Checksum Buffer Register
MIC08: Rate Control Register
MIC0C: Control Register
MIC10: Stop-Page Register
MIC14: Error Status and Interrupt Mask Register
MIC18: First Page Error Status Register
MIC1C: Secondary Page Error Status Register

13.2 Features

- Directly connected to AHB bus
- Automatic memory integrity check by constant checksum scanning
- Directly access SDRAM memory through M-bus
- 4K bytes of memory per checksum unit
- Each checksum unit can be individually enabled or skipped
- Support Fletchers Checksum algorithm
- Programmable scanning rate and scanning range control
- Slight extra memory bandwidth and capacity demand
- Generate an interrupt whenever detecting checksum errors

13.3 Registers : Base Address = 0x1E64:0000

Offset: 00h		MIC00: Base Address of Control Buffer Register	Init = X
Bit	**R/W**	**Description**	
31:28		**Reserved (0)**	
27:3	RW	**Base address of control buffer [27:3]** This register determines the base address of control buffer which storing the control information for all memory pages. The base address of control buffer has to be 8-byte aligned. Therefore, address bit [2:0] are always 0.	
2 :0		**Reserved (0)**	

Offset: 04h		MIC04: Base Address of Checksum Buffer Register Init = X
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:3	RW	**Base address of checksum buffer [27:3]** This register determines the base address of checksum buffer which storing the checksum information for all memory pages. The base address of checksum buffer has to be 8-byte aligned. Therefore, address bit [2:0] are always 0.
2 :0		**Reserved (0)**

Offset: 08h		MIC08: Rate Control Register Init = X
Bit	**R/W**	**Description**
31:16		**Reserved (0)**
15:0	RW	**Rate control setting** This register determines the rate of MICE doing memory integrity check. Higher value means slower rate.

Offset: 0Ch		MIC0C: Control Register Init = 0xxx_xxxxh
Bit	**R/W**	**Description**
31:29		**Reserved (0)**
28	RW	**Enable MICE** 0: Reset MICE (default) 1: Enable MICE Before enabling MICE, the contents of control buffer and checksum buffer have to be prepared. The related registers have to be programmed as well.
27:12	RW	**Number of pages to be checked** There are 16 bits of register to program the number of pages to be checked when MICE is enabled. Therefore, the maximum number of pages to be checked is 64K pages. It implies that the maximum memory size that MICE can support is 4KB*64K = 256MB. The first page to be checked is always page #0 which is with starting memory address 0000_0000h. The number of pages must be 16 aligned (#15, or #31, or #47, ...). See "Control Buffer Format" & "Checksum Buffer Format" for detail information.
11:0		**Reserved (0)**

Offset: 10h		MIC10: Stop-Page Register Init = X
Bit	**R/W**	**Description**
31:16	RW	**Writ-back value for checksum buffer**
15:0	RW	**Page number of stop-page**

to next page

from previous page

Note :
MIC10 [15:0] must not bigger than MIC0C [27:12].

If (MIC10 [31:16] != 0), MICE will write a 32-bits value (Bit 31∼16 = MIC10 [31:16], Bit 15∼0 = 0) to the checksum buffer of page number #N which N equal to MIC10 [15:0].

Writing this register will also cause MICE to stop and skip the memory integrity check at current process page when current process page number equal to MIC10 [15:0].

Offset: 14h — MIC14: Error Status and Interrupt Mask Register — Init = 0

Bit	R/W	Description
31		**Reserved (0)**
30	R	**Lost page error flag** 0: Lost page error has NOT been detected 1: Lost page error has been detected
29	R	**Secondary page error flag (come from MIC1C [29])** 0: Secondary page error has NOT been detected 1: Secondary page error has been detected
28	R	**First page error flag (come from MIC18 [28])** 0: Secondary page error has NOT been detected 1: Secondary page error has been detected
27:18		**Reserved (0)**
17:16	RW	**Interrupt mask bits** 0x: Disable engine interrupt to CPU when secondary page error flag is set 1x: Enable engine interrupt to CPU when secondary page error flag is set x0: Disable engine interrupt to CPU when first page error flag is set x1: Enable engine interrupt to CPU when first page error flag is set
15:0	R	**Process page number of engine**

Offset: 18h — MIC18: First Page Error Status Register — Init = 0000_xxxx

Bit	R/W	Description
31		**Reserved (0)**
30	R	**Lost page error flag (come from MIC14 [30])** 0: Lost page error has NOT been detected 1: Lost page error has been detected
29		**Reserved (0)**
28	RW	**First Page Error Flag (cleared by writing 1)** 0: First page error has NOT been detected 1: First page error has been detected
27:16		**Reserved (0)**
15:0	R	**Page Number of First Page Error**

Offset: 1Ch — MIC1C: Secondary Page Error Status Register — Init = 0000_xxxx

Bit	R/W	Description
31		**Reserved (0)**
30	R	**Lost page error flag (come from MIC14 [30])** 0: Lost page error has NOT been detected 1: Lost page error has been detected

to next page

from previous page

29	RW	**Secondary Page Error Flag (cleared by writing 1)** 0: Secondary page error has NOT been detected 1: Secondary page error has been detected
28:16		**Reserved (0)**
15:0	R	**Page Number of Secondary Page Error**

13.4 Page Control Bits

- Page Control Bits: Behavior of MICE on each page is presented with 2 bits.

Page Control Bits	Read DRAM Data	Update Checksum Buffer	Update Error Status
`00 (Skip)`	No	No	No
`01 (ECC Mode)`	Yes	No	No
`10 (Debug Mode)`	Yes	Always	No
`11 (MIC Mode)`	Yes	When checksum buffer is initiative value	Yes

13.5 Control Buffer Format

- Page Control Bits: Behavior of MICE on each page is presented with **2 bits**.
- M = MIC0C [27:12]

Bit Range	Description
`000 - 001`	Page control bits of page number #0 (address 0000_0000h - 0000_0FFFh)
`002 - 003`	Page control bits of page number #1 (address 0000_1000h - 0000_1FFFh)
`004 - 005`	Page control bits of page number #2 (address 0000_2000h - 0000_2FFFh)
`006 - 007`	Page control bits of page number #3 (address 0000_3000h - 0000_3FFFh)
`008 - 009`	Page control bits of page number #4 (address 0000_4000h - 0000_4FFFh)
`00A - 00B`	Page control bits of page number #5 (address 0000_5000h - 0000_5FFFh)
`00C - 00D`	Page control bits of page number #6 (address 0000_6000h - 0000_6FFFh)
`00E - 00F`	Page control bits of page number #7 (address 0000_7000h - 0000_7FFFh)
`.....`	.....
`2*M - 2*M+1`	Page control bits of page number #M (address 1000h*M - 1000h*M+FFFh)

13.6 Checksum Buffer Format

- Checksum Value Bytes: Checksum value on each page is presented with **4 bytes**.
- Software need to initiate the checksum buffer with value 0.
- M = MIC0C [27:12]

Byte Range	Description
`000 - 003`	Checksum value of page number #0 (address 0000_0000h - 0000_0FFFh)
`004 - 007`	Checksum value of page number #1 (address 0000_1000h - 0000_1FFFh)
`008 - 00B`	Checksum value of page number #2 (address 0000_2000h - 0000_2FFFh)
`00C - 00F`	Checksum value of page number #3 (address 0000_3000h - 0000_3FFFh)
`010 - 013`	Checksum value of page number #4 (address 0000_4000h - 0000_4FFFh)
`014 - 017`	Checksum value of page number #5 (address 0000_5000h - 0000_5FFFh)
`018 - 01B`	Checksum value of page number #6 (address 0000_6000h - 0000_6FFFh)
`01C - 01F`	Checksum value of page number #7 (address 0000_7000h - 0000_7FFFh)
`.....`	.....
`4*M - 4*M+3`	Checksum value of page number #M (address 1000h*M - 1000h*M+FFFh)

13.7 Porgramming Sequence

13.7.1 Parameter Definition

- $Max\_CheckSumMem\_Size$ (4K-byte aligned):
  The maximum memory size to be checked when MICE is enable.

- Max_Page_Number:
  The maximum page number to be checked when MICE is enable.

  Max_Page_Number = ($Max\_CheckSumMem\_Size$ - 1) $>>$ 12.

- $Max\_Page\_Number\_16Aligned$ (16 page aligned):
  The 16 aligned maximum page number.

  $Max\_Page\_Number\_16Aligned$ = (((Max_Page_Number $>>$ 4) + 1) $<<$ 4) - 1.

- $Page\_ControlBuf\_Base\_Adr$ (8-byte aligned):
  Base address of page control buffer which store page control bits.
  Size of page control buffer is 2*$Max\_Page\_Number\_16Aligned$ bits.

- $Page\_CheckSumBuf\_Base\_Adr$ (8-byte aligned):
  Base address of page checksum buffer which store checksum value.
  Size of page checksum buffer is 4*$Max\_Page\_Number\_16Aligned$ bytes.

- $Rate\_Control\_Val$:
  Maximum DRAM read/write request rate of MICE.

  $Request\_Rate$ = 1 / (1 + $Rate\_Control\_Val$).

- $Interrupt\_Mask$:
  Set bit with value 1 to enable interrupt.

- TAG:
  A 16 bits accumulative tag value with value 0 is invalid.

13.7.2 MIC Engine Initiation

1. Clear page control buffer with value 0.
2. Clear page checksum buffer with value 0.
3. Clear TAG with value 1.
4. HACE00 = $Page\_ControlBuf\_Base\_Adr$ (8-byte aligned).
5. HACE04 = $Page\_CheckSumBuf\_Base\_Adr$ (8-byte aligned).
6. HACE08 = $Rate\_Control\_Val$.
7. HACE0c = ($Max\_Page\_Number\_16Aligned$ $<<$ 12).
8. HACE14 = $Interrupt\_Mask$ $<<$ 16.
9. Read HACE0c untill HACE0c equal to ($Max\_Page\_Number\_16Aligned$ $<<$ 12).
10. HACE0c = (10000000h $|$ ($Max\_Page\_Number\_16Aligned$ $<<$ 12)).

13.7.3 Start Page CheckSum Process

This process will start checksum algorithm at page number #n.

1. Set page control bits of page number #n to 3h.

13.7.4 Stop Page CheckSum Process

This process will stop checksum algorithm at page number #N.

1. Set page control bits of page number #N to 0h.
2. Write MIC10[31:0] = (N $|$ ($TAG <<$ 16)).
3. Read checksum value of page number #N, until this value equal to ($TAG <<$ 16).
4. If TAG equal to ffffh then TAG = 1, otherwise TAG = TAG + 1.

13.8 Interrupt Behavior

- page error: MICE find page checksum error

- Write MIC18[28] to 1: The action of clear First Page Error Flag
- MIC18[28]: First Page Error Flag

- Write MIC1C[29] to 1: The action of clear Secondary Page Error Flag
- MIC1C[29]: Secondary Page Error Flag

- MIC14[30]: Lost Page Error Flag

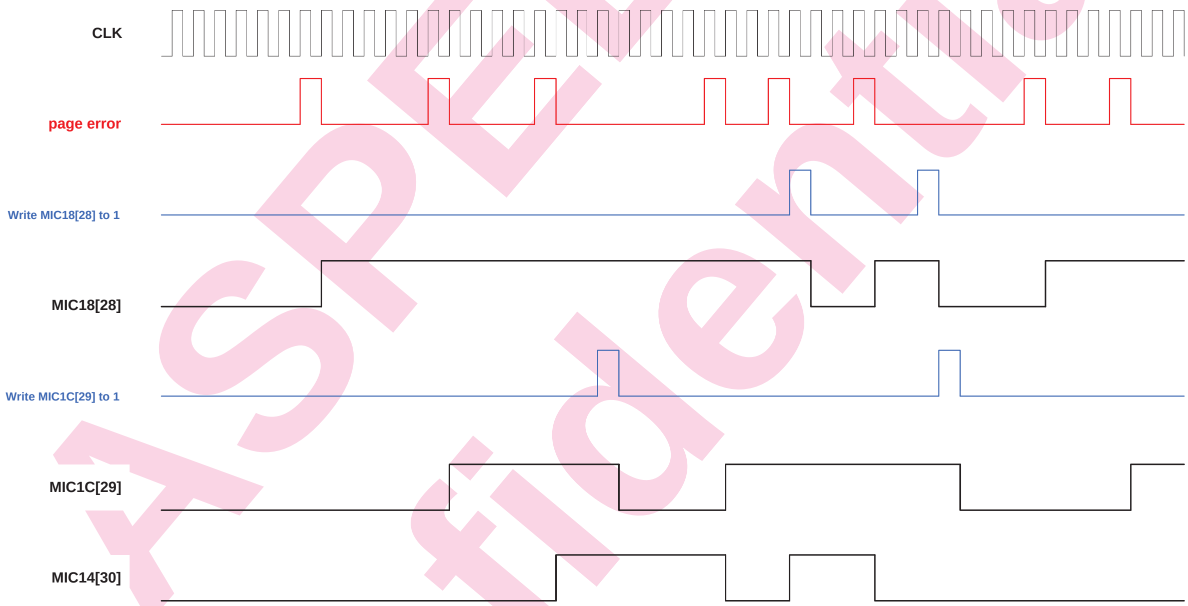


Figure 65: Interrupt Flag Priority

14 10/100 Ethernet MAC Controller

14.1 Overview

AST2050 / AST1100 integrates two sets of high performance Ethernet MAC modules which can operate at 10 Mbps or 100 Mbps. The two MAC modules are totally identical and can be enabled or disabled independently. The only difference is that each MAC has its own base addresses for register programming. The digital interface of each MAC can be RMII, MII, or GMII. Due the pin count limitation, there will be only one GMII interface allowed. It means that when GMII is enabled, there should be only one MAC to be enabled.

This is a superset of registers definition. For AST2050/AST1100 chip, only 10M/100M interface is supported.

Base address of Ethernet MAC #1 = 0x1E66_0000
Base address of Ethernet MAC #2 = 0x1E68_0000
Physical address = (Base address of Ethernet MAC) + Offset

MAC00: Interrupt Status Register (ISR)
MAC04: Interrupt Enable Register (IME)
MAC08: MAC Most Significant Address Register (MAC_MADR)
MAC0C: MAC Least Significant Address Register (MAC_LADR)
MAC10: Multicast Address Hash Table 0 Register (MAHT0)
MAC14: Multicast Address Hash Table 1 Register (MAHT1)
MAC18: Normal Priority Transmit Poll Demand Register (NPTXPD)
MAC1C: Receive Poll Demand Register (RXPD)
MAC20: Normal Priority Transmit Ring Base Address Register (NPTXR_BADR)
MAC24: Receive Ring Base Address Register (RXR_BADR)
MAC28: High Priority Transmit Poll Demand Register (HPTXPD)
MAC2C: High Priority Transmit Ring Base Address Register (HPTXR_BADR)
MAC30: Interrupt Timer Control Register (ITC)
MAC34: Automatic Polling Timer Control Register (APTC)
MAC38: DMA Burst Length and Arbitration Control Register (DBLAC)
MAC3C: DMA/FIFO State Register (DMAFIFOS)
MAC44: Feature Register (FEAR)
MAC48: Transmit Priority Arbitration and FIFO Control Register (TPAFCR)
MAC4C: Receive Buffer Size Register (RBSR)
MAC50: MAC Control Register (MACCR)
MAC54: MAC Status Register (MACSR)
MAC58: Test Mode Register (TM)
MAC60: PHY Control Register (PHYCR)
MAC64: PHY Data Register (PHYDATA)
MAC68: Flow Control Register (FCR)
MAC6C: Back Pressure Register (BPR)
MAC70: Power Control Register (PWRTC)
MAC90: Normal Priority Transmit Ring Pointer Register (NPTXR_PTR)
MAC94: High Priority Transmit Ring Pointer Register (HPTXR_PTR)
MAC98: Receive Ring Pointer Register (HPTXR_PTR)
MACA0: TPKT_CNT Counter Register
MACA4: TXMCOL_CNT and TXSCOL_CNT Counter Register
MACA8: TXECOL_CNT and TXFAIL_CNT Counter Register
MACAC: RUNT_CNT and TXLCOL_CNT Counter Register
MACB0: RPKT_CNT Counter Register
MACB4: BROPKT_CNT Counter Register
MACB8: MULPKT_CNT Counter Register
MACBC: RPF_CNT and AEP_CNT Counter Register
MACC0: RUNT_CNT Counter Register

MACC4: CRCER_CNT and FTL_CNT Counter Register
MACC8: RCOL_CNT and RLOST_CNT Counter Register

14.2 Features

- Integrate dual MAC modules compliant with IEEE802.3 and IEEE802.3z specification
- Support 10/100/1000M bps transfer rates
- Support Media Independent Interface (MII x2), Reduced Media Independent Interface (RMII x2), Gigabit Media Independent Interface (GMII x1)
- Adopt AHB bus interface supporting bus master and slave modes
- Integrated link list DMA engine with direct M-Bus accesses for transmitting and receiving packets
- Support IEEE 802.1Q VLAN tag insertion and deletion
- Support High Priority Transmit Queue for QoS and CoS applications
- Independent TX/RX FIFO
- Support half and full duplex (1000 Mbps mode only supports full duplex)
- Support flow control for full duplex and backpressure for half duplex

14.3 Registers :

Base address of Ethernet MAC #1 = 0x1E66:0000
Base address of Ethernet MAC #2 = 0x1E68:0000

Offset: 00h		MAC00: Interrupt Status Register (ISR) Init = 0
Bit	**R/W**	**Description**
31:11		**Reserved (0)**
10	RW	**HPTXBUF_UNAVA: High priority transmit buffer unavailable** Writing ”1” to this bit will clear this status flag.
9	RW	**PHYSTS_CHG: PHY link status change** Writing ”1” to this bit will clear this status flag.
8	RW	**AHB_ERR: AHB bus error** Writing ”1” to this bit will clear this status flag.
7	RW	**TPKT_LOST** Packets transmitted to Ethernet lost due to late collision or excessive collision or under-run Writing ”1” to this bit will clear this status flag.
6	RW	**NPTXBUF_UNAVA: Normal priority transmit buffer unavailable** Writing ”1” to this bit will clear this status flag.
5	RW	**TPKT2F: TXDMA has moved data into the TX FIFO** Writing ”1” to this bit will clear this status flag.
4	RW	**TPKT2E: Packets transmitted to Ethernet successfully** Writing ”1” to this bit will clear this status flag.
3	RW	**RPKT_LOST: Received packet lost due to RX FIFO ful** Writing ”1” to this bit will clear this status flag.
2	RW	**RXBUF_UNAVA: Receiving buffer unavailable** Writing ”1” to this bit will clear this status flag.

to next page

from previous page

1	RW	**RPKT2F: Packets received into RX FIFO successfully** Writing ”1” to this bit will clear this status flag.
0	RW	**RPKT2B: RXDMA has received packets to RX buffer successfully** Writing ”1” to this bit will clear this status flag.

Offset: 04h		**MAC04: Interrupt Enable Register (IME)** **Init = 0**
Bit	**R/W**	**Description**
31:11		**Reserved (0)**
10	RW	**HPTXBUF_UNAVA_EN** Interrupt enable of ISR [10]
9	RW	**PHYSTS_CHG_EN** Interrupt enable of ISR [9]
8	RW	**AHB_ERR_EN** Interrupt enable of ISR [8]
7	RW	**TPKT_LOST_EN** Interrupt enable of ISR [7]
6	RW	**NPTXBUF_UNAVA_EN** Interrupt enable of ISR [6]
5	RW	**TPKT2F_EN** Interrupt enable of ISR [5]
4	RW	**TPKT2E_EN** Interrupt enable of ISR [4]
3	RW	**RPKT_LOST_EN** Interrupt enable of ISR [3]
2	RW	**RXBUF_UNAVA_EN** Interrupt enable of ISR [2]
1	RW	**RPKT2F_EN** Interrupt enable of ISR [1]
0	RW	**RPKT2B_EN** Interrupt enable of ISR [0]

Offset: 08h		**MAC08: MAC Most Significant Address Register (MAC_MADR)** **Init = 0**
Bit	**R/W**	**Description**
31:16		**Reserved (0)**
15:0	RW	**MAC_MADR** The most significant 2 bytes of MAC address

Offset: 0Ch		**MAC0C: MAC Least Significant Address Register (MAC_LADR)** **Init = 0**
Bit	**R/W**	**Description**
31:0	RW	**MAC_LADR** The least significant 4 bytes of MAC address

Offset: 10h		**MAC10: Multicast Address Hash Table 0 Register (MAHT0)** **Init = 0**
Bit	**R/W**	**Description**
31:0	RW	**MAHT0** Multicast address hash table bytes 3∼0 (Hash table 31:0)

Offset: 14h	MAC14: Multicast Address Hash Table 1 Register (MAHT1)	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**MAHT1** Multicast address hash table bytes 7∼4 (Hash table 63:32)

Offset: 18h	MAC18: Normal Priority Transmit Poll Demand Register (NPTXPD)	Init = 0
Bit	**R/W**	**Description**
31:0	W	**NPTXPD** When writing any value to the register, MAC engine reads the normal priority transmit descriptor, process and checks the TXDMA_OWN (TXDES#0 [31]) bit. If TXDMA_OWN (TXDES#0 [31]) = 1, it will move the transmit buffer data into the TX FIFO. The read value of the register is always 0.

Offset: 1Ch	MAC1C: Receive Poll Demand Register (RXPD)	Init = 0
Bit	**R/W**	**Description**
31:0	W	**RXPD** When writing any value to the register, MAC engine reads the receive descriptor, process and checks the RXPKT_RDY (RXDES#0 [31]) bit. If RXPKT_RDY (RXDES#0 [31]) = 0, it will move the receive packet data from the RX FIFO into the receiving buffer in the system memory. The read value of the register is always 0.

MAC20: Normal Priority Transmit Ring Base Address Register (NPTXR_BADR)		
Offset: 20h		Init = 0
Bit	**Attr.**	**Description**
31:28		**Reserved (0)**
27:4	RW	**NPTXR_BADR: Base address of the normal priority transmit ring [27:4]** The base address must be 16 byte aligned
3 :0		**Reserved (0)**

Offset: 24h	MAC24: Receive Ring Base Address Register (RXR_BADR)	Init = 0
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:4	RW	**RXR_BADR: Base address of the receive ring [27:4]** The base address must be 16 byte aligned
3 :0		**Reserved (0)**

Offset: 28h		MAC28: High Priority Transmit Poll Demand Register (HPTXPD) Init = 0
Bit	**R/W**	**Description**
31:0	W	**HPTXPD** When writing any value to the register, MAC engine reads the high priority transmit descriptor, process and check the TXDMA_OWN (TXDES#0 [31]) bit. if TXDMA_OWN (TXDES#0 [31]) = 1, it will move the transmit buffer data into the TX FIFO. The read value of the register is always 0

MAC2C: High Priority Transmit Ring Base Address Register (HPTXR_BADR)		
Offset: 2Ch		**Init = 0**
Bit	**Attr.**	**Description**
31:28		**Reserved (0)**
27:4	W	**HPTXR_BADR : Base address of the high priority transmit ring [27:4]** The base address must be 16 byte aligned
3 :0		**Reserved (0)**

Offset: 30h		MAC30: Interrupt Timer Control Register (ITC) Init = 0
Bit	**R/W**	**Description**
31:16		**Reserved (0)**
15	RW	**TXINT_TIME_SEL** This field defines the period of TX cycle time. When set, TX cycle times are MODE: Value 1000 Mbps: 16.384 us 100 Mbps: 81.92 us 10 Mbps: 819.2 us When cleared, TX cycle times are MODE: Value 1000 Mbps: 1.024 us 100 Mbps: 5.12 us 10 Mbps: 51.2 us
14:12	RW	**TXINT_THR** This field defines the maximum number of transmit interrupts that can be pending before an interrupt is generated. When TXINT_THR != 0, MAC engine issues a transmit interrupt if the transmit packet number transmitted by MAC engine reaches TXINT_THR. When TXINT_THR = 0 and TXINT_CNT = 0, issuing a transmit interrupt or not depends on TXIC in TXDES#1.

to next page

from previous page

11:8	RW	**TXINT_CNT** This field defines the maximum wait time to issue transmit interrupt after a packet has been transmitted by MAC engine. The time unit is 1 TX cycle time. When TXINT_CNT = 0, the function would be disabled. When TXINT_THR = 0 and TXINT_CNT = 0, issuing a transmit interrupt or not depends on TXIC in TXDES#1.
7	RW	**RXINT_TIME_SEL** This field defines the period of RX cycle time. When set, RX cycle times are <table><tr><th>MODE</th><th>Value</th></tr><tr><td>1000 Mbps</td><td>16.384 us</td></tr><tr><td>100 Mbps</td><td>81.92 us</td></tr><tr><td>10 Mbps</td><td>819.2 us</td></tr></table>When cleared, RX cycle times are <table><tr><th>MODE</th><th>Value</th></tr><tr><td>1000 Mbps</td><td>1.024 us</td></tr><tr><td>100 Mbps</td><td>5.12 us</td></tr><tr><td>10 Mbps</td><td>51.2 us</td></tr></table>
6 :4	RW	**RXINT_THR** This field defines the maximum number of receive interrupts that can be pending before an interrupt is generated. When RXINT_THR != 0, MAC engine issues a receive interrupt if the receive packet number received by MAC engine reaches RXINT_THR. If RXINT_THR = 0 and RXINT_CNT = 0, a receive interrupt will be issued when MAC engine finishes receiving a receive packet.
3 :0	RW	**RXINT_CNT** This field defines the maximum wait time to issue receive interrupt after a packet has been received by MAC engine. The time unit is 1 RX cycle time. When RXINT_CNT = 0, the function is disabled. If RXINT_THR = 0 and RXINT_CNT = 0, a receive interrupt is issued when a packet is received by MAC engine.

to next page

from previous page

Note :
Recommended value = 0000_1010h

The Interrupt Timer Control Register allows the software driver to reduce the number of transmit interrupt (ISR[4]) and receive interrupt (ISR[0]) by setting the register. This can lower CPU utilization for handling a large number of interrupts.

The register defines two threshold values for the receive packet number and transmit packet number, and two associated timers.
The threshold value defines the maximum number of receive or transmit interrupts that can be pending before an interrupt is generated.
The timer defines the maximum wait time to issue transmit/receive interrupt after a packet has been transmitted/received by MAC engine.

The threshold value and timer combination allows for batching of several packets into a single interrupt with a limit for how long it can be pending.
The combination prevents throughput from being impeded in heavy traffic, and the time limit prevents resources from being held for too long in low traffic.

The mitigation mechanism is similar for both receive and transmit interrupts.
There is a counter (TXPKT_CNT) in MAC engine to count the packets transmitted by MAC engine.
When the counter reaches TXINT_THR and TXINT_THR != 0, MAC engine issues transmit interrupt.

There is also a counter (RXPKT_CNT) in MAC engine to count the packets received by MAC engine.
When the counter reaches RXINT_THR and RXINT_THR != 0, MAC engine issues receive interrupt.

TXPKT_CNT is cleared when transmit interrupt is issued.
RXPKT_CNT is cleared when receive interrupt is issued.

† The following is the condition for MAC engine to issue a transmit interrupt.

TXINT_THR	TXINT_CNT	MAC engine Action
0	0	1. Issues transmit interrupt after a packet is transmitted and TXIC of the packet is set. 2. Clears TXPKT_CNT.
0	1	1. Issues transmit interrupt after a packet is transmitted and timer reaches the value of TXINT_CNT. 2. Clears TXPKT_CNT.
1	0	1. Issues transmit interrupt if TXPKT_CNT = TXINT_THR. 2. Clears TXPKT_CNT.
1	1	1. Issues transmit interrupt if the following condition holds: * TXPKT_CNT = TXINT_THR * TXPKT_CNT = 1 and timer reaches the value of TXINT_CNT 2. Clears TXPKT_CNT.

† The following is the condition for MAC engine to issue a receive interrupt.

RXINT_THR	RXINT_CNT	MAC engine Action
0	0	1. Issues receive interrupt after a packet is received by MAC engine. 2. Clears RXPKT_CNT.
0	1	1. Issues receive interrupt after a packet is received by MAC engine and timer reaches the value of RXINT_CNT. 2. Clears RXPKT_CNT.
1	0	1. Issues receive interrupt if RXPKT_CNT = RXINT_THR. 2. Clears TXPKT_CNT.
1	1	1. Issues receive interrupt if the following condition holds: * RXPKT_CNT = RXINT_THR * RXPKT_CNT = 1 and timer reaches the value of RXINT_CNT 2. Clears RXPKT_CNT.

Offset: 34h MAC34: Automatic Polling Timer Control Register (APTC) Init = 0

Bit	R/W	Description
31:13		**Reserved (0)**
12	RW	**TXPOLL_TIME_SEL** This field defines the period of TX poll time. When set, TX poll times are MODE / Value: 1000 Mbps: 16.384 us 100 Mbps: 81.92 us 10 Mbps: 819.2 us When cleared, TX poll times are MODE / Value: 1000 Mbps: 1.024 us 100 Mbps: 5.12 us 10 Mbps: 51.2 us
11:8	RW	**TXPOLL_CNT** This field defines the period of transmit automatic poll time. The unit is 1 TX poll time. When TXPOLL_CNT != 0, MAC engine polls the transmit descriptor automatically. If TXPOLL_CNT = 0, MAC engine does not poll the transmit descriptor automatically.
7 :5		**Reserved (0)**

to next page

from previous page

4	RW	**RXPOLL_TIME_SEL** This field defines the period of RX poll time. When set, RX poll times are <table><tr><th>MODE</th><th>Value</th></tr><tr><td>1000 Mbps</td><td>16.384 us</td></tr><tr><td>100 Mbps</td><td>81.92 us</td></tr><tr><td>10 Mbps</td><td>819.2 us</td></tr></table>When cleared, RX poll times are <table><tr><th>MODE</th><th>Value</th></tr><tr><td>1000 Mbps</td><td>1.024 us</td></tr><tr><td>100 Mbps</td><td>5.12 us</td></tr><tr><td>10 Mbps</td><td>51.2 us</td></tr></table>
3 :0	RW	**RXPOLL_CNT** This field defines the period of receive automatic poll time. The unit is 1 RX poll time. When RXPOLL_CNT != 0, MAC engine polls the receive descriptor automatically. If RXPOLL_CNT = 0, MAC engine does not poll the receive descriptor automatically.

Note :
Recommended value = 0000_0001h

The Automatic Polling Timer Control Register allows MAC engine to automatically poll the descriptors. This could lower CPU utilization.

If the transmit automatic poll function is enabled, MAC engine automatically polls the transmit descriptor when the transmit automatic poll timer expires.
If the function is disabled, software needs to write Transmit Poll Demand Register (MAC18) to trigger MAC engine to read transmit descriptors after software has prepared the transmit packets in transmit buffers.

If the receive automatic poll function is enabled, MAC engine automatically polls the receive descriptor when the receive automatic poll timer expires.
If the function is disabled, software needs to write Receive Poll Demand Register (MAC1C) to trigger MAC engine to read receive descriptors after software has released the receive descriptors to MAC engine.

Offset: 38h MAC38: DMA Burst Length and Arbitration Control Register (DBLAC) Init = 0002_2F00h

Bit	R/W	Description
31:24		**Reserved (0)**
23	RW	**IFG_INC: IFG(InterFrame Gap) increase** The field defines the increase or decrease of IFG in Ethernet. When IFG_INC=1'b1, the IFG would increase. When IFG_INC=1'b0, the IFG would decrease.

to next page

from previous page

22:20	RW	**IFG_CNT: IFG(InterFrame Gap) count** The field defines the increase or decrease number of IFG in Ethernet. When IFG_INC=1'b1, the IFG would increase. When IFG_INC=1'b0, the IFG would decrease. The unit is 1 transmit clock in Ethernet. (8 ns in 1000 Mbps mode, 40ns in 100 Mbps mode, 400 ns in 10 Mbps mode). When in 1000 Mbps mode, if IFG_INC= 1'b0, the value in the field should not be set more than 2. For example: When IFG_CNT=3'h1 and IFG_INC=1'b1 in 1000 Mbps mode, then the IFG = 96+1x8 = 104 ns. When IFG_CNT=3'h1 and IFG_INC=1'b0 in 1000 Mbps mode, then the IFG = 96-1x8 = 88 ns.
19:16	RW	**TXDES_SIZE: Transmit descriptor size** This field defines the transmit descriptor size. Writing 0 to this field is illegal. The unit is 8 bytes.
15:12	RW	**RXDES_SIZE: Receive descriptor size** This field defines the receive descriptor size. Writing 0 to this field is illegal. The unit is 8 bytes.
11:10	RW	**TXBST_SIZE: TXDMA maximum burst size per TXDMA burst** This field sets the maximum size of TXDMA burst. The burst sizes are as follows: * 00: 64 bytes * 01: 128 bytes * 10: 256 bytes * 11: 512 bytes
9 :8	RW	**RXBST_SIZE: RXDMA maximum burst size per RXDMA burst** This field sets the maximum size of RXDMA burst. The burst sizes are as follows: 00: 64 bytes 01: 128 bytes 10: 256 bytes 11: 512 bytes
7		**Reserved (0)**
6	RW	**RX_THR_EN: Enable RX FIFO threshold arbitration**

to next page

from previous page

5 :3	RW	**RXFIFO_HTHR: RX FIFO high threshold value for arbitration** When the used space in the RX FIFO is larger than or equal to the RX FIFO high threshold value, the RXDMA has higher priority over the TXDMA for using the DMA channel. The RXDMA keeps the higher priority until the used space in the RX FIFO is less than or equal to the RX FIFO low threshold value. Then the TXDMA gets higher priority over the RXDMA. So software must set RXFIFO_HTHR larger than RXFIFO_LTHR to keep MAC engine work correctly. 000: Threshold = 0 001: Threshold = 1/8 space of RX FIFO 010: Threshold = 2/8 space of RX FIFO 011: Threshold = 3/8 space of RX FIFO 100: Threshold = 4/8 space of RX FIFO 101: Threshold = 5/8 space of RX FIFO 110: Threshold = 6/8 space of RX FIFO 111: Threshold = 7/8 space of RX FIFO
2 :0	RW	**RXFIFO_LTHR: RX FIFO low threshold value for arbitration** When the used space in the RX FIFO is less than or equal to the RX FIFO low threshold value, the TXDMA has higher priority over the RXDMA for using the DMA channel. 000: Threshold = 0 001: Threshold = 1/8 space of RX FIFO 010: Threshold = 2/8 space of RX FIFO 011: Threshold = 3/8 space of RX FIFO 100: Threshold = 4/8 space of RX FIFO 101: Threshold = 5/8 space of RX FIFO 110: Threshold = 6/8 space of RX FIFO 111: Threshold = 7/8 space of RX FIFO
Note : Recommended value = 0002_2F72h		

Offset: 3Ch — MAC3C: DMA/FIFO State Register (DMAFIFOS) — Init = 0C00_0000h

Bit	R/W	Description
31	R	**TXD_REQ: TXDMA request** **(for debugging purpose only)**
30	R	**RXD_REQ: RXDMA request** **(for debugging purpose only)**
29	R	**DARB_TXGNT: TXDMA grant** **(for debugging purpose only)**
28	R	**DARB_RXGNT: RXDMA grant** **(for debugging purpose only)**
27	R	**TXFIFO_EMPTY: TX FIFO is empty** **(for debugging purpose only)**
26	R	**RXFIFO_EMPTY: RX FIFO is empty** **(for debugging purpose only)**
25:22		**Reserved (0)**
21:18	R	**TXDMA3_SM: TXDMA 3 state machine** **(for debugging purpose only)** The state machine is in charge of the read data flow from TX FIFO to TX pre-buffer.
17:16	R	**TXDMA2_SM: TXDMA 2 state machine** **(for debugging purpose only)** The state machine is in charge of the burst read/write of transmit descriptor and buffer.

to next page

from previous page

15:12	R	**TXDMA1_SM: TXDMA 1 state machine** **(for debugging purpose only)** The state machine is in charge of the read/write of transmit descriptor and buffer.
11		**Reserved (0)**
10:8	R	**RXDMA3_SM: RXDMA 3 state machine** **(for debugging purpose only)** The state machine is in charge of RXDMA PVCI interface read/write.
7 :4	R	**RXDMA2_SM: RXDMA 2 state machine** **(for debugging purpose only)** The state machine is in charge of the burst read/write of receive descriptor and buffer.
3 :0	R	**RXDMA1_SM: RXDMA 1 state machine** **(for debugging purpose only)** The state machine is in charge of the read/write of receive descriptor and buffer.

Offset: 44h **MAC44: Feature Register (FEAR)** **Init = 0**

Bit	R/W	Description
31:6		**Reserved (0)**
5 :3	R	**TFIFO_RSIZE: TX FIFO Real Size** The FIFO sizes are as follows: 000: 2K (default) 001: 4K **(Invalid)** 010: 8K **(Invalid)** 011: 16K **(Invalid)** 100: 32K **(Invalid)** 111∼3'b101: Reserved
2 :0	R	**RFIFO_RSIZE: RX FIFO Real Size** The FIFO sizes are as follows: 000: 2K (default) 001: 4K **(Invalid)** 010: 8K **(Invalid)** 011: 16K **(Invalid)** 100: 32K **(Invalid)** 111∼3'b101: Reserved

MAC48: Transmit Priority Arbitration and FIFO Control Register (TPAFCR)

Offset: 48h **Init = 0000_00F1**

Bit	Attr.	Description
31:30		**Reserved (0)**
29:27	RW	**TFIFO_SIZE: TX FIFO Size** Software can program this field to decide the TX FIFO size. Before programming this field, software must read the actual TX FIFO size in the FEATURE register. It is not allowed to program a value larger than the actual TX FIFO size. The FIFO sizes are as follows: 000: 2K 001: 4K **(Invalid)** 010: 8K **(Invalid)** 011: 16K **(Invalid)** 100: 32K **(Invalid)** 111∼3'b101: Reserved

to next page

from previous page

26:24	RW	**RFIFO_SIZE: RX FIFO Size** Software can program this field to decide the RX FIFO size. Before programming this field, software must read the actual RX FIFO size in the FEATURE register. It is not allowed to program a value larger than the actual RX FIFO size. The FIFO sizes are as follows: 000: 2K 001: 4K **(Invalid)** 010: 8K **(Invalid)** 011: 16K **(Invalid)** 100: 32K **(Invalid)** 111∼3'b101: Reserved
23:16	RW	**EARLY_TXTHR: Early Transmit Threshold** This field specifies the threshold level in the TX FIFO to begin transmission. When the byte count of the data in the Tx FIFO reaches the threshold or there is at least one packet in TX FIFO, hardware would begin to transmit the packet to network. Writing 0 to this field indicates that hardware should begin to transmit the packet after one whole packet has been saved in TX FIFO. The value software programs in this field should be less than TX FIFO size. The unit is 64 bytes.
15:8	RW	**EARLY_RXTHR: Early Receive Threshold** This field specifies the threshold level in the RX FIFO to move packet data to system memory. When the byte count of the data in the RX FIFO reaches the threshold or there is at least one packet in RX FIFO, hardware would begin to move the packet from RX FIFO to system memory. Writing 0 to this field indicates that hardware should begin to move the packet after one whole packet has been stored in RX FIFO. The value software programs in this field should be less than RX FIFO size. The unit is 64 bytes.
7 :4	RW	**HPKT_THR: High Priority Transmit Packet Threshold** When the packet number TXDMA moves from the high priority transmit ring to TX FIFO is **less** than the threshold and the high priority packet is still available, TXDMA would switch to service the high priority transmit ring if TXDMA is servicing the normal priority transmit ring. But if TXDMA is servicing the high priority transmit ring at that time, it would continue to service the high priority transmit ring. When the packet number TXDMA moves from the high priority transmit ring to TX FIFO is **equal to** or **greater** than the threshold and the normal priority packet is still available, TXDMA would switch to service the normal priority transmit ring. The hardware behavior is the same whether writing 0 or 1 to this field.

to next page

from previous page

3 :0	RW	**NPKT_THR: Normal Priority Transmit Packet Threshold** Under the following conditions, TXDMA would switch to service the high priority transmit ring from the normal priority transmit ring: 1. When the high priority packet number is **less** than the HPKT_THR and high priority packet is available, TXDMA would switch to service the high priority transmit ring after it finishes servicing a normal priority transmit packet. 2. When the packet number TXDMA moves from the normal priority transmit ring to TX FIFO is **equal to** or **greater** than the threshold and the high priority packet number is available, TXDMA would switch to service the high priority transmit ring. 3. When the high priority packet is available and the normal priority packet is not available, TXDMA would switch to service the high priority transmit ring. The hardware behavior is the same whether writing 0 or 1 to this field.

Offset: 4Ch — MAC4C: Receive Buffer Size Register (RBSR) — Init = 0000_0640h

Bit	R/W	Description
31:14		**Reserved (0)**
13:3	RW	**RXBUF_SIZE: Receive buffer size [13:3]** The unit is 1 byte. Receive buffer size must be 8-byte alignment.
2 :0		**Reserved (0)**

Offset: 50h — MAC50: MAC Control Register (MACCR) — Init = 0

Bit	R/W	Description
31	RW	**SW_RST: Software reset** Writing 1 to this bit enables software reset. Software reset would last **175** AHB bus clocks, and then be auto-cleared.
30:20		**Reserved (0)**
19	RW	**SPEED_100: Speed mode** 1: 100 Mbps 0: 10 Mbps The field and GMAC_MODE (Bit 9) are used to determine MAC engine speed mode. <table><tr><th>GMAC_MODE</th><th>SPEED_100</th><th>Function</th></tr><tr><td>0</td><td>1</td><td>100 Mbps mode</td></tr><tr><td>0</td><td>0</td><td>10 Mbps mode</td></tr><tr><td>1</td><td>0</td><td>1000 Mbps mode</td></tr><tr><td>1</td><td>1</td><td>1000 Mbps mode</td></tr></table>This field cannot be software reset.
18	RW	**DISCARD_CRCERR** Discard the CRC error packet if there is CRC error status in the transmit packet.
17	RW	**RX_BROADPKT_EN** Receive broadcast packets.
16	RW	**RX_MULTIPKT_EN** Receive all multicast packets.

to next page

from previous page

15	RW	**RX_HT_EN** Enable storing incoming packet if the packet passes hash table address filtering and is a multicast packet.
14	RW	**RX_ALLADR** Destination address of incoming packet not checked
13	RW	**JUMBO_LF: Jumbo Long Frame** When set, packets with length more than **9216** (**9220** for packets with VLAN tag) are treated as long frames. When cleared, packets with length more than **1518** (**1522** for packets with VLAN tag) are treated as long frames.
12	RW	**RX_RUNT** Receive the incoming packet even if its length is less than **64** bytes. The incoming packet length must be longer than or equal to **10** bytes.
11		**Reserved (0)**
10	RW	**CRC_APD: Append CRC to transmitted packets**
9	RW	**GMAC_MODE: GMAC mode** If GMAC_MODE = 1, MAC engine is in 1000 Mbps mode; otherwise, MAC engine is in 10/100 Mbps mode. This field cannot be software reset.
8	RW	**FULLDUP: Full duplex** If FULLDUP = 1, MAC engine is in full duplex mode; otherwise, MAC engine is in half duplex mode.
7	RW	**ENRX_IN_HALFTX** Enable packet reception when transmitting packets in half duplex mode.
6	RW	**PHY link status detection** 1: Rising and falling edge trigger 0: High-level sensitive
5	RW	**HPTXR_EN: High priority transmit ring enable** If HPTXR_EN = 1, software can use the high priority transmit ring; otherwise, software cannot use the high priority transmit ring.
4	RW	**REMOVE_VLAN** Remove VLAN tag from packets received with VLAN tag.
3	RW	**RXMAC_EN: RXMAC enable** When set, enable RXMAC to receive packets.
2	RW	**TXMAC_EN: TXMAC enable** When set, enable TXMAC to transmit packets.
1	RW	**RXDMA_EN: Enable receive DMA channel** If this bit is zero, reception is stopped immediately.
0	RW	**TXDMA_EN: Enable transmit DMA channel** If this bit is zero, transmission is stopped immediately.

Offset: 54h — MAC54: MAC Status Register (MACSR) — Init = 0

Bit	R/W	Description
31:12		**Reserved (0)**
11	RW	**COL_EXCEED: Collision amount exceeds 16** Writing ”1” to this bit will clear this status flag.

to next page

from previous page

10	RW	**LATE_COL: Transmitter detects late collision** Writing "1" to this bit will clear this status flag.
9	RW	**TPKT_LOST** Packets transmitted to Ethernet lost due to late collision or excessive collision. Writing "1" to this bit will clear this status flag.
8	RW	**TPKT_OK: Packets transmitted to Ethernet successfully** Writing "1" to this bit will clear this status flag.
7	RW	**RUNT: Receiver detects a runt packet** Writing "1" to this bit will clear this status flag.
6	RW	**FTL: Receiver detects a frame that is too long** Writing "1" to this bit will clear this status flag.
5	RW	**CRC_ERR** Incoming packets CRC check result is invalid, unless the CRC_DIS bit is set. Writing "1" to this bit will clear this status flag.
4	RW	**RPKT_LOST: Received packets lost due to RX FIFO ful** Writing "1" to this bit will clear this status flag.
3	RW	**RPKT_SAVE: Packets received to RX FIFO successfully** Writing "1" to this bit will clear this status flag.
2	RW	**COL: Incoming packet dropped due to collision** Writing "1" to this bit will clear this status flag.
1	RW	**BROADCAST: Incoming packet for broadcast address** Writing "1" to this bit will clear this status flag.
0	RW	**MULTICAST: Incoming packet for multicast address** Writing "1" to this bit will clear this status flag.

Offset: 58h — MAC58: Test Mode Register (TM) — Init = 0

Bit	R/W	Description
31:21		**Reserved (0)**
20	RW	**PTIMER_TEST: Automatic polling timer test mode**
19	RW	**ITIMER_TEST: Interrupt timer test mode**
18:16		**Reserved (0)**
15	RW	**TEST_COL: Transmit collision test mode**
14:5	RW	**TEST_BKOFF: Backoff value in transmission collision test mode**
4 :0	RW	**TEST_EXSTHR: Retry upper limit in transmit collision test mode**

Offset: 60h — MAC60: PHY Control Register (PHYCR) — Init = 0000_0034h

Bit	R/W	Description
31:28		**Reserved (0)**
27	RW	**MIIWR** Setting this bit to 1 initializes a write sequence to PHY. This bit would be auto cleared after the write operation is finished.
26	RW	**MIIRD** Setting this bit to 1 initializes a read sequence to PHY. This bit would be auto cleared after the read operation is finished.
25:21	RW	**REGAD: PHY register address**

to next page

from previous page

20:16	RW	**PHYAD: PHY address**
15:6		**Reserved (0)**
5 :0	RW	**MDC_CYCTHR: MDC cycle threshold** This field defines the period of MDC. The MDC period = MDC_CYCTHR x RX clock period. When first reading/writing PHY register, or PHY link status change, software must set these two bits as 6'b34. After identifying the mode used, software may set the field to the corresponding mode. The allowable values are: MODE / Value 1000 Mbps: 33h ∼ 3Fh 100 Mbps: 0Bh ∼ 3Fh 10 Mbps: 02h ∼ 3Fh

Offset: 64h — MAC64: PHY Data Register (PHYDATA) — Init = 0

Bit	R/W	Description
31:16	R	**MIIRDATA: Read data from PHY**
15:0	RW	**MIIWDATA: Write data to PHY**

Offset: 68h — MAC68: Flow Control Register (FCR) — Init = 0000_0400h

Bit	R/W	Description
31:16	RW	**PAUSE_TIME: Pause time in pause frame** The unit is 1 slot time.
15:9	RW	**FC_HIGH/FC_LOW** **RX FIFO free space high threshold:** A pause frame is sent with pause time = 0 when RX FIFO free space is larger than the high threshold. The unit is 256 bytes, and the default value is 7'h5. **RX FIFO free space low threshold:** A pause frame is sent with pause time set in bits 31∼16 when RX FIFO free space is lower than the low threshold. The unit is 256 bytes, and the default value is 7'h2. When FC_HTHR_SEL = 1, RX FIFO free space high threshold is selected. When FC_HTHR_SEL = 0, RX FIFO free space low threshold is selected. The value software programs in this field should be less than RX FIFO size.
8	RW	**FC_HTHR_SEL: RX FIFO free space high threshold select** When set, RX FIFO free space high threshold is selected. When cleared, RX FIFO free space low threshold is selected.
7 :5		**Reserved (0)**
4	RW	**RX_PAUSE: Receive pause frame (Writing "1" to clear)**
3	R	**TXPAUSED** Packet transmission paused due to receive pause frame
2	RW	**FCTHR_EN: Enable flow control threshold mode** This bit enables transmit pause frame for high/low threshold.

to next page

from previous page

1	RW	**TX_PAUSE: Transmit pause frame** Software can set this bit to send pause frames. This bit is auto-cleared after the pause frame has been transmitted.
0	RW	**FC_EN: Flow control mode enable**

Offset: 6Ch		MAC6C: Back Pressure Register (BPR) Init = 0000_0200h
Bit	**R/W**	**Description**
31:15		**Reserved (0)**
14:8	RW	**BK_LOW: RX FIFO free space low threshold** MAC generates a jam pattern if RX FIFO free space is lower than the low threshold when packets are incoming. The unit is 256 bytes, and the default value is 7'h2.
7 :4	RW	**BKJAM_LEN: Back pressure jam length** 0000: 4 bytes 0001: 8 bytes 0010: 16 bytes 0011: 32 bytes 0100: 64 bytes 0101: 128 bytes 0110: 256 bytes 0111: 512 bytes 1000: 1024 bytes 1001: 1518 bytes 1010: 2048 bytes 1011 ~ 1111: 4 bytes
3 :2		**Reserved (0)**
1	RW	**BKADR_MODE: Back pressure address mode** 1: Generate jam pattern when packet address matches. 0: Generate jam pattern when any packet is incoming.
0	RW	**BK_EN: Back pressure mode enable**

Offset: 70h		MAC70: Power Control Register (PWRTC) Init = 0
Bit	**R/W**	**Description**
31:19		**Reserved (0)**
18	RW	**SW_PDNPHY: Software power down PHY**
17:16		**Reserved (0)**
15	RW	**PWRSAV: Power saving mode** This field is used to determine the current power state. When set, MAC engine enters power saving mode. When cleared, MAC engine is in normal mode.
14:0		**Reserved (0)**

MAC90: Normal Priority Transmit Ring Pointer Register (NPTXR_PTR)		
Offset: 90h		**Init = X**
Bit	**Attr.**	**Description**
31:0	R	**NPTXR_PTR: Normal Priority Transmit Ring Pointer Register** **(for debugging purpose only)** This field indicates the current value of the transmit descriptor pointer for the normal priority transmit ring pointer register.

MAC94: High Priority Transmit Ring Pointer Register (HPTXR_PTR)		
Offset: 94h		**Init = X**
Bit	**Attr.**	**Description**
31:0	R	**HPTXR_PTR: High Priority Transmit Ring Pointer Register** **(for debugging purpose only)** This field indicates the current value of the transmit descriptor pointer for the high priority transmit ring pointer register.

Offset: 98h	MAC98: Receive Ring Pointer Register (HPTXR_PTR)	Init = X
Bit	**R/W**	**Description**
31:0	R	**RXR_PTR: Receive Ring Pointer Register** **(for debugging purpose only)** This field indicates the current value of the transmit descriptor pointer for the receive ring pointer register.

Offset: A0h	MACA0: TPKT_CNT Counter Register	Init = 0
Bit	**R/W**	**Description**
31:0	R	**TPKT_CNT** **(for debugging purpose only)** Counter for counting packets transmitted successfully

Offset: A4h	MACA4: TXMCOL_CNT and TXSCOL_CNT Counter Register	Init = 0
Bit	**R/W**	**Description**
31:16	R	**TXMCOL_CNT** **(for debugging purpose only)** Counter for counting packets transmitted OK with 2∼15 collisions
15:0	R	**TXSCOL_CNT** **(for debugging purpose only)** Counter for counting packets transmitted OK with single collision

Offset: A8h	MACA8: TXECOL_CNT and TXFAIL_CNT Counter Register	Init = 0
Bit	**R/W**	**Description**
31:16	R	**TXFAIL_CNT** **(for debugging purpose only)** Counter for counting packets failed in transmission (due to late collision or collision count >= 16 or transmit underrun)
15:0	R	**TXECOL_CNT** **(for debugging purpose only)** Counter for counting packets failed in transmission (due to collision count >= 16)

Offset: ACh	MACAC: RUNT_CNT and TXLCOL_CNT Counter Register	Init = 0
Bit	**R/W**	**Description**
31:16	R	**TXUNDERUN_CNT** **(for debugging purpose only)** Counter for counting packets failed in transmission (due to transmit underrun)

to next page

from previous page

15:0	R	**TXLCOL_CNT** **(for debugging purpose only)** Counter for counting packets failed in transmission (due to late collision)

Offset: B0h	**MACB0: RPKT_CNT Counter Register**	**Init = 0**
Bit	**R/W**	**Description**
31:0	R	**RPKT_CNT** **(for debugging purpose only)** Counter for counting packets received successfully

Offset: B4h	**MACB4: BROPKT_CNT Counter Register**	**Init = 0**
Bit	**R/W**	**Description**
31:0	R	**BROPKT_CNT** **(for debugging purpose only)** Counter for counting received broadcast packets

Offset: B8h	**MACB8: MULPKT_CNT Counter Register**	**Init = 0**
Bit	**R/W**	**Description**
31:0	R	**MULPKT_CNT** **(for debugging purpose only)** Counter for counting received multicast packets

Offset: BCh	**MACBC: RPF_CNT and AEP_CNT Counter Register**	**Init = 0**
Bit	**R/W**	**Description**
31:16	R	**RPF_CNT** **(for debugging purpose only)** Receive pause frame counter
15:0	R	**AEP_CNT** **(for debugging purpose only)** Counter for counting packets with alignment error The counter is to count packets with CRC error and no octet-boundary discarded by MAC engine.

Offset: C0h	**MACC0: RUNT_CNT Counter Register**	**Init = 0**
Bit	**R/W**	**Description**
31:16		**Reserved (0)**
15:0	R	**RUNT_CNT** **(for debugging purpose only)** Counter for counting received runt packets

Offset: C4h	**MACC4: CRCER_CNT and FTL_CNT Counter Register**	**Init = 0**
Bit	**R/W**	**Description**
31:16	R	**CRCER_CNT: CRC error packet counter** **(for debugging purpose only)** The counter counts the number of octet-boundary frames discarded due to CRC error.
15:0	R	**FTL_CNT** **(for debugging purpose only)** Counter for counting received FTL packets

Offset: C8h	**MACC8: RCOL_CNT and RLOST_CNT Counter Register**	**Init = 0**
Bit	**R/W**	**Description**
31:16	R	**RLOST_CNT** **(for debugging purpose only)** Counter for counting loss of received packets (due to RX FIFO full)

to next page

from previous page

15:0	R	**RCOL_CNT** **(for debugging purpose only)** Receive collision counter

14.4 Function Description

14.4.1 Transmit Descriptor

MAC engine uses a descriptor ring to manage transmit buffers. The transmit descriptors and data buffers are all located in system memory. MAC engine moves the transmit packet data from the transmit buffers in system memory to the TX FIFO inside MAC engine and then transmits the packet to Ethernet. The transmit descriptors reside in system memory act as pointers to the transmit buffers.

Each transmit descriptor contains a transmit buffer. A transmit buffer consists of either an entire frame or part of a frame, but not multiple frames. The transmit descriptor contains transmit buffer status and the transmit buffer can only contain the transmit data. MAC engine supports two descriptor rings for transmission. These descriptor rings are normal priority transmit ring and high priority transmit ring. The normal priority transmit ring is for normal packet transmission; the high priority transmit ring is for high priority packet transmission. Higher priority packets can be put into the high priority transmit ring for quicker transmission.

- The start address of each transmit descriptor must be 16-byte aligned.
- The maximum transmit packet size including CRC is **9216** bytes (**9220** bytes if VLAN tag is inserted).
- MAC engine only supports IPV4 checksum offload. So software must be certain the transmit packet is an IPV4 packet when software request MAC engine to do checksum offload.
- LLC packet is IEEE 802.3/802.2/SNAP format packet.
- MAC engine doesn't support the following two packets to do checksum offload; they are IEEE 802.3 with IEEE 802.2 packet and IEEE 802.3 with 802.1Q and 802.2 packet.

Offset: 00h		**TXDES#0: Control Bits and Ownership Information.** **Init = X**
Bit	**R/W**	**Description**
31		**TXDMA_OWN: TXDMA ownership bit** When set, it indicates that the descriptor is owned by the MAC engine. When reset, it indicates that the software owns the descriptor. MAC engine clears this bit when it completes the frame transmission.
30		**EDOTR: End Descriptor of Transmit Ring** When set, it indicates that the descriptor is the last descriptor of the transmit ring.
29		**FTS: First Transmit Segment descriptor** When set, it indicates that this is the first descriptor of a TX packet.
28		**LTS: Last Transmit Segment descriptor.** When set, it indicates that this is the last descriptor of a TX packet.
27:20		**Reserved (0)**
19		**CRC_ERR: CRC error** When CRC_ERR=1 and DISCARD_CRCERR (MAC50 [18] = 1), TXDMA would discard the transmit packet, not send it to Ethernet.
18:14		**Reserved (0)**
13:0		**TXBUF_SIZE: Transmit buffer size in byte** The transmit buffer size can not be zero.

Offset: 04h	TXDES#1: VLAN Control Bits and VLAN Tag Control Information.	Init = X
Bit	**R/W**	**Description**
31		**TXIC: Transmit Interrupt on Completion** When set, the MAC engine would assert transmit interrupt after the present frame has been transmitted. It is valid only when FTS = 1 and MAC30 [14:8] = 0 (TXINT_THR, TXINT_CNT).
30		**TX2FIC: Transmit to FIFO Interrupt on Completion** When set, the MAC engine would assert transmit interrupt after the present frame has been moved into the TX FIFO. It is valid only when FTS = 1.
29:23		**Reserved (0)**
22		**LLC_PKT: LLC packet** When set, MAC engine would treat the packet as LLC packet. It is valid only when FTS = 1
21:20		**Reserved (0)**
19		**IPCS_EN: IP checksum offload enable** When set, MAC engine would offload IP checksum. It is valid only when FTS = 1
18		**UDPCS_EN: UDP checksum offload enable** When set, MAC engine would offload UDP checksum. It is valid only when FTS = 1
17		**TCPCS_EN: TCP checksum offload enable.** When set, MAC engine would offload TCP checksum. It is valid only when FTS = 1
16		**INS_VLAN: Insert VLAN Tag** When set, 0x8100 (IEEE 802.1Q VLAN Tag Type) is inserted after source address, and 2 bytes VLAN_TAGC are inserted after IEEE 802.1Q VLAN Tag Type. When clear, the packet content would not be changed when transmitting to network. It is valid only when FTS = 1
15:0		**VLAN_TAGC: VLAN Tag Control Information** The 2-byte VLAN Tag Control Information contains information, from the upper layer, of user priority, canonical format indicator, and VLAN ID. Please refer to IEEE 802.1Q for more VLAN tag information. <table><tr><th>Bits</th><th>Function</th></tr><tr><td>15 - 13</td><td>User priority</td></tr><tr><td>12</td><td>CFI (Canonical Format Indicator)</td></tr><tr><td>11 - 0</td><td>VID (VLAN Identifier)</td></tr></table>It is valid only when FTS = 1

Offset: 08h	TXDES#2: Reserved	Init = X
Bit	**R/W**	**Description**
31:0		**Reserved (0)**

Offset: 0Ch		TXDES#3: Transmit Buffer Base Address Init = X
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:1		**TXBUF_BADR: Transmit buffer base address [27:1]** Transmit buffer base address must be at least 2-byte alignment. This means TXBUF_BADR[0] must be zero.
0		**Reserved (0)**

14.4.2 Receive Descriptor

MAC engine uses a descriptor ring to manage the receive buffers. The receive descriptors and data buffers are all located in system memory. MAC engine first stores the packet received from the network in the RX FIFO and then moves the received packet data to the receive buffers in system memory. The receive descriptors reside in system memory act as pointers to the receive buffers.

There is a descriptor ring for reception. The base address of the receive ring is in the Receive Ring Base Address Register (RXR_BADR, MAC24). Each receive descriptor contains a receive buffer. A receive buffer consists of either an entire frame or part of a frame, but not multiple frames. The receive descriptor contains receive buffer status and the receive buffer can only contain the receive packet data.

MAC engine supports the receive buffer base address as 2-byte alignment for the zero-copy feature. But there is a limitation when software program the receive buffer base address as 2-byte alignment. The limitation is the receive packet can only occupy one receive buffer. This means that the receive buffer size must be greater than the receive packet length. For example, if the length of the incoming packet is always less than 1600 bytes, software can program the receive buffer size as 1600 bytes. Then the limitation sustained when the receive buffer base address is 2-byte alignment. If software program the receive buffer base address as 8-byte alignment, then the limitation does not hold.

- The start address of each receive descriptor must be 16-byte aligned.
- The maximum receive packet size is **9216** bytes (**9220** bytes for packets with VLAN tag).
- MAC engine only supports IPV4 checksum offload. So if the incoming packet is not an IPV4 packet, MAC engine would not do checksum verification. The IPCS_FAIL, UDPCS_FAIL, TCPCS_FAIL field is always zero.
- LLC packet is IEEE 802.3/802.2/SNAP format packet.
- MAC engine doesn't support the following tow packets to do checksum offload; they are IEEE 802.3 with IEEE 802.2 packet and IEEE 802.3 with 802.1Q and 802.2 packet.
- The receive buffer size must be greater than the receive packet length when software programs the receive buffer base address as 2-byte alignment.

Offset: 00h		RXDES#0: Frame Status and Descriptor Ownership Information. Init = X
Bit	**R/W**	**Description**
31		**RXPKT_RDY: RX packet ready** When clear, it indicates that the descriptor is owned by the MAC engine. When set, it indicates that the descriptor is owned by the software. MAC engine set this bit when it completes the frame reception or when the receive buffer of the receive descriptor is full.
30		**EDORR: End Descriptor of Receive Ring** When set, it indicates that the descriptor is the last descriptor of the receive ring.

to next page

from previous page

29		**FRS: First Receive Segment descriptor** When set, it indicates that this is the first descriptor of a received packet. Bits 25 ∼ 0 are valid only when the FRS = 1.
28		**LRS: Last Receive Segment descriptor.** When set, it indicates that this is the last descriptor of a received packet.
27:26		**Reserved (0)**
25		**PAUSE_FRMAE: PAUSE_FRAME Pause frame** When set, it indicates that the receive packet is a pause frame.
24		**PAUSE_OPCODE: Pause frame OP code** When set, it indicates that there is pause frame OP code in the receive packet.
23		**FIFO_FULL: FIFO full** When set, it indicates that RX FIFO is full when the packet is received.
22		**RX_ODD_NB: Receive Odd Nibbles** When set, it indicates receiving a packet with odd nibbles.
21		**RUNT: Runt packet** When set, it indicates that the received packet length is less than 64 bytes.
20		**FTL: Frame Too Long** When set, it indicates that the received packet length exceeds the long frame length. If JUMBO_LF=0, the long frame length is **1518** (**1522** for the receive packet with VLAN tag) bytes. If JUMBO_LF=1, the long frame length is **9216** (**9220** for the receive packet with VLAN tag) bytes.
19		**CRC_ERR: CRC error** When set, it indicates that the CRC error occurs on the received packet.
18		**RX_ERR: Receive error** When set, it indicates that receive error happens when receiving a packet.
17		**BROADCAST: Broadcast frame.** When set, it indicates that the received packet is a broadcast frame.
16		**MULTICAST: Multicast frame.** When set, it indicates that the received packet is a multicast frame.
15:14		**Reserved (0)**
13:0		**VDBC: valid data byte count.** The field indicates the valid data in the receive buffer. The unit is 1 byte.

Offset: 04h	**RXDES#1: VLAN Status Bits and VLAN Tag Control Information.**	**Init = X**
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27		**IPCS_FAIL: IP checksum failure** When set, MAC engine detects IP checksum failure. The field is valid only when the FRS = 1.
26		**UDPCS_FAIL: UDP checksum failure** When set, MAC engine detects UDP checksum failure. The field is valid only when the FRS = 1.

to next page

from previous page

25		**TCPCS_FAIL: TCP checksum failure** When set, MAC engine detects TCP checksum failure. The field is valid only when the FRS = 1.
24		**VLAN_AVA: VLAN Tag Available** When set, the receive packet is a packet with IEEE 802.1Q VLAN Tag Type. The field is valid only when the FRS = 1.
23		**DF: Datagram Fragment** When set, the IP packet is not fragment. When clear, the IP packet may fragment. Checksum status is valid only when DF=1.
22		**LLC_PKT: LLC packet** When set, it indicates that the receive packet is LLC packet. The field is valid only when the FRS = 1.
21:20		**PROTL_TYPE: Protocol Type** These 2 bits indicate which protocol in the receive packet. 00: Not IP protocol. 01: IP protocol. 10: TCP/IP protocol. 11: UDP/IP protocol. The field is valid only when the FRS = 1.
19:16		**Reserved (0)**
15:0		**VLAN_TAGC: VLAN Tag Control Information** If the receive packet contains VLAN tag. MAC engine would extracts 4 bytes from the receive packet. The 4 bytes data contains 0x8100 and 2 bytes VLAN Tag Control Information. MAC engine would move the 2 bytes VLAN Tag Control Information to this field. The 2-byte VLAN Tag Control Information contains information, from the upper layer, of user priority, canonical format indicator, and VLAN ID. Please refer to IEEE 802.1Q for more VLAN tag information. Bits / Function: 15 - 13: User priority 12: CFI (Canonical Format Indicator) 11 - 0: VID (VLAN Identifier) The field is valid only when the FRS = 1.

Offset: 08h	RXDES#2: Reserved	Init = X
Bit	**R/W**	**Description**
31:0		**Reserved (0)**

Offset: 0Ch	RXDES#3: Receive Buffer Base Address	Init = X
Bit	**R/W**	**Description**
31:28		**Reserved (0)**

to next page

from previous page

27:1		**RXBUF_BADR: Receive buffer base address [27:1]** Receive buffer base address must be at least 2-byte alignment. This means RXBUF_BADR[0] must be zero.
0		**Reserved (0)**

14.4.3 Transmitting Packet

When software wants to transmit a packet to the Ethernet, it moves the packet data into the transmit buffer first. Then software writes the packet length and position into the transmit descriptor and triggers MAC engine to send the packet. After the entire packet has been moved into the TX FIFO, MAC engine begins to transmit it to the Ethernet. When the packet has been transmitted, MAC engine asserts interrupt to notify software that the packet has been transmitted successfully. Higher priority packets can be put into the high priority descriptor for quicker transmission.

14.4.4 Receiving Packet

When there is an incoming packet, MAC engine first saves the received packet in the RX FIFO if the address check result is correct. After the incoming packet is successfully saved in RX FIFO, MAC engine initiates Direct Memory Access (DMA) function to move the received packet data from the RX FIFO to the system memory. Then MAC engine asserts interrupt to notify software that the packet has been received successfully.

14.4.5 Ethernet Address Filtering

- ALL: bit 14 of MAC Control Register (MACCR) (offset: 50h)
- MULTI: bit 16 of MAC Control Register (MACCR) (offset: 50h)
- BROAD: bit 17 of MAC Control Register (MACCR) (offset: 50h)
- HT: bit 15 of MAC Control Register (MACCR) (offset: 50h)
- MAC_ADR: MAC Addres Register (MAC_MADR & MAC_LADR) (offset: 08h & 0ch)
- MAHT: Multicast Address Hash Table Register (MAHT0 & MAHT1) (offset: 10h & 14h)
- multicast: Multicast Address
- broadcast: Broadcast Address

Group	ALL	MULTI	BROAD	HT	MAC_ADR	MAHT	multicast	broadcast
A	0	0	0	0	**O**	**X**	**X**	**X**
B	0	0	0	1	**O**	**O**	**O**	**X**
C	0	0	1	0	**O**	**X**	**X**	**O**
D	0	0	1	1	**O**	**O**	**O**	**O**
E	0	1	x	x	**O**	**X**	**O**	**X**
F	1	x	x	x	**O**	**O**	**O**	**O**

”O” : MAC Controller receives a frame whose destination address exactly matches the register/address listed in the column.
”X” : MAC Controller does not compare destination address with the register/address listed in the column.

14.4.6 MII Management Interface

MAC contains an MII Management Interface for an MII compliant PHY device. This allows control and status parameters to be passed between MAC and PHY by MDIO and MDC, thereby reducing the number of control pins required for PHY mode control.

The protocol consists of the bit stream that is sampled at rising edge of the MDC, the bit stream format is described below.

	PRE	ST	OP	PHYAD	REGAD	TA	DATA	IDLE
Read	1...1	01	10	AAAAA	RRRRR	Z0	D..D(16)	Z
Write	1...1	01	01	AAAAA	RRRRR	10	D..D(16)	Z

14.5 Initialization

14.5.1 Frame Transmitting Procedure

The frame transmitting procedure is as follows:

Initialization:

1. Set GMAC_MODE (MAC50 [9]) and SPEED_100 (MAC50 [19]) to proper setting.
2. Set SW_RST (MAC50 [31]) = 1 to do software reset. It takes about 200 system clocks for hardware to finish the software reset.
3. Read Feature Register (MAC44) to get the real TX/RX FIFO size in hardware.
4. Allocate system memory for the transmit descriptor ring and transmit buffer.
5. Initialize the transmit descriptor ring.
6. Set the Normal Priority Transmit Ring Base Address Register (MAC20) to the base address of the normal priority transmit descriptor ring in the system memory.
7. Set the High Priority Transmit Ring Base Address Register (MAC2C) to the base address of the high priority transmit descriptor ring in the system memory if necessary.
8. Set Interrupt Enable Register (MAC04).
9. Set MAC Address Register (MAC08).
10. Set Multicast Address Hash Table Register (MAC10).
11. Set Interrupt Timer Control Register (MAC30) to select the manner of the transmit interrupt.
12. Set Automatic Polling Timer Control Register (MAC34) to select the manner of transmit poll.
13. Set Transmit Priority Arbitration and FIFO Control Register (MAC48) to set transmit priority arbitration and proper TX/RX FIFO size in use.
14. Set DMA Burst Length and Arbitration Control Register (MAC38) to set proper TX/RX descriptor size and DMA burst length.
15. Set MAC Control Register (MAC50) to set valid configuration for MAC engine and to enable transmit channel.

Transmit procedure:

1. Software checks if the remainder of the normal priority transmit descriptors is enough for the next packet transmission. If not, software needs to wait until the transmit descriptors are enough.
2. Prepare the transmit packet data to the transmit buffer.
3. Set the normal priority transmit descriptor.
4. Write the Normal Priority Transmit Poll Demand Register (MAC18) to trigger MAC engine to poll the transmit descriptor if necessary when the packet is put in the normal priority transmit ring.
5. Wait for interrupt.
6. When interrupt occurs, software checks if it is a transmit interrupt. If ISR [4] = 1, it means the packet has been transmitted to network successfully. If ISR [7] = 1, it means the packet has been aborted during transmission due to late collision or excessive collision or under-run.
7. Steps 1 through 6 are for normal packets in the normal priority transmit ring. If software wants to transmit high priority packets, repeat these steps for the high priority transmit ring.

Note:

1. When setting the transmit descriptor, TXDES#0 must be set last. Thus, the setting procedure should be as follows:
   (a) Set TXDES#3
   (b) Set TXDES#2
   (c) Set TXDES#1
   (d) Set TXDES#0
2. When preparing a transmit packet which contains more than one transmit descriptors, the first transmit descriptor must be the last set descriptor of the transmit packet.

14.5.2 Frame Receiving Procedure

The frame receiving procedure is as follows:

Initialization:

1. Set GMAC_MODE (MAC50 [9]) and SPEED_100 (MAC50 [19]) to proper setting.
2. Set SW_RST (MAC50 [31]) = 1 to do software reset. It takes about 200 system clocks for hardware to finish the software reset.
3. Read Feature Register (MAC44) to get the real TX/RX FIFO size in hardware.
4. Allocate system memory for the receive descriptor ring and receive buffer.
5. Initialize the receive descriptor ring.
6. Set Receive Ring Base Address Register (MAC24) to the base address of the receive descriptor ring in the system memory.
7. Set Interrupt Enable Register (MAC04).
8. Set MAC Address Register (MAC08).
9. Set Multicast Address Hash Table Register (MAC10).
10. Set Interrupt Timer Control Register (MAC28) to select the manner of the receive interrupt.
11. Set Automatic Polling Timer Control Register (MAC34) to select the manner of receive poll.
12. Set Transmit Priority Arbitration and FIFO Control Register (MAC48) to set transmit priority arbitration and proper TX/RX FIFO size in use.
13. Set DMA Burst Length and Arbitration Control Register (MAC38) to set proper TX/RX descriptor size and DMA burst length.
14. Set MAC Control Register (MAC50) to set valid configuration for MAC engine and to enable receive channel.
15. Write Receive Poll Demand Register (MAC1C) to trigger MAC engine to poll the receive descriptor.

Receive procedures:

1. Wait for interrupt.

2. When interrupt occurs, software checks if it is a receive interrupt. If ISR [0] = 1, it means the packet has been moved to the receive buffer successfully. Then software needs to fetch the receive descriptor to get the receive packet until the owner bit of the next receive descriptor does not belong to software.
3. Software releases the receive descriptors to MAC engine after accessing the received packet.
4. If the receive automatic poll function is disabled, software needs to write Receive Poll Demand Register (MAC1C) to trigger MAC engine to poll the receive descriptor.

15 USB2.0 Virtual Hub Controller

15.1 Overview

USB2.0 Controller implements 1 set of USB Hub register and 7 sets of USB Device registers. The physical address of these registers can be derived as the following:

Base address of USB Hub= 0x1E6A_0000
Physical address = (Base address) + Offset

15.2 Features

- Complies with the Universal Serial Bus specification Rev. 2.0, Supports USB Full Speed (12Mb/sec) and High Speed (480 Mb/sec), backward compatible with USB1.1.
- USB Hub architecture, supports 1 hub device port and 7 downstream device ports.
- Supports 21 programmable endpoints which can be assigned to any devices, and can be configured to Bulk IN/OUT, Interrupt IN/OUT and Isochronous IN/OUT type endpoint.
- For Hub device, supports :
  1. 1 default Control endpoint.
  2. 1 dedicated hub status Interrupt IN endpoint.
  3. Any number (1-15) of programmable endpoints.
- For each Downstream device controller, supports :
  1. 1 default Control endpoint.
  2. Any number (1-15) of programmable endpoints.
- Automatic retry of failed packets, and PING Flow control.
- Separate data buffers for the SETUP data of a CONTROL transfer
- Integrated DMA engine for direct memory bus accesses (bypass AHB bus).
- Supports independent DMA channel for each endpoint.
- Supports 32 stages descriptor mode for all 21 programmable endpoints.
- Supports USB remote wake-up function (Suspend/Resume operation).

15.3 Registers : Base Address = 0x1E6A:0000

15.3.1 Address Definition

Offset	Size(Byte)	Description
0x03F-0x000	64	Root/Global Device Register
0x07F-0x040	64	Reserved
0x087-0x080	8	Root Device SETUP Data Buffer
0x08F-0x088	8	Device 1 SETUP Data Buffer
0x097-0x090	8	Device 2 SETUP Data Buffer
0x09F-0x098	8	Device 3 SETUP Data Buffer
0x0A7-0x0A0	8	Device 4 SETUP Data Buffer
0x0AF-0x0A8	8	Device 5 SETUP Data Buffer
0x0B7-0x0B0	8	Device 6 SETUP Data Buffer
0x0BF-0x0B8	8	Device 7 SETUP Data Buffer
0x0FF-0x0C0	64	Reserved
0x10F-0x100	16	Device 1 Register
0x11F-0x110	16	Device 2 Register
0x12F-0x120	16	Device 3 Register
0x13F-0x130	16	Device 4 Register
0x14F-0x140	16	Device 5 Register
0x15F-0x150	16	Device 6 Register
0x16F-0x160	16	Device 7 Register
0x1FF-0x170	144	Reserved
0x20F-0x200	16	Programmable Endpoint 0 Register
0x21F-0x210	16	Programmable Endpoint 1 Register
0x22F-0x220	16	Programmable Endpoint 2 Register
0x23F-0x230	16	Programmable Endpoint 3 Register
0x24F-0x240	16	Programmable Endpoint 4 Register
0x25F-0x250	16	Programmable Endpoint 5 Register
0x26F-0x260	16	Programmable Endpoint 6 Register
0x27F-0x270	16	Programmable Endpoint 7 Register
0x28F-0x280	16	Programmable Endpoint 8 Register
0x29F-0x290	16	Programmable Endpoint 9 Register
0x2AF-0x2A0	16	Programmable Endpoint 10 Register
0x2BF-0x2B0	16	Programmable Endpoint 11 Register
0x2CF-0x2C0	16	Programmable Endpoint 12 Register
0x2DF-0x2D0	16	Programmable Endpoint 13 Register
0x2EF-0x2E0	16	Programmable Endpoint 14 Register
0x2FF-0x2F0	16	Programmable Endpoint 15 Register
0x30F-0x300	16	Programmable Endpoint 16 Register
0x31F-0x310	16	Programmable Endpoint 17 Register
0x32F-0x320	16	Programmable Endpoint 18 Register
0x33F-0x330	16	Programmable Endpoint 19 Register
0x34F-0x340	16	Programmable Endpoint 20 Register

15.3.2 Root/Global Register Definition

Offset: 00	HUB00: Root Function Control & Status Register	Init = 0
Bit	**R/W**	**Description**
31	R	**USB PHY clock enable status** 0: USB PHY clock is disabled by SCU0C bit[14]. 1: USB PHY clock is enabled. The procedure to enable USB2.0 controller was updated for A1 or newer chip versions: 1. Enable USB2.0 clock running(SCU0C[14] = 1), wait **10 ms** for clock stable 2. Disable USB2.0 global reset by setting SCU04[14] = 0 3. Disable USB2.0 PHY reset by setting HUB00[11] = 1 4. Start using USB2.0 controller
30:18	RW	**Reserved (0)**
17	RW	**Isochronous IN null data response control** 0: No response, wait host timeout 1: Return 0 byte DATA0 packet This bit controls the response action for Isochronous IN type endpoints when the IN data for transmitting not ready. The action can be no response or return an 0 byte DATA0 packet. When no response action is selected, then CSPLIT IN retry is possible.
16	RW	**Complete a "SPLIT IN Transaction" after SOF has been received** 0: Disable 1: Enable Because "Complete a SPLIT IN Transaction" has no ACK, its very difficult to define the end of the transaction. Set this bit to '1' will add SOF packet into the transaction finish check list. This bit must be set to 1 when Set_Address transfer cycle, else hardware can not finish the status phase IN transaction.
15	R	**Loop Back test result** 0: Fail 1: Pass This status will be cleared when disabling USB Test Mode
14	R	**Loop Back test finished** 0: Not yet finished 1: Finished This status will be cleared when disabling USB Test Mode
13	R	**USB PHY BIST result** 0: Pass 1: Fail This flag will be cleared by disabling USB PHY BIST function (HUB00 [12] = 0). Note: BIST stands for Built-In-Self-Test
12	RW	**USB PHY BIST control** 0: Turn off USB PHY BIST 1: Turn on USB PHY BIST
11	RW	**Disable USB PHY reset** 0: Enable USB PHY reset 1: Disable USB PHY reset

to next page

from previous page

10:8	RW	**USB Test Mode selection** 000: Disable 001: Enable Test J 010: Enable Test K 011: Enable Test SE0_NAK 100: Enable Test Packet 101: Reserved 110: Reserved 111: Enable Test Loop Back (for debugging purpose only)
7	RW	**Force USB bus state timer to work at test mode (for debugging purpose only)** 0: Normal operation mode 1: Force USB bus state to High Speed mode (X32 faster)
6	RW	**USB Force to High Speed State Mode (for debugging purpose only)** 0: Normal operation 1: Force the bus state to High Speed
5	RW	**USB Remote Wakeup signaling pulse width selection** 0: 8ms 1: 12ms
4	RW	**Enable manual Remote Wakeup** 0: No operation 1: Enable manual Remote Wakeup, can be set only in Suspend state This register can be effectively set to "1" only when USB Controller enters Suspend state, and this register will be automatically cleared by H/W after Remote Wakeup.
3	RW	**Enable automatic Remote Wakeup** 0: Disable 1: Enable automatic Remote Wakeup When this register is enabled, Remote Wakeup singling will be automatically issued whenever firmware write commands has been received in Suspend state.
2	RW	**Enable clock stopping in suspend state** 0: USB Controller won't stop clock even in Suspend state 1: USB Controller will stop clock in Suspend state This bit must set to '1' when using remote wakeup function.
1	RW	**Upstream port connection speed selection** 0: Select High Speed and Full Speed modes 1: Select Full Speed mode only
0	RW	**Enable upstream port connection** 0: Disable upstream port connection 1: Enable upstream port connection

Offset: 04		HUB04: Root Configuration Setting Register Init = 0
Bit	**R/W**	**Description**
31:16	R	**Status of DMA page buffer** **(for debugging purpose only)** 0: Page Free 1: Page Allocated Bit16: Status of Page #0 .... Bit31: Status of Page #15 USB Controller totally integrates 2K byes of SRAM to be allocated for data transmit of IN transaction. The 2K bytes of SRAM are uniformly divided into 16 pages, each of them is 128 bytes long. Each EP can only allocate one page buffer from them, but there are 3 pages (Page 0, Page 1 and Page 2) are arranged as a ring buffer and dedicated for the usage of the active EP. Therefore, the status bits of these three pages are reserved.
15:7		**Reserved**
6:0	RW	**Root function device address** Change the address will affect the packet receiving immediately. The address should be set after the status phase of the Set_Address control transfer command. The following is the Set_Address command sequence: 1. SETUP data packet, contains the new address information. 2. IN data packet, status phase with zero byte data returned. 3. Change to the new address.

Offset: 08		HUB08: Interrupt Control Register Init = 0
Bit	**R/W**	**Description**
31:18		**Reserved (0)**
17	RW	**Enable Programmable Endpoint Pool NAK Interrupt**
16	RW	**Enable Programmable Endpoint Pool ACK/STALL Interrupt**
15	RW	**Enable Device #7 Controller Interrupt**
14	RW	**Enable Device #6 Controller Interrupt**
13	RW	**Enable Device #5 Controller Interrupt**
12	RW	**Enable Device #4 Controller Interrupt**
11	RW	**Enable Device #3 Controller Interrupt**
10	RW	**Enable Device #2 Controller Interrupt**
9	RW	**Enable Device #1 Controller Interrupt**
8	RW	**Enable USB Suspend Resume Interrupt** This interrupt is used to notify software that the USB host is leaving suspend mode. Software can do something because USB device will wake up. If software doesn't need to know the resume event, please don't enable this bit, so that hardware will skip to wait software to do something.
7	RW	**Enable USB Suspend Entry Interrupt** This interrupt is used to notify software that the USB host is entering suspend mode. Software can do something because USB device will entering sleep state. If software doesn't need to know the suspend event, please don't enable this bit, so that hardware will skip to wait software to do something.
6	RW	**Enable USB Bus Reset Interrupt** When Bus Reset occurred, hardware will automatically reset the connection status and return to the not configured state. Software must also clear the related states and prepare for the new connection.

to next page

from previous page

5	RW	**Enable Hub EP1 IN Data packet ACK Interrupt**
4	RW	**Enable Hub EP0 IN Data packet NAK Interrupt**
3	RW	**Enable Hub EP0 IN Data packet ACK/STALL Interrupt**
2	RW	**Enable Hub EP0 OUT Data packet NAK Interrupt**
1	RW	**Enable Hub EP0 OUT Data packet ACK/STALL Interrupt**
0	RW	**Enable Hub EP0 SETUP Data packet ACK Interrupt**

Note :
The definition of this register is :
0 : Disable
1 : Enable

The enable control for Endpoint Pool is the first level interrupt enable bit for all the EPs allocated from Endpoint Pool.

The enable control for Device is the first level interrupt enable bit for all the downstream device controller.

Offset: 0C	**HUB0C: Interrupt Status Register**	**Init = 0**
Bit	**R/W**	**Description**
31:19		**Reserved (0)**
18	R	**USB command bus is dead locked** 0: Normal 1: Dead locked When this bit is set to ”1”, it means a FATAL ERROR has occurred. This kind of error happens whenever an CPU bus command couldnt successfully finish a bus transaction within USB Controller. The potential root causes include: **USB clock is stopped,** **USB PHY is failed,** **or USB Controller is still under Suspend state.** This interrupt can not be masked, and it will be cleared whenever USB Controller gets back to a normal condition, or has been reset.
17	R	**Programmable Endpoint Pool NAK Interrupt Occurs**
16	R	**Programmable Endpoint Pool ACK/STALL Interrupt Occurs**
15	R	**Device #7 Controller Interrupt Occurs**
14	R	**Device #6 Controller Interrupt Occurs**
13	R	**Device #5 Controller Interrupt Occurs**
12	R	**Device #4 Controller Interrupt Occurs**
11	R	**Device #3 Controller Interrupt Occurs**
10	R	**Device #2 Controller Interrupt Occurs**
9	R	**Device #1 Controller Interrupt Occurs**
8	RW	**USB Suspend Resume event has occurred** When this register is set to ”1”, it indicates that USB Upstream Port has resumed from Suspend state.
7	RW	**USB Suspend Entry event has occurred** When this register is set ”1”, it indicates that the USB Upstream Port has entered Suspend state.
6	RW	**USB Bus Reset has Occurred** When set, indicates a USB Bus Reset state occurs.

to next page

from previous page

5	RW	**EP1 IN Data Packet ACK/STALL Returned** When set, indicates an IN transaction finished with ACK transmitted.
4	RW	**EP0 IN Data Packet NAK Returned** When set, indicates an IN packet received and responded with NAK. Control Endpoint IN/OUT NAK response interrupt can used to indicate the current Host transaction type, and S/W can based on this to determine the current Control Transfer stage.
3	RW	**EP0 IN Data Packet ACK/STALL Returned** When set, indicates an IN transaction finished with ACK or STALL transmitted.
2	RW	**EP0 OUT Data Packet NAK Returned** When set, indicates an OUT/PING packet received and responded with NAK. Control Endpoint IN/OUT NAK response interrupt can used to indicate the current Host transaction type, and S/W can based on this to determine the current Control Transfer stage.
1	RW	**EP0 OUT Data Packet ACK/STALL Returned** When set, indicates an OUT transaction finished with ACK/STALL returned, or a PING packet received and responded with STALL.
0	RW	**EP0 Setup Data Arrives** When set, indicates an SETUP transaction has been received successfully.

Note :
Each status bit of this register is set by H/W automatically when the corresponding event occurred or the transaction finished. The corresponding interrupt enable bit wont impact the setting of the bit. The enable bit only determines the interrupt generation.
Whenever S/W has finished the handling of the status bit, it must clear the status bit by writing '1', else it will not be able to recognize new events.

Offset: 10 HUB10: Programmable Endpoint Pool ACK Interrupt Enable Register Init = 0

Bit	R/W	Description
31:21		**Reserved (0)**
20:0	RW	**Programmable Endpoint ACK Interrupt Enable** bit 0: Programmable Endpoint number #0 bit 1: Programmable Endpoint number #1 .... bit20: Programmable Endpoint number #20 This register is to control the ACK interrupt enable bits of the 21 programmable Endpoints in the Endpoint Pool. The definition of each bit in this register is as the following: 0: Disable interrupt 1: Enable interrupt

Offset: 14 HUB14: Programmable Endpoint Pool NAK Interrupt Enable Register Init = 0

Bit	R/W	Description
31:21		**Reserved (0)**

to next page

from previous page

20:0	RW	**Programmable Endpoint NAK Interrupt Enable** bit 0: Programmable Endpoint number #0 bit 1: Programmable Endpoint number #1 .... bit20: Programmable Endpoint number #20 This register is to control the ACK interrupt enable bits of the 21 programmable Endpoints in the Endpoint Pool. The definition of each bit in this register is as the following: 0: Disable interrupt 1: Enable interrupt

Offset: 18 HUB18: Programmable Endpoint Pool ACK Interrupt Status Register Init = 0

Bit	R/W	Description
31:21		**Reserved (0)**
20:0	RW	**Programmable Endpoint ACK Interrupt Occurs** bit 0: Programmable Endpoint number #0 bit 1: Programmable Endpoint number #1 .... bit20: Programmable Endpoint number #20 This status flag will be set to '1' under any one of the following conditions: 1. STALL response. 2. When short packet received from OUT transaction. 3. When Interrupt Generation Enable been detected from the DMA descriptor, or for a single descriptor mode. 4. When the DMA descriptor list becomes empty, this indicates that the last descriptor been used.

Note :
Each status is automatically set to "1" whenever the ACK event has been occurred, no matter the corresponding interrupt enable bit has been enabled or not.
S/W must clear each status by writing '1' to its bit after finished the process; otherwise the next ACK event will not be recognized.

Offset: 1C HUB1C: Programmable Endpoint Pool NAK Interrupt Status Register Init = 0

Bit	R/W	Description
31:21		**Reserved (0)**
20:0	RW	**Programmable Endpoint NAK Interrupt Occurs** bit 0: Programmable Endpoint number #0 bit 1: Programmable Endpoint number #1 .... bit20: Programmable Endpoint number #20 This status flag will be set to '1' when the endpoint response with NAK.

Note :
Each status is automatically set to "1" whenever the NAK event has been occurred, no matter the corresponding interrupt enable bit has been enabled or not.
S/W must clear each status by writing '1' to its bit after finished the process; otherwise the next NAK event will not be recognized.

Offset: 20		HUB20: Device Controller Soft Reset Enable Register Init = 0x3FF
Bit	**R/W**	**Description**
31:10		**Reserved (0)**
9	RW	**Enable Programmable Endpoint Pool software Reset**
8	RW	**Enable DMA Controller software Reset**
7	RW	**Enable Device #7 Controller software Reset**
6	RW	**Enable Device #6 Controller software Reset**
5	RW	**Enable Device #5 Controller software Reset**
4	RW	**Enable Device #4 Controller software Reset**
3	RW	**Enable Device #3 Controller software Reset**
2	RW	**Enable Device #2 Controller software Reset**
1	RW	**Enable Device #1 Controller software Reset**
0	RW	**Enable Root HUB Controller software Reset**
Note : 0 : Normal operation 1 : Reset the device controller These software reset bits only reset the controllers status registers, not including all the registers supported by the controller. To reset all the registers of the controllers, please reference the registers of SCU04. Software sets the specific bit to '1' to start the reset process, and sets the specific bit to '0' to stop the reset process. There is no need to put time delay between the two processes.		

Offset: 24		HUB24: USB Status Register (for debugging purpose only) Init = X
Bit	**R/W**	**Description**
31	R	**USB Suspend State**
30	R	**USB Bus Reset State**
29	R	**USB Bus Line State DN**
28	R	**USB Bus Line State DP**
27	R	**USB Bus Speed** 0: Full Speed 1: High Speed SW reads this bit to determine the current host connection speed. But SW must read this bit after the first packet is received that behind the bus reset cycle.
26:16	R	**USB Last Frame Number record**
15	R	**UTMI State XcvrSelect** 0: High Speed 1: Full Speed
14	R	**UTMI State TermSelect** 0: High Speed 1: Full Speed
13:12	R	**UTMI State OPMode** 0: Normal mode 1: Non Driving 2: Disable Bit-Stuff and NRZI encoding 3: Reserved
11:8	R	**Endpoint Number of the Last USB Transaction**
7		**Reserved (0)**
6:0	R	**Device Address of the Last USB Transaction**

Offset: 28		HUB28: Programmable Endpoint Pool Data Toggle Value Set Init = X
Bit	**R/W**	**Description**
31:9		**Reserved (0)**
8	W	**Endpoint data toggle bit initial value set** 0: Initial to sequence DATA0 1: Initial to sequence DATA1 The indexed data toggle bit is determined by Bit [4:0] of this register. Only one data toggle bit can be initialized for each register write. Reading this register always returns "0".
7:5		**Reserved (0)**
4:0	W	**Programmable Endpoint Index** 0-20: Endpoint number index 21-31: Invalid This index value determines which Endpoint data toggle bit will be initialized. Reading this register always returns "0".

Note :
Data toggle sequence initialization can only be applied to Control/Bulk/Interrupt type Endpoints.
Isochronous type Endpoints should not be initialized; it will be reset automatically when SOF receives.

HUB2C: Isochronous Transaction Fail Accumulator (for debugging purpose only)		
Offset: 2C		Init = 0
Bit	**Attr.**	**Description**
31:26		**Reserved (0)**
25:16	RW	**Isochronous OUT Failure Counter**
15:10		**Reserved**
9:0	RW	**Isochronous IN Failure Counter**

Note :
Writing any data to this register will clear the two counters to 0.
The two counter values show the number of isochronous transaction failures which are due to either buffer unavailable or data not yet ready.

Offset: 30		HUB30: Endpoint 0 Control/Status Register Init = 0
Bit	**R/W**	**Description**
31:23		**Reserved (0)**
22:16	R	**Endpoint 0 OUT received data byte count**
15		**Reserved (0)**
14:8	RW	**Endpoint 0 IN data byte count for transfer**
7:3		**Reserved (0)**
2	RW	**Endpoint 0 OUT buffer ready for receiving data** 0: Buffer is not ready to receive data 1: Buffer is ready to receive data S/W can set this bit to '1' when it is ready to receive data from Host by OUT transactions. When H/W receives the data successfully, this bit will be automatically cleared to "0". S/W can monitor this bit to check the data has been received or not.

to next page

from previous page

1	RW	**Endpoint 0 IN buffer ready for transferring data** 0: Data is not ready to transfer data 1: Data is ready to transfer data S/W can set this bit to '1' when there is a need to transfer data to Host by IN transactions. When H/W has finished the data transfer, this bit will be automatically cleared. S/W can monitor this bit to check the data has been transferred or not. Only one of Bit [1] and Bit [2] can be set to "1" for each time. The two settings are exclusive.
0	RW	**Endpoint 0 STALL control** When this register is set to "1", Endpoint 0 returns STALL response for Data and Status stages. This bit is automatically cleared to "0" after a new SETUP packet is received.

Offset: 34 — HUB34: Base Address of Endpoint 0 IN/OUT Data Buffer Register — Init = X

Bit	R/W	Description
31:28		**Reserved (0)**
27:3	RW	**Base address of Endpoint 0 IN/OUT data buffer** This register defines the base address of Default Control Endpoint transaction data buffer, which is 64 bytes long and at 64-bit boundary. The direction of this buffer is determined by S/W and depends on the Control Transfer Command mode.
2:0		**Reserved (0)**

Offset: 38 — HUB38: Endpoint 1 Control/Status Register — Init = 0

Bit	R/W	Description
31:3		**Reserved (0)**
2	W	**Reset Endpoint 1 data toggle bit to DATA0** 0: No operation 1: Reset data toggle bit to DATA0
1	RW	**Endpoint 1 STALL control** When this register is set to 1, Endpoint 1 will return STALL response for this endpoint polling.
0	RW	**Enable Endpoint 1** 0 : Disabled 1 : Enabled

Offset: 3C — HUB3C: Endpoint 1 Status Change Bitmap Data — Init = 0

Bit	R/W	Description
31:8		**Reserved**
7	RW	**Port #7 Status Change Bit (Device #7)**
6	RW	**Port #6 Status Change Bit (Device #6)**
5	RW	**Port #5 Status Change Bit (Device #5)**
4	RW	**Port #4 Status Change Bit (Device #4)**
3	RW	**Port #3 Status Change Bit (Device #3)**
2	RW	**Port #2 Status Change Bit (Device #2)**
1	RW	**Port #1 Status Change Bit (Device #1)**
0	RW	**Hub Port Status Change Bit**

to next page

from previous page

Note :
When any bits of this register is not equal to zero, then USB Controller will automatically response with this byte of data; otherwise it will response with NAK when being polled for this Endpoint.

15.3.3 Device #1 — #7 Register Definition

Offset: 00		DEV00: Downstream Device Function Enable Control Register Init = 0
Bit	**R/W**	**Description**
31:15		**Reserved (0)**
14:8	RW	**Downstream Device Address** Change the address will affect the packet receiving immediately. The address should be set after the status phase of the Set_Address control transfer command. The following is the Set_Address command sequence: 1. SETUP data packet, contains the new address information. 2. IN data packet, status phase with zero byte data returned. 3. Change to the new address.
7		**Reserved (0)**
6	RW	**Enable Endpoint 0 IN data packet NAK interrupt** 0: Disable 1: Enable
5	RW	**Enable Endpoint 0 IN data packet ACK/STALL interrupt** 0: Disable 1: Enable
4	RW	**Enable Endpoint 0 OUT data packet NAK interrupt** 0: Disable 1: Enable
3	RW	**Enable Endpoint 0 OUT data packet ACK/STALL interrupt** 0: Disable 1: Enable
2	RW	**Enable Endpoint 0 SETUP data packet ACK interrupt** 0: Disable 1: Enable
1	RW	**Device port speed selection** 0 : Full Speed mode or Low Speed mode 1 : High Speed mode
0	RW	**Enable device port** 0 : Disable device port 1 : Enable device port Whenever Upstream Port Bus Reset occurs, this bit will be cleared.

Offset: 04		DEV04: Interrupt Status Init = 0
Bit	**R/W**	**Description**
31:5		**Reserved (0)**
4	RW	**Endpoint 0 IN data packet NAK returned** When this register is set to ”1”, it indicates that an IN transaction has been finished with NAK response. Endpoint 0 IN/OUT NAK response interrupt can be used to indicate the current Host transaction type, and S/W can based on this to determine the current Control Transfer stage.

to next page

from previous page

3	RW	**Endpoint 0 IN data packet ACK received or STALL returned** When this register is set to ”1”, it indicates that an IN transaction has been finished with ACK/STALL response.
2	RW	**Endpoint 0 OUT data packet NAK returned** When this register is set to ”1”, it indicates that an OUT transaction has been finished with NAK response. Endpoint 0 IN/OUT NAK response interrupt can be used to indicate the current Host transaction type, and S/W can based on this to determine the current Control Transfer stage.
1	RW	**Endpoint 0 OUT data packet ACK/STALL returned** When this register is set to ”1”, it indicates that an OUT transaction has been finished with ACK/STALL response. Or a PING packet received and responded with STALL.
0	RW	**Endpoint 0 SETUP data packet received** When this register is set to ”1”, it indicates that a SETUP transaction been finished.

Note :
Each status bit of this register is set by H/W automatically when the corresponding event occurred or the transaction finished. The corresponding interrupt enable bit wont impact the setting of the bit. The enable bit only determines the interrupt generation.
Whenever S/W has finished the handling of the status bit, it must clear the status bit by writing ’1’, else it will not be able to recognize new events.

Offset: 08 — DEV08: Endpoint 0 Control/Status Register — Init = 0

Bit	R/W	Description
31:29		**Reserved (0)**
28	R	**CSPLIT IN Wait** **(for debugging purpose only)**
27	R	**Normal IN Wait** **(for debugging purpose only)**
26	R	**Start SPLIT Cycle** **(for debugging purpose only)**
25:24	R	**Status of Transmit DMA State Machine** **(for debugging purpose only)** 00 : Idle 01 : DMA Request 10 : DMA Done and Data Ready 11 : Reserved
23		**Reserved (0)**
22:16	R	**Endpoint 0 OUT received data byte count**
15		**Reserved (0)**
14:8	RW	**Endpoint 0 IN data byte count for transfer**
7:3		**Reserved (0)**
2	RW	**Endpoint 0 OUT buffer ready for receiving data** 0: Buffer is not ready to receive data 1: Buffer is ready to receive data S/W can set this bit to ’1’ when it is ready to receive data from Host by OUT transactions. When H/W receives the data successfully, this bit will be automatically cleared to ”0”. S/W can monitor this bit to check the data has been received or not.

to next page

from previous page

1	RW	**Endpoint 0 IN buffer ready for transferring data** 0: Data is not ready to transfer data 1: Data is ready to transfer data S/W can set this bit to '1' when there is a need to transfer data to Host by IN transactions. When H/W has finished the data transfer, this bit will be automatically cleared. S/W can monitor this bit to check the data has been transferred or not. Only one of Bit [1] and Bit [2] can be set to "1" for each time. The two settings are exclusive.
0	RW	**Endpoint 0 STALL control** When this register is set to "1", Endpoint 0 returns STALL response for Data and Status stages. This bit is automatically cleared to "0" after a new SETUP packet is received.

Offset: 0C — DEV0C: Base Address of Endpoint 0 IN/OUT Data Buffer Register — Init = X

Bit	R/W	Description
31:28		**Reserved (0)**
27:3	RW	**Base address of data buffer** This register defines the base address of Default Control Endpoint transaction data buffer, which is 64 bytes long and at 64-bit boundary. The direction of this buffer is determined by S/W and depends on the Control Transfer Command mode.
2:0		**Reserved (0)**

15.3.4 Programmable Endpoint #0 — #20 Register Definition

Offset: 00 — EPP00: Endpoint Configuration Register — Init = 0

Bit	R/W	Description
31:16		**Reserved (0)**
15:14	RW	**Endpoint Isochronous Data Stages** Value | Stages | Isochronous IN | Isochronous OUT 00 | 1 | DATA0 | DATA0 01 | 2 | DATA1−>DATA0 | MDATA−>DATA1 1x | 3 | DATA2−>DATA1−>DATA0 | MDATA−>MDATA−>DATA2 The stages number setting is only applied to isochronous type endpoint with auto data toggle mode enabled. When SOF received, the data PID sequencing will be reset to the start data PID.
13	RW	**Endpoint Auto Data Toggle Disable** 0 : Enable auto data toggle 1 : Disable auto data toggle When auto data toggle is disabled for OUT direction endpoint, then all packets will be received without error, ignoring the data sequence errors. When auto data toggle is disabled for IN direction endpoint, then the data sequence PID will be replaced by the PID fetched from the descriptor list.
12	RW	**Endpoint Stall Control** When this bit is set to 1, the endpoint will always return STALL response until it is cleared to 0. The Stall control can only be set for Bulk/Interrupt type endpoints.

to next page

from previous page

11:8	RW	**Endpoint Number** This value defines the endpoint number of this endpoint.
7		**Reserved (0)**
6:4	RW	**Endpoint type selection** 00x : Disable 010 : Bulk In 011 : Bulk Out 100 : Interrupt In 101 : Interrupt Out 110 : Isochronous In 111 : Isochronous Out
3:1	RW	**Allocated Device Port Number** 000 : Root device 001 : Downstream device 1 010 : Downstream device 2 011 : Downstream device 3 100 : Downstream device 4 101 : Downstream device 5 110 : Downstream device 6 111 : Downstream device 7
0	RW	**Enable Endpoint** 0 : Disabled (this endpoint will be reset) 1 : Enabled

Note :
Endpoint Reset can be initiated by the following method:

1. System global reset controlled in SCU, this will reset full controller including registers.
2. Release the endpoint, disable endpoint.
3. Set the reset bit at HUB20 Bit[9] or HUB20 Bit[n], where n is device number set in the Port Number field.

Only the first item will reset the register value, others don't.

Offset: 04		EPP04: DMA Descriptor List Control/Status Register — Init = 0
Bit	**R/W**	**Description**
31:21		**Reserved (0)**
20	R	**Occupied Transmit IN Buffer Status** **(for debugging purpose only)** 0 : No buffer 1 : 1 buffer occupied
19:16	R	**Occupied Transmit IN Buffer Index** **(for debugging purpose only)**
15:13		**Reserved (0)**
12	R	**The Current Interrupt Generation Flag** **(for debugging purpose only)** This bit shows the interrupt generation status, fetched from the current stage descriptor.
11	R	**CSPLIT IN Wait** **(for debugging purpose only)**
10:9	R	**Auto Data Toggle Count** **(for debugging purpose only)**
8	R	**Start SPLIT Cycle** **(for debugging purpose only)**

to next page

from previous page

7:4	R	**Current Descriptor Processing Status** **(for debugging purpose only)** 00 : RX Idle 01 : RX Read Descriptor Request 02 : RX Read Descriptor Grant 03 : RX Read Descriptor Data Back and Buffer Ready 04 : RX OUT Data Receive Cycle 05 : RX OUT Transaction ACK and Descriptor Write Back Request 06 : RX Descriptor Write Back Grant 07 : RX DMA Done 08 : TX Idle 09 : TX Read Descriptor Request 10 : TX Read Descriptor Grant 11 : TX Read Descriptor Data Back and Buffer Ready 12 : TX IN Data DMA Fetch Request 13 : TX IN Data Ready 14 : TX IN Data Transfer Cycle 15 : TX IN Transaction DONE
3		**Reserved (0)**
2	RW	**Descriptor List Operation Reset** Sets this bit to '1' will reset the descriptor operation and flush buffers. Sets this bit to '0' to disable reset.
1	RW	**Single Stage Descriptor Mode** 0 : 32 stages descriptor 1 : 1 stage descriptor, when the OUT/IN transaction done, the CPU Write pointer will be cleared to 0 automatically. The single mode operation is started by set the Write pointer at EPP0C to 1, this means the data/buffer ready for TX/RX transfer. And when transfer done, the Write pointer will be cleared to 0 by H/W.
0	RW	**Descriptor List Operation Enable** 0 : Disable, for single stage mode 1 : Enable normal operation The descriptor list operates at 32 stages Ring mode. Descriptor List Operation Enable and Single Stage Descriptor modes are mutually exclusive. They cannot be set at the same time. Single mode has the higher priority.

Offset: 08		**EPP08: DMA Descriptor/Buffer Base Address** **Init = X**
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:3	RW	**Base address** Descriptor Enabled : Descriptor list base address. Descriptor Disabled : DMA data buffer base address. 8 bytes(64 bits) boundary
2:0		**Reserved (0)**

<table>
<tr><th colspan="3">EPP0C: DMA Descriptor List Read(DMA)/Write(CPU) Pointer and Status</th></tr>
<tr><th colspan="2">Offset: 0C</th><th>Init = 0</th></tr>
<tr><th>Bit</th><th>Attr.</th><th>Description</th></tr>
<tr><td>31</td><td>R</td><td>Descriptor List Empty Flag (Default=0)
0 : not empty
1 : empty</td></tr>
<tr><td>30</td><td></td><td>Reserved (0)</td></tr>
<tr><td>29:28</td><td>RW</td><td>Endpoint Current Data Toggle Sequence Value (Default=0)
00 : Indicates the Data PID of next transaction would be DATA0
01 : Indicates the Data PID of next transaction would be DATA2
10 : Indicates the Data PID of next transaction would be DATA1
11 : Indicates the Data PID of next transaction would be MDATA

1. When read, this value indicates the current internal register state used for next transaction Data sequence PID.
2. When write, used to setting the next transaction Data sequence PID. Only valid at Single descriptor mode, endpoint type ”IN” and auto data toggle disabled.</td></tr>
<tr><td>27</td><td></td><td>Reserved (0)</td></tr>
<tr><td>26:16</td><td>RW</td><td>Packet Size (Default=X)
The unit is byte.
This field has 4 definitions:
<table>
<tr><td rowspan="2">Descriptor Type</td><td colspan="2">Endpoint Type</td></tr>
<tr><td>IN</td><td>OUT</td></tr>
<tr><td>Single</td><td>TxDataByteCnt(1)</td><td>RxDataByteCnt(3)</td></tr>
<tr><td>32 stages</td><td>TxDataByteCnt(2)</td><td>MaxPacketSize(4)</td></tr>
</table>
(1) When endpoint type = IN and single mode enabled :
Transmit data packet length
This is the transmit data packet length for IN transfer.
Setting by SW.
(2) When endpoint type = IN and descriptor 32 stages :
Transmit data packet length
This is the transmit data packet length for IN transfer.
Fetched from descriptor list entry.
(3) When endpoint type = OUT and single mode enabled :
Received data packet length
This is the received data packet length of OUT transfer.
Setting by RXDMA controller.
(4) When endpoint type = OUT and descriptor 32 stages :
Endpoint Maximum Packet Size
This is used for OUT transaction short packet interrupt trigger.
When the OUT data length received less than the Maximum Packet Size, then the OUT ACK interrupt flag will raise.
Setting by SW.
SW can write this value when descriptor is under disabled state, including single descriptor mode.</td></tr>
</table>

to next page

from previous page

15:13		**Reserved (0)**
12:8	RW	**Descriptor List DMA Read Pointer (Default=0)** This shows the current descriptor read position that will be read by DMA controller if it is not empty(equal to CPU write pointer). This pointer can be initialized by S/W when descriptor operation is disabled and not at single mode. The correct initialize procedure is as follows: 1. Disable the descriptor operation first 2. Wait descriptor operation status Idle 3. Update the Read Pointer 4. Enable the descriptor operation
7:5		**Reserved (0)**
4:0	RW	**Descriptor List CPU Write Pointer (Default=0)** For transmit (IN) direction, this pointer indicates the transmit data buffer allocated to DMA. For receive (OUT) direction, this pointer indicates the receiver free buffer allocated to DMA. When descriptor operation is enabled, this value indicates the next descriptor write position that will be writing by CPU. And the DMA operation will increment until read pointer equals to write pointer that indicates the Empty condition. The descriptor stage that Write Pointer addressed is not valid, and DMA will not process it. The descriptor list usage cannot be fully, there needs 1 free space for differentiation full and empty cases. That is when WPTR = RPTR, it means empty status. And full status equals WPTR = RPTR-1. The SW can maximum fills the descriptor entry until the full condition; otherwise it will conflict with the empty condition.
Note : The descriptor operation is that the Write Pointer is the leading pointer, and Read Pointer will track at the tail of Write Pointer. Read and Write pointer will be reset to 0 when device reset (HUB20) or USB bus reset occurs.		

15.3.5 Programmable Endpoint DMA Descriptor Definition

Offset: 31:0		DES_0: Data Buffer Base Address Init = X
Bit	**R/W**	**Description**
31:28		**Reserved**
27:3	RW	**DMA Data Buffer Base Address** It should be 8 bytes (64 bits) address boundary
2:0		**Reserved**

Offset: 63:32		DES_1: Descriptor Control/Status Init = X
Bit	**R/W**	**Description**
63	RW	**Enable Interrupt Generation** 0 : No interrupt 1 : Interrupt generated when transaction done. This bit is used by SW to control the generation of interrupt, when SW wants to reduce the interrupts number. By this interrupt control, interrupt will be generated on this endpoint only when : a. Interrupt generation enable = 1 and transaction done for this descriptor. b. Descriptor list empty and the last transaction done. c. Stall response occurs. d. NAK response occurs and descriptor list empty and NAK interrupt enabled.
62:60	R	**Device Port Number (RX only)** 000 : Root Hub 001 : Port1 010 : Port2 011 : Port3 100 : Port4 101 : Port5 110 : Port6 111 : Port7 This port number status only used for indicating the received transaction status. No any meaning for transmit path.
59:56	R	**Endpoint Number (RX only)** This endpoint number status only used for indicating the received transaction status. No any meaning for transmit path.
55		**Reserved**
54:48	R	**Device Address (RX only)** This device address status only used for indicating the received transaction status. No any meaning for transmit path.
47:46	RW	**Data Packet PID** 00 : DATA0 01 : DATA2 10 : DATA1 11 : MDATA When RX, this indicates the Data PID that current packet received. When TX, the Data PID will be used for transmitting if the HW auto data toggle (EPP00.bit[13]) is been disabled.
45	R	**End of Packet(E) (RX only)** This is used for Isochronous OUT to a Full Speed device.

to next page

from previous page

44	R	**Start of a Packet(S) (RX only)** S E | High-Speed to Full-Speed Isochronous OUT Data Relation 0 0 | High-speed data is the middle of the full-speed data payload 0 1 | High-speed data is the end of the full-speed data payload 1 0 | High-speed data is the beginning of the full-speed data payload 1 1 | High-speed data is all of the full-speed data payload This is used for Isochronous OUT to a Full Speed device.
43	RW	**OUT Packet Valid Flag** This bit will be written 1 by hardware when OUT Transaction DMA done.
42:32	RW	**Packet Length in Bytes** When RX, this will be written by DMA to indicate the received packet length, not include CRC. When TX, CPU must initialize this field to indicate the packet length for transmit.

15.3.6 Register Reset Table

Reg	Bit	SCU bit[14]	Bus Reset	HUB20 bit[0]	HUB20 bit[7:1]	HUB20 bit[9]	EPP00 bit[0]	Others
HUB00	30:16	Y						
HUB00	15:14	Y						HUB00[10:8] != 111
HUB00	13:0	Y						
HUB04	6:0	Y	Y					
HUB08	17:0	Y						
HUB0C	17:16	Y	Y			Y		
HUB0C	15:9	Y	Y		Y(Device specific)			
HUB0C	8:0	Y						
HUB10	20:0	Y						
HUB14	20:0	Y						
HUB18	20:0	Y	Y		Y(EPP00[3:1] port specific)	Y	Y(EP Specific)	
HUB1C	20:0	Y	Y		Y(EPP00[3:1] port specific)	Y	Y(EP Specific)	
HUB20	9:0	Y(All 1)						
HUB2C	25:16	Y	Y	Y				
HUB2C	9:0	Y	Y	Y				
HUB30	2:0	Y						
HUB38	1:0	Y						
HUB3C	7:0	Y						
DEV00	14:8	Y	Y					
DEV00	6:1	Y						
DEV00	0	Y	Y					
DEV04	4:0	Y	Y		Y(Device specific)			
DEV08	28:24	Y	Y		Y(Device specific)			
DEV08	2:0	Y						
EPP00	15:0	Y						
EPP04	12:8	Y	Y		Y(EPP00[3:1] port specific)	Y	Y	
EPP04	7:4	Y	Y		Y(EPP00[3:1] port specific)	Y	Y	EPP04[2] = 1
EPP04	2:0	Y						
EPP0C	12:8	Y	Y		Y(EPP00[3:1] port specific)	Y	Y	EPP04[1] = 1
EPP0C	4:0	Y	Y		Y(EPP00[3:1] port specific)	Y	Y	

For all other registers not included, no reset control implemented.

15.4 Software Programming Guide

15.4.1 Reset Control

The RESET control of USB2.0 Virtual Hub Controller have the following types :

Global Reset This is an asynchronous reset control and will reset full controller to its initial state, including all registers.
This reset can be initiated from SCU04 reset control register.

Bus State Reset When upstream USB host controller issues a Bus Reset state, then the root hub controller will be reset to an un-configured state, which with ZERO device address. And all downstream device controllers and endpoints will be removed from plugging state and all state machine and pointer will be reset to its initial state.
SW must send a bus reset command to remote USB host controller and restart the device initialization process.

Device Reset HUB20 contains reset control for each device. The reset control only reset the device state into its initial state; register value will not be cleared. The state will be reset including all state machines, FIFO pointers and descriptor pointers. For all unused downstream device ports, the device reset must be enabled. The following is a sequence for controlling device reset:

- Attach a device.
  1. Disable the specific device port reset.
  2. Enable device port and doing initialization.
- Remove a device.
  1. Disable device port function.
  2. Enable the specific device port reset.

Endpoint Pool Reset The Pool of 21 programmable endpoints can be reset by the following conditions:

1. Global reset at SCU will reset all registers with initial value defined.
2. HUB20 bit[9] will reset all endpoints in endpoint pool.
3. Upstream USB Bus Reset will reset all endpoints.
4. The specific device's reset that it was been allocated.
5. Endpoints been disabled will also been reset.

15.4.2 Initialization Sequence

Hub Connection

1. Enable USB2.0 clock running(SCU0C[14] = 1), wait **10 ms** for clock stable
2. Disable USB2.0 global reset by setting SCU04[14] = 0
3. Enable VIC interrupt setting.
4. Write HUB0C = 0xffffffff
5. Write HUB18 = 0xffffffff
6. Write HUB1C = 0xffffffff

7. Write HUB20 = 0x2FE
8. Write HUB08 = interrupt necessarily
9. Write HUB38 = 0x5
10. Write HUB00 = 0x800
11. Write HUB00 = 0x805
12. Start handshake with USB host controller for Hub configuration

Hub Disconnection

1. Enable USB2.0 global reset by setting SCU04[14] = 1
2. Disable USB2.0 clock running by setting SCU0C[14] = 0

Downstream Device Attachment

1. Disable device and endpoint pool reset by setting the specific bit in HUB20 to '0'
2. Enable device and endpoint pool interrupt by setting the specific bit in HUB08 to '1'
3. Assign endpoint to device
4. Write DEV04 = 0xffffffff
5. Write DEV00 = 0x01 + (interrupt necessarily)
6. Set the specific port status change bit in HUB3C
7. Start handshake with USB host controller for device configuration

Assign Endpoint to Device

1. Enable endpoint interrupt by setting the specific bit in HUB10 and HUB14 to '1'
2. Enable endpoint, assign the device port, set the endpoint type and endpoint number
3. Create endpoint's DMA, single buffer mode or descriptor list mode

15.4.3 Set Device Address

Hub Controller

- When received Set_Address command, record the address first.
- Prepare a zero length data packet for the status phase IN transaction.
- Wait IN transaction ACK interrupt.
- Modify the controller device address field with the new address.

Downstream Device Controller

- When received Set_Address command, record the address first.
- If the attached device is working at full/low speed and hub is working at high speed, write HUB00 bit[16] = 1.
- Prepare a zero length data packet for the status phase IN transaction.
- Wait IN transaction ACK interrupt.
- Modify the controller device address field with the new address.
- If the attached device is working at full/low speed and hub is working at high speed, write HUB00 bit[16] = 0.

15.4.4 Response STALL

When software wants to response a STALL handshake to Host, then Host will send a Clear_Feature command to endpoint. When software received this command, it must do a toggle reset operation to reset the data toggle sequence of this endpoint to DATA0.

The toggle reset command is defined at HUB28 and HUB38.

15.4.5 Programmable Endpoint OUT Transfer Finish Check

For programmable endpoints OUT transfer that using descriptor list mode. Software must check both the descriptor read pointer and the ”valid” bit in the descriptor bit[43] to determine a finished transfer. Especially for the ”valid” bit in the descriptor list. Sometimes there is a latency delay between software can detect the change of the read pointer and the valid bit updated. This is caused by DRAM controller arbitration mechanism. So somtimes CPU read an empty descriptor before the correct descriptor flush into DRAM. This case happens rarely, but it is possible to happen. So software must polling the valid bit several times after it received an interrupt and detected the change of descriptor read pointer.

15.4.6 Prevent a Transient Read Pointer Value

There is a possibility to read a temp value for the descriptor read pointer. This is caused by the USB registers read and the pointer update happened on 2 clock domains. So it is possible to read a wrong transient value when the pointer is updating. This fail case often will be encountered when multiple stages descriptor enabled concurrently. To prevent the fail case, SW must continuously read the descriptor read pointer until 2 consecutive same value got.

15.4.7 Procedure to enable Interrupt

The USB2.0 interrupt function must follow the sequence to enable.

1. Enable USB2.0 clock at SCU0C[14]=1 and wait **10 ms** for clock stable.
2. Disable USB2.0 global reset at SCU04[14]=0.
3. Set interrupt type as sensitive-high level trigger at VIC24[5]=1 and VIC2C[5]=1.
4. Enable USB2.0 interrupt at VIC10[5]=1.

15.5 Hardware Limitation

15.5.1 Set_Address transfer can not finished

When upstream host speed is high-speed and downstream device is working at full-speed, SPLIT transfer is used. Under this operating mode, hardware will finish a SPLIT IN transaction only when a next START SPLIT packet for this endpoint was received. This way can work well for most conditions except Set_Address command. At Set_Address command, device needs to return a status IN transaction. After this, host will send packet to device using the new address. But firmware didn’t change to new address until the status IN transaction finished. This made hardware could not receive the new transaction from host to finish the status IN transaction. Thus dead lock.

There is a bit control HUB00.bit[16] that can make hardware to finish the SPLIT IN transaction without waiting the next START SPLIT packet. Set this bit to ’1’, then hardware will finish the SPLIT IN transaction whenever a SOF packet was received. This can solve the dead lock issue.

15.5.2 PRE packet not supported

This Virtual Hub controller didn't support "PRE" packet command.

When upstream host speed is working at Full speed mode, if a Low speed device attached, then host controller will send a preamble packet "PRE" before each normal packet command. PRE belongs to hardware physical layer communication protocol, not software layer command. This PRE packet is meaningless and invisible to device, it is a Hub command.

Since this Hub controller can not recognize the PRE packet and it will make state machine fail. Software Hub driver must prevent this to occur. When a Low speed device is attached, driver must tell the host that the attached device is Full speed. This will not affect the host controller driver to communicate with the device. All the commands from host or device can transmit to each other without any loss.

When upstream host speed is High speed mode, there is no any limitation to support Full or Low speed devices.

15.5.3 DMA buffer overflow

There is a possibility to receive more data bytes than expected when the USB bus signal is worse. The USB controller DMA engine only protect for packet size larger than 1024 bytes. For the packet size smaller or equal to 1024 bytes, it will be transferred to DRAM. This will cause memory over-run if software didn't reserve an extra space for the un-expected data.

This case happen rarely. We only ever seen less than 8 bytes of extra data. But for safe design, it is recommended to allocate at least 1024 bytes buffer for the last stage DMA buffer.

The following are some rules for DMA usage:

- For single buffer case, allocate at least 1024 bytes buffer.
- For descriptor list case, allocate at least 1024 bytes buffer for the last stage descriptor.
- The buffer arrangement for the descriptor list must use incremental order, lower stage occupied lower address space. So the over-run only affect the back stage.

15.5.4 Unique Endpoint Number for each Device

There is a limitation on the endpoint number assigned for each device attached.

The programmable endpoint can only be configured as one type of the following at a time:

- Bulk In
- Bulk Out
- Interrupt In
- Interrupt Out
- Isochronous In
- Isochronous Out

The endpoint number can be set from 1 to 15. And the endpoint number set to each programmable endpoint must be unique when attached to the same device. It can be the same number for different devices. For example:

1. Device ID 1:
   (a) Endpoint pool ID 0: EP number = 1

   (b) Endpoint pool ID 1: EP number = 2
   (c) Endpoint pool ID 2: EP number = 3

2. Device ID 2:
   (a) Endpoint pool ID 3: EP number = 1
   (b) Endpoint pool ID 4: EP number = 2
   (c) Endpoint pool ID 5: EP number = 3

3. Device ID 3:
   (a) Endpoint pool ID 6: EP number = 1
   (b) Endpoint pool ID 7: EP number = 2
   (c) Endpoint pool ID 8: EP number = 3

15.5.5 Full Speed Isochronous IN for Large Packet Size

There is a transfer condition that hardware didn't support. When the packet size of a Full speed Isochronous IN endpoint larger than 188 bytes. The traffic period on the full speed bus will longer than 125 us, which is the micro-frame period on the high speed bus. Under this condition, hub must split the long full speed packet to multiple 188 bytes base segment, and using the packet ID of "MDATA" for the intermediate isochronous IN polling, and "DATA0" for the last segment. For example, a 896 bytes full speed isochronous IN packet can be splitted to 5 high speed packets as following:

Full Speed:

1. IN $\Longrightarrow$ DATA0(896 bytes)

High Speed:

1. IN $\Longrightarrow$ MDATA(188 bytes)
2. IN $\Longrightarrow$ MDATA(188 bytes)
3. IN $\Longrightarrow$ MDATA(188 bytes)
4. IN $\Longrightarrow$ MDATA(188 bytes)
5. IN $\Longrightarrow$ DATA0(144 bytes)

There is a software workaround method that can meet this criterion. By the software manu data toggle mode, it can enumerate the desired data sequence. The following is the procedure of software workaround:

1. Set EPP00[13] = 1, disable the auto-data-toggle mode.
2. Software manually split the long data packet to 188 bytes segments, and place the sub-packets in the descriptor list in order.
3. Program the descriptor data bit[47:46] to the correct data toggle sequence.

15.5.6 High Speed High Bandwidth can not support

For High Speed High Bandwidth transfer type, there will have maximum 3 packets in a microframe. The host sends the 3 packets in consecutive with very short inter-packet delay (smaller than 1 us). It is too short for USB DMA controller to prepare next buffer (usually 700 ns when DRAM loading is low). So it is possible to lose packet. This is an intrinsic limitation in the USB IP. And the high bandwidth transfer type can not be used.

16 Interrupt Controller

16.1 Overview

Vector Interrupt Controller (VIC) is an AMBA salve device directly connected to AHB bus. It provides a hardware interface to interrupt ARM CPU, which provides two kinds of priority levels for different interrupt requests.

Fast Interrupt Request (FIQ): For higher priority and low latency interrupt requests
Interrupt Request (IRQ): For general interrupt requests

Its highly recommending that only one of all the interrupt sources is assigned as FIQ interrupt request. VIC supports up to 32 interrupt requests. Each interrupt request can be programmed to be with different trigger modes. AST2050 / AST1100 has assigned different interrupt requests for VIC as shown in Section 10.

VIC implements the following 13 registers to support various interrupt functions. Each register has its own specific offset value to derive its physical address location.

Base address of VIC = 0x1E6C_0000
Physical address = (Base address of VIC) + Offset

VIC00: IRQ Status Register
VIC04: FIQ Status Register
VIC08: Raw Interrupt Status Register
VIC0C: Interrupt Selection Register
VIC10: Interrupt Enable Register
VIC14: Interrupt Enable Clear Register
VIC18: Software Interrupt Register
VIC1C: Software Interrupt Clear Register
VIC20: Protection Enable Register
VIC24: Interrupt Sensitivity Register
VIC28: Interrupt Both Edge Trigger Control Register
VIC2C: Interrupt Event Register
VIC30: Reserved
VIC38: Edge-Triggered Interrupt Clear Register

16.2 Features

- Directly connected to AHB bus interface
- Support up to 32 interrupt sources
- Support rise/fall edge-triggered and high/low level-triggered interrupt settings
- Support programmable 32 levels of interrupt priority settings

16.3 Registers : Base Address = 0x1E6C:0000

Offset: 00h		VIC00: IRQ Status Register Init = 0
Bit	**R/W**	**Description**
31:0	R	**IRQ status** Shows the status of the interrupt after masking by **VIC10** and **VIC0C** registers. Value ”1” indicates that the interrupt is active, and generates an interrupt to the processor.

Offset: 04h		VIC04: FIQ Status Register Init = 0
Bit	**R/W**	**Description**
31:0	R	**FIQ status** Shows the status of the interrupt after masking by the **VIC10** and **VIC0C** register. Value ”1” indicates that the interrupt is active, and generates an interrupt to the processor.

Offset: 08h		VIC08: Raw Interrupt Status Register Init = 0
Bit	**R/W**	**Description**
31:0	R	**Raw interrupt status** Shows the status of the interrupt before masking by the **VIC10** register. Value ”1” indicates that an interrupt request is active before masking.

Offset: 0Ch		VIC0C: Interrupt Selection Register Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Interrupt selection** Select the types of interrupt for ARM interrupt request input: 1: FIQ interrupt 0: IRQ interrupt

Offset: 10h		VIC10: Interrupt Enable Register Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Enable the interrupt source** When read: 1: interrupt enable. Allow interrupt request to processor 0: interrupt disable When write: 1: enables the interrupt 0: no effect To clear this register bits from ”1” to ”0”, write the corresponding bits in VIC14 with value ’1’.

Offset: 14h		VIC14: Interrupt Enable Clear Register Init = 0
Bit	**R/W**	**Description**
31:0	W	**Clear bits in the VIC10 register** Write ’1’ clears the corresponding bit in the VIC10 register to value ”0”, and write ’0’ has no effect.

Offset: 18h		VIC18: Software Interrupt Register Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Software Interrupt Generation** Setting a bit generates a software interrupt for the specific source interrupt before interrupt masking. Write '1' sets the corresponding bit to value "1", and write '0' has no effect.

Offset: 1Ch		VIC1C: Software Interrupt Clear Register Init = 0
Bit	**R/W**	**Description**
31:0	W	**Clear bits in the VIC18 register** Write '1' clears the corresponding bit in the VIC18 register to value "0", and write '0' has no effect.

Offset: 20h		VIC20: Protection Enable Register Init = 0
Bit	**R/W**	**Description**
31:1		**Reserved (0)**
0	RW	**Enable or disable protected register access** When enabled, only privileged mode accesses can access the interrupt controller registers. When disabled, both user and privileged modes can access the registers. This register can only be accessed in privileged mode.

Offset: 24h		VIC24: Interrupt Sensitivity Register Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Select sensitivity type of interrupt for interrupt request** 1: level sensitive 0: edge trigger

Offset: 28h		VIC28: Interrupt Both Edge Trigger Control Register Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Select both or single edge for edge-trigger interrupt request** 1: both edge 0: single edge When sensitivity type is level sensitive, this register has no effect.

Offset: 2Ch		VIC2C: Interrupt Event Register Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Select sensitivity type of interrupt for interrupt request** 1: High-level sensitive or rising edge triggered. 0: Low-level sensitive or falling edge triggered.

Offset: 30h		VIC30: Reserved Init = X
Bit	**R/W**	**Description**
31:0		**Reserved** Any read/write to this register can cause incorrect operation.

Offset: 38h	VIC38: Edge-Triggered Interrupt Clear Register	Init = 0
Bit	**R/W**	**Description**
31:0	W	**Clear bits in the edge detection register** Write ’1’ clears the corresponding bit in the edge detection register, and write ’0’ has no effect. When select edge triggered interrupt sensitivity type, firstly it must clear the detection register by writing ’1’ in this register, then it can enable the interrupt in VIC10, otherwise old interrupt status will take effect again.

17 SDRAM Memory Controller

17.1 Overview

SDRAM Controller implements 30 registers, which is listed below, to program the various SDRAM functions supported by AST2050 / AST1100 . Each register has its own specific offset value to derive its physical address location.

Base address of MCR = 0x1E6E_0000
Physical address = (Base address of SDRAM Controller) + Offset

MCR00: Protection Key Register
MCR04: Configuration Register
MCR08: Graphics Memory Protection Register
MCR0C: Refresh Timing Register
MCR10: Normal Speed AC Timing Register #1
MCR14: Low Speed AC Timing Register #1
MCR18: Normal Speed AC Timing Register #2
MCR1C: Low Speed AC Timing Register #2
MCR20: Normal Speed Delay Control Register
MCR24: Low Speed Delay Control Register
MCR28: Mode Setting Control Register
MCR2C: MRS/EMRS2 Mode Setting Register
MCR30: EMRS/EMRS3 Mode Setting Register
MCR34: Power Control Register
MCR38: Page Miss Latency Mask Register
MCR3C: Priority Group Setting Register
MCR40: Maximum Grant Length Register #1
MCR44: Maximum Grant Length Register #2
MCR48: Maximum Grant Length Register #3
MCR60: IO Buffer Mode Register
MCR64: DLL Control Register #1
MCR68: DLL Control Register #2
MCR6C: DLL Control Register #3
MCR70: Testing Control/Status Register
MCR74: Testing Start Address and Length Register
MCR78: Testing Fail DQ Bit Register
MCR7C: Test Initial Value Register
MCR100: AST2000 Backward Compatible SCU Password
MCR120: AST2000 Backward Compatible SCU MPLL Parameter
MCR170: AST2000 Backward Compatible SCU Hardware Strapping Value

Changing Memory Controller registers usually results in significant impact on SOC operations. Therefore, all these registers have to be well protected.

17.2 Fixed Priority DRAM Request

Priority	Request ID	Request Source
1	REQ0	VGA hardware cursor read
2	REQ1	VGA text mode CG font read
3	REQ2	VGA text mode ASCII code read
4	REQ3	VGA CRT controller read
5	REQ4	Video high priority write
6	REQ5	USB2.0 DMA read/write
7	REQ6	CPU data read/write
8	REQ7	CPU instruction read
9	REQ8	PCI bus write
10	REQ9	PCI bus read
11	REQ10	AHB bus read/write
12	REQ11	MAC1 DMA read/write
13	REQ12	MAC2 DMA read/write
14	REQ13	Reserved
15	REQ14	Reserved
16	REQ15	Encryption engine read/write
17	REQ16	2D command queue read
18	REQ17	Video flag read/write
19	REQ18	Video low priority write
20	REQ19	MDMA read/write
21	REQ20	2D engine data read/write
22	REQ21	I2C DMA buffer mode read/write
23	REQ22	Memory Integrity Check engine read

17.3 Registers: Base Address = 0x1E6E:0000

Offset: 00h		MCR00: Protection Key Register Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Protection key** This register is designed to protect the registers (MCR04 ~ MCR7C) of SDRAM controller from unpredictable updates, especially when ARM CPU is out of control; even under such a circumstance, properly protecting SDRAM registers can make sure that Graphic Display Controller is still able to successfully access SDRAM for displaying graphics correctly. The password of the protection key is **0xFC600309**. Reading back SDRAM registers is irrelevant with this register. **Unlock SDRAM registers**: Write 0xFC600309 to this register **Lock SDRAM registers**: Write others value to this register Whenever finished the initialization of SDRAM registers, please always set SDRAM registers into **locking** mode. The initial state of this register is at locked mode. When this register is locked, the read back value of this register is 0x00000000. When this register is unlocked, the read back value of this register is 0x00000001.

Offset: 04h		MCR04: Configuration Register Init = 0
Bit	**R/W**	**Description**
31:12		**Reserved (0)**
11	RW	**Select bank mode** 0 : Select 4-bank addressing mode 1 : Select 8-bank addressing mode The selection of this register is dependent on SDRAM specification. The bank information must be set exactly the same as the SDRAM specification, or the controller will mal-function un-predictably.
10	RW	**Enable SDRAM auto pre-charge command** 0: Disable auto pre-charge command 1: Enable auto pre-charge command Disabling auto pre-charge command will suffer SDRAM performance penalty in some extent. This register is designed for insurance policy only.
9:8	RW	**Select SDRAM data bus width** 01: Select 16-bit data bus width (DQ15–DQ0) others: Reserved
7	RW	**Select DRAM burst length** 0: Select burst length 2 (1 clock cycle for each read/write transaction) 1: Select burst length 4 (2 clock cycles for each read/write transaction)
6	R	**SDRAM Bus Width Status** 0: 32 bits 1: 16 bits This status is used for AST2000 backward compatible; it is decoded from bit [9:8].
5:4	RW	**Select graphics memory aperture size** 00: Select 8 M bytes of graphics memory aperture size 01: Select 16M bytes of graphics memory aperture size 10: Select 32M bytes of graphics memory aperture size 11: Select 64M bytes of graphics memory aperture size The size of graphics memory is set by external trapping resistors SCU70 [3:2] at page 217. The setting of this register should be always consistent with the setting of the corresponding trapping resistor. The graphics memory is always located at the highest address segment as shown in Page 98.
3:2	RW	**Select the total memory capacity** 00: Select 32M bytes or smaller as the total data memory capacity 01: Select 64M bytes as the total data memory capacity 10: Select 128M bytes as the total data memory capacity 11: Select 256M bytes as the total data memory capacity The mentioned total data memory capacity doesnt count ECC memory capacity, if any. The required ECC memory capacity depends on the selected ECC mode. Selecting ECC8 (by x16 SDRAM) will require 100% more memory capacity with full ECC capability. Selecting ECC16 (by x8 SDRAM) will require 50% more memory capacity but with partial ECC capability.

to next page

from previous page

1:0	RW	**Select the number of column address** 00 : Select 9 bits of column address DDR SDRAM: 128Mbit(x16), 256Mbit(x16) DDR2 SDRAM: 256Mbit(x16) 01 : Select 10 bits of column address DDR SDRAM: 256Mbit(x8), 512Mbit(x16), 1Gbit(x16) DDR2 SDRAM: 256Mbit(x8), 512Mbit(x8/x16), 1Gbit(x8/x16), 2Gbit(x16) 10 : Select 11 bits of column address DDR SDRAM: 512Mbit(x8), 1Gbit(x8) 11 : Reserved The number of column address listed above just follows JEDEC standard.

Offset: 08h — MCR08: Graphics Memory Protection Register — Init = 0

Bit	R/W	Description
31:0	RW	**Enable graphics memory request protection** Bit[n]: Protect REQn .... Bit[1]: Protect REQ1 Bit[0]: Protect REQ0 1: re-allocated to the highest memory space 0: not changed This register is designed to protect SDRAM memory from improper graphics memory updates by host CPU. SDRAM memory controller can serve a bunch of memory access requests from REQ0 to REQn. All the memory requests from Graphics Display Controller and PCI Bus Controller are among them. When the register bit corresponding to REQn is programmed to be 1, all the accesses of REQn will be re-allocated by address re-mapping to the highest memory space defined by MCR04 [5:4]. When being programmed to be 0, the request is not changed.

Offset: 0Ch — MCR0C: Refresh Timing Register — Init = 0

Bit	R/W	Description
31:16		**Reserved (0)**
15:8	RW	**Period of high-priority SDRAM refresh cycle** AST2050 / AST1100 uses 12MHz clock source as the reference clock to generate high-priority SDRAM refresh cycle requests according to the following equation. SDRAM Refresh Frequency = 12MHz / (period of refresh cycles)
7:6		**Reserved (0)**
5	RW	**Enable low-priority SDRAM refresh cycle** 0: Disable low-priority SDRAM refresh cycle 1: Enable low-priority SDRAM refresh cycle When this bit is enabled, SDRAM Controller will issue SDRAM refresh cycle requests with the lowest priority to fully utilize the available memory bandwidth. Whenever SDRAM refresh counter is counting over HALF of the period of SDRAM refresh cycles (MCR0C [15:8]), SDRAM Controller will automatically issue low-priority SDRAM refresh cycle requests. The maximum number of refresh cycles is 8 times per issued request. Low-Priority refresh cycles will stop whenever other requests are pending for execution.

to next page

from previous page

4	RW	**Force all banks to be pre-charged before refresh cycles** 0: Disabled 1: Enabled In general, SDRAM will only pre-charge the banks needed to be pre-charged before refresh cycle. When this bit is enabled, SDRAM will pre-charge all the banks, no matter the bank status. This register is designed for insurance policy only.
3:0	RW	**Refresh cycles per refresh period** 0000: Disable refresh cycles 0001: 1 refresh cycle per refresh period 0010: 2 refresh cycles per refresh period .... 1xxx: 8 refresh cycles per refresh period DRAM Read data will be valid only if refresh is enabled, else the read back data will be random value.

Offset: 10h — MCR10: Normal Speed AC Timing Register #1 — Init = 0
Offset: 14h — MCR14: Low Speed AC Timing Register #1 — Init = 0

Bit	Attr.	Description
31:28	RW	**t-RP timing setting** 0000: 2T 0001: 3T 1111: 17T
27:24	RW	**t-RRD timing setting (active-to-active)** 0000: 1T 0001: 2T 1111: 16T
23:20	RW	**t-RCD timing setting (active-to-read/write)** 0000: 2T 0001: 3T 1111: 17T
19:16	RW	**t-APD timing setting** 0000: 1T 0001: 2T .... 1111: 16T This timing setting determines the number of delay cycles from ACT/PRE command to read/write command between different banks.
15:12	RW	**t-RTP timing setting (read to pre-charge)** 0000: 1T 0001: 2T .... 1111: 16T This timing setting determines the number of delay cycles from read command to pre-charge command of the same bank.

to next page

from previous page

11:8	RW	**t-WTP Timing Setting (write to pre-charge)** 0000: 1T 0001: 2T .... 1111: 16T This timing setting determines the number of delay cycles from write command to pre-charge command of the same bank.
7:4	RW	**t-RTW Timing Setting (Read to Write)** 0000: 2T 0001: 3T .... 1111: 17T This timing setting determines the number of delay cycles from read command to write command.
3:0	RW	**t-WTR Timing Setting (Write to Read)** 0000: 2T 0001: 3T .... 1111: 17T This timing setting determines the number of delay cycles from write command to read command.

Offset: 18h — MCR18: Normal Speed AC Timing Register #2 — Init = 0
Offset: 1Ch — MCR1C: Low Speed AC Timing Register #2 — Init = 0

Bit	Attr.	Description
31:30		**Reserved (0)**
29:24	RW	**t-XSNR timing setting** 000000: Disabled 000001: Disabled 000010: 3T 000011: 4T .... 111111: 64T This value must be larger than 2.
23:21	RW	**Write latency timing setting** 000: 1T 001: 2T 010: 3T 011: 4T 100: 5T For DDR SDRAM, write latency timing is always 1T. For DDR2 SDRAM, write latency time is equal to CAS latency (CL) - 1T.
20:16	RW	**t-RAS timing setting (active to minimum precharge timing)** 00000: 1T 00001: 2T 11111: 32T
15:12		**Reserved (0)**

to next page

from previous page

11:8	RW	**tMRD Mode Set Interval** 0000: 1T 0001: 2T 1111: 16T
7:6		**Reserved (0)**
5:0	RW	**t-RFC refresh interval timing setting** 000000: 2T 000001: 3T 111111: 65T

Offset: 20h — MCR20: Normal Speed Delay Control Register — Init = 0
Offset: 24h — MCR24: Low Speed Delay Control Register — Init = 0

Bit	Attr.	Description
31:24		**Reserved (0)**
23	RW	**DQS window size** 0 : Small mode Window size = 1T for burst length = 2 Window size = 2T for burst length = 4 1 : Big mode Window size = 2T for burst length = 2 Window size = 3T for burst length = 4
22:21	RW	**DQS window mode** 00: Normal mode 01: Extend 0.5T 10: Delay 0.5T 11: Invalid
20:18	RW	**Window enable delay from read command to DQS** 000: 1T 001: 2T 110: 7T 111: 8T
17	RW	**Memory read data latch clock and edge selection** 0 : (DQS positive edge + MCLK positive edge) and (DQS negative edge + MCLK negative edge) 1 : (DQS positive edge + MCLK negative edge) and (DQS negative edge + MCLK positive edge)
16	RW	**CK/CKN output phase selection** 0 : Normal phase 1 : Inverted phase
15:12	RW	**CK/CKN output delay value** This register is effective only when DLL is disabled. The delay time is around 0.3ns + (CK/CKN output window delay value) * 0.25ns
11:8	RW	**DQS read window delay** The delay time is around 0.3ns + (DQS read window delay value) * 0.25ns
7:4	RW	**DQS feedback delay value** This register is effective only when DLL is disabled. The delay time is around 0.3ns + (DQS feedback delay value) * 0.25ns

to next page

from previous page

3:0	RW	**DQS output delay value** This register is effective only when DLL is disabled. The delay time is around 0.3ns + (DQS output delay value) * 0.25ns

Offset: 28h	MCR28: Mode Setting Control Register	Init = 0
Bit	**R/W**	**Description**
31:3		**Reserved (0)**
2:1	RW	**Mode register selections** 00 : MRS : Normal mode register 01 : EMRS : Extended mode register 1 10 : EMRS2 : Extended mode register 2 11 : EMRS3 : Extended mode register 3
0	RW	**Fire mode register setting and status flag** 0: No fire/command setting done flag 1: Fire command Set this bit to '1' will fire mode register setting. When finished, HW will automatically clear this bit to 0. Then SW can do the next mode setting command; HW will automatically control the timing requirement. Before this command has been done, AHB bus will be locked to prevent other command entering SDRAM controller, so SW can set the command continuously without delay.

Offset: 2Ch	MCR2C: MRS/EMRS2 Mode Setting Register	Init = X
Bit	**R/W**	**Description**
31:29		**Reserved (0)**
28:16	RW	**EMRS2**
15:13		**Reserved (0)**

to next page

from previous page

12: 0	RW	**Mode Register Setting (MRS)** The definition of this register depends on the populated SDRAM type. **For DDR SDRAM type**: Bit [12:9] (all '0') Bit [8] (DLL Reset) Bit [7] (Test Mode) Bit [6:4] (CAS Latency) 010: CAS latency = 2T 011: CAS latency = 3T others: Invalid Bit [3] (Burst Type) 0: Sequential burst type 1: Interleaving burst type (Not Supported) Bit [2:0] (Burst Length) 001: Burst length = 2 010: Burst length = 4 011: Burst length = 8 others : Invalid **For DDR2 SDRAM type**: Bit [12] (Active power down exit time) 0: Fast exit, use t-XARD 1: Slow exit, use t-XARDS Bit [11:9] (Write Recovery for auto precharge) 001 : 2T 010 : 3T 011 : 4T 100 : 5T 101 : 6T others : Invalid Bit [8] (DLL Reset) Bit [7] (Test Mode) Bit [6:4] (CAS Latency) 010: CAS latency = 2T 011: CAS latency = 3T 100: CAS latency = 4T 101: CAS latency = 5T 110: CAS latency = 6T others : Invalid Bit [3] (Burst Type) 0 : Sequential 1 : Interleave (Not Supported) Bit [2:0] (Burst Length) 010 : Burst 4 011 : Burst 8 others : Invalid

Offset: 30h		MCR30: EMRS/EMRS3 Mode Setting Register — Init = X
Bit	**R/W**	**Description**
31:29		**Reserved (0)**
28:16	RW	**EMRS3**
15:13		**Reserved (0)**

to next page

from previous page

12: 0	RW	**Extended Mode Register Setting (EMRS)** The definition of this register depends on the populated SDRAM type. **For DDR SDRAM type**: Bit [12:2] : Reserved (0) Bit [1] (Output Drive Strength) 0: Select normal driving strength 1: Select weak driving strength Bit [0] (Disable DLL) 0: Enable DLL 1: Disable DLL **For DDR2 SDRAM type**: Bit [12]: Reserved (0) Bit [11]: Enable RDQS Enable (Not supported) Bit [10] (DQS# Control) 0 : Enable 1 : Disable Bit [9:7] (OCD calibration program) 000: OCD calibration mode exit, maintain setting 001: Drive (1) 010: Drive (0) 100: Adjust mode 111: OCD calibration default Bit [6,2] (On-Die Terminator (ODT) resistance control) 00: Disable ODT 01: 75 ohm 10: 150 ohm 11: Invalid Bit [5:3] (Additive Latency) 000: 0 001: 1 (Not supported) 010: 2 (Not supported) 011: 3 (Not supported) 100: 4 (Not supported) others: Invalid Bit [1] (Output driver impedance control) 0: Select normal driving impedance (100%) 1: Select weak driving impedance (60%) Bit [0] (DLL Disable) 0: Enable DLL 1: Disable DLL

Offset: 34h		**MCR34: Power Control Register** Init = 0
Bit	**R/W**	**Description**
31	R	**Current Clock Speed Mode** **(for debugging purpose only)** 0 : Normal clock speed 1 : Low clock speed

to next page

from previous page

30:28	R	**Auto Slow-down Clock Control State Machine Status (for debugging purpose only)** 000 : Idle 001 : wait entering self refresh state 011 : switching clock speed and wait stable 010 : exiting self refresh state 110 : issue reset DLL MRS command 111 : wait DLL recovery 101 : issue normal MRS command 100 : finished
27	R	**Current CKE pin output value (for debugging purpose only)**
26:24	R	**Self Refresh Control State Machine Status (for debugging purpose only)** 000 : Idle 001 : wait memory controller all parts entering idle state 010 : entering self refresh state 011 : wait tXSNR for starting refresh command 100 : wait MRS DLL reset command 101 : wait 512 clock cycle for DLL recovery
23		**Reserved (0)**
22	RW	**Clock switch mode selection** 0: Manual mode. S/W must do the entire clock slow down procedures. 1: Auto mode. H/W will do all the clock slow down procedures.
21	RW	**Clock speed selection** 0 : Select normal speed clock mode 1 : Select low speed clock mode Setting this bit will force SDRAM Controller to switch clock speed between normal speed mode and low speed mode according to the related register parameters.
20	RW	**Clock switch control** 0: Disable or flag for clock switch done 1: Enable clock switch For auto mode, this bit will be cleared by hardware automatically when clock switch has been finished. For manual mode, SW must clear this bit.
19:17	RW	**Slow clock frequency selection** 0 : divided by 2 1 : divided by 4 2 : divided by 6 3 : divided by 8 4 : divided by 10 5 : divided by 12 6 : divided by 14 7 : divided by 16 Slow speed clock is directly derived from the divided clock of normal speed clock. The support of slow speed clock is for reducing power consumption in standby mode.
16	RW	**ODT Auto-OFF Post-amble Time** 0 : 0T after DQ/DQS leaving driving active 1 : 1T after DQ/DQS leaving driving active
15	RW	**ODT Auto-ON Preamble Time** 0: 2T before DQ/DQS driving active 1: 3T before DQ/DQS driving active, only valid for tWL > 2T

to next page

from previous page

14	RW	**Enable internal ODT auto-ON/OFF for read commands** 0: Disable auto mode 1: Auto turns ON/OFF Internal ODT when read cycles. (ODT must be enabled firstly.)
13	RW	**Enable internal ODT auto-ON/OFF for write commands** 0: Disable auto mode 1: Auto turn ON/OFF internal ODT when write cycles. (ODT must be enabled firstly.)
12	RW	**Enable SDRAM ODT auto-ON/OFF for read commands** 0: Disable auto mode 1: Auto turn ON/OFF SDRAM ODT for read cycles. (ODT must be enabled firstly.)
11	RW	**Enable SDRAM ODT auto-ON/OFF for write commands** 0: Disable auto mode 1: Auto turn ON/OFF SDRAM ODT for write cycles. (ODT must be enabled firstly.)
10	RW	**Enable SDRAM ODT** 0: Disable 1: Enable
9:7	RW	**CKE signal delay from power down mode to active command** 000: 1T 001: 2T 111: 8T
6	RW	**Disable SDRAM read buffer power saving control** 0 : SDRAM read buffers are disabled when no read command 1 : SDRAM read buffers are always enabled
5		**Reserved (0)**
4	RW	**Disable all control signals output when entering self refresh mode** 0: Enable (Output buffers are always enabled) 1: Disable (Output buffers are ON only when not in self refresh mode)
3	RW	**Disable CLK/CLKn output when entering self refresh mode** 0 : Enable (Output buffers are always enabled) 1 : Disable (Output buffers are ON only when not in self refresh mode)
2	RW	**Force SDRAM to enter self refresh mode** 0 : Normal mode or exit from self refresh mode 1 : Force SDRAM to enter self refresh mode
1	RW	**Enable auto power down function** 0: Disable auto power down function 1: Enable auto power down function when no SDRAM request When enabling auto power down function, SDRAM Controller will dynamically drive CKE signal to control the power consumption of SDRAM chips.
0	RW	**SDRAM CKE Enable** 0 : Disable CKE function (force CKE signal at 0 state after power-on reset) 1 : Enable CKE function

Offset: 38h		MCR38: Page Miss Latency Mask Register Init = 0
Bit	**R/W**	**Description**
31:3	RW	**Page miss latency mask** Bit [3]: Page miss latency mask bit for REQ0 Bit [4]: Page miss latency mask bit for REQ1 .... 0: This request will not be masked forever 1: This request will be masked (when Page Miss Counter is over page miss latency threshold value MCR38 [2:0]). This register is designed to protect high priority requests, especially like CRT refresh request which, if suffering long latency time from waiting many low priority requests pending in the request queue, may cause serious screen noise. Therefore, high priority requests are usually not masked so that they will not be ignored even when Page Miss Counter is over page miss latency threshold value.
2:0	RW	**Page miss latency threshold value** This register is designed to control the page miss rate in Request Queue

Offset: 3Ch		MCR3C: Priority Group Setting Register Init = 0
Bit	**R/W**	**Description**
n	RW	**Priority setting for REQ(n) and REQ(n+1)** 0: Priority of REQ(n) > Priority of REQ(n+1) 1: Priority of REQ(n) = Priority of REQ(n+1)
...	RW	
1	RW	**Priority setting for REQ(0) and REQ(1)** 0: Priority of REQ(1) > Priority of REQ(2) 1: Priority of REQ(1) = Priority of REQ(2)
0	RW	**Priority setting for REQ(0) and REQ(1)** 0: Priority of REQ(0) > Priority of REQ(1) 1: Priority of REQ(0) = Priority of REQ(1)

Offset: 40h Offset: 44h Offset: 48h		MCR40: Maximum Grant Length Register #1 Init = 0 MCR44: Maximum Grant Length Register #2 Init = 0 MCR48: Maximum Grant Length Register #3 Init = 0
Bit	**Attr.**	**Description**
95:92	RW	**MCR48 [31:28]: Reserved**
91:88	RW	**MCR48 [27:24]: Maximum grant length for REQ22**
87:84	RW	**MCR48 [23:20]: Maximum grant length for REQ21**
83:80	RW	**MCR48 [19:16]: Maximum grant length for REQ20**
79:76	RW	**MCR48 [15:12]: Maximum grant length for REQ19**
75:72	RW	**MCR48 [11:08]: Maximum grant length for REQ18**
71:68	RW	**MCR48 [07:04]: Maximum grant length for REQ17**
67:64	RW	**MCR48 [03:00]: Maximum grant length for REQ16**
63:60	RW	**MCR44 [31:28]: Maximum grant length for REQ15**
59:56	RW	**MCR44 [27:24]: Maximum grant length for REQ14**
55:52	RW	**MCR44 [23:20]: Maximum grant length for REQ13**
51:48	RW	**MCR44 [19:16]: Maximum grant length for REQ12**

to next page

from previous page

47:44	RW	**MCR44 [15:12]: Maximum grant length for REQ11**
43:40	RW	**MCR44 [11:08]: Maximum grant length for REQ10**
39:36	RW	**MCR44 [07:04]: Maximum grant length for REQ9**
35:32	RW	**MCR44 [03:00]: Maximum grant length for REQ8**
31:28	RW	**MCR40 [31:28]: Maximum grant length for REQ7**
27:24	RW	**MCR40 [27:24]: Maximum grant length for REQ6**
23:20	RW	**MCR40 [23:20]: Maximum grant length for REQ5**
19:16	RW	**MCR40 [19:16]: Maximum grant length for REQ4**
15:12	RW	**MCR40 [15:12]: Maximum grant length for REQ3**
11:8	RW	**MCR40 [11:08]: Maximum grant length for REQ2**
7:4	RW	**MCR40 [07:04]: Maximum grant length for REQ1**
3:0	RW	**MCR40 [03:00]: Maximum grant length for REQ0**

Note :
SDRAM Controller totally supports up to 23 SDRAM requests (REQ0 ~ REQ22).
The maximum grant length of each request can be programmed by 4 bits assigned by registers MCR48 ~ MCR40. The maximum grant length is defined by the following table:

Bit[3:0]	Maximum Grant Length
0, 1	2 times
2, 3	4 times
4, 5	6 times
6, 7	8 times
8, 9	10 times
10, 11	12 times
12, 13	14 times
14, 15	16 times

Register MCR48 ~ MCR40 defines the maximum grant length allowed to be put into SDRAM request FIFO for each SDRAM request. Properly setting of these registers can effectively control the bandwidth allocations for different SDRAM requests.

Offset: 60h — MCR60: IO Buffer Mode Register — Init = 0

Bit	R/W	Description
31:28		**Reserved (0)**
27:26	RW	**Reserved, must keep at value 0**
25	RW	**Enable SDRAM IO pins DQ[15:8], DM1, DQS1, DQS1n** Please reference the description for MCR60[24]
24	RW	**Enable SDRAM IO pins DQ[7:0], DM0, DQS0, DQS0n** Enabling SDRAM IO pins provide the flexibility to control the function of each 8-bit data bus. When disabled, the corresponding IO pins will be turned OFF (disabling output buffer, gating input buffer, turning off ODT). Therefore, the power consumption of each IO buffer can be effectively controlled.
23	RW	**DDR IO LVCMOS selection** 0: For 2.5V SSTL2 class I, class II, or 1.8V SSTL18 receiving applications 1: For 3.3V LVTTL, 1.8V MDDR receiving applications
22	RW	**DDR IO DS selection** 0: For SSTL18 differential receiving applications (1.8V DDR) 1: For SSTL2 differential receiving applications (2.5V DDR)

to next page

from previous page

21:20	RW	**Programmable driving strength for pin ODT** The two register bits (S1, S0) can be used to program the driving strength for the designated DDR IO buffers. Mode / IOH/IOL (mA) / S1 / S0: SSTL18 Full Drive / 13.4 / 0 / 0 SSTL2 Class II / 16.2 / 0 / 0 MDDR (10mA) / 10.0 / 0 / 0 MDDR (8mA) / 8.0 / 0 / 1 SSTL18 Half Drive / 6.7 / 1 / 0 SSTL2 Class I / 8.1 / 1 / 0 MDDR (4mA) / 4.0 / 1 / 0 LVTTL High Drive / 16.0 / 1 / 0 MDDR (2mA) / 2.0 / 1 / 1 LVTTL Low Drive / 12.0 / 1 / 1
19:18	RW	**Programmable driving strength for pins CS, RASN, CASN, WEN, CKE, MA, BA** Please reference the table for MCR60[21:20]
17:16	RW	**Programmable driving strength for pins CK, CKN** Please reference the table for MCR60[21:20]
15:14	RW	**Programmable driving strength for pins DQS, DQSn** Please reference the table for MCR60[21:20]
13:12	RW	**Programmable driving strength for pins DQ, DM** Please reference the table for MCR60[21:20]
11:10	RW	**ODT mode for pin ODT** These two register bits (A6, A2) can be used to program the resistance of On-Die Terminator (ODT). A6 / A2 / Resistance of Terminator: 0 / 0 / Disabled ODT 0 / 1 / 75 ohm ODT 1 / 0 / 150 ohm ODT 1 / 1 / Invalid
9:8	RW	**ODT mode for pins CS, RASN, CASN, WEN, CKE, MA, BA** Please reference the table for MCR60 [11:10]
7:6	RW	**ODT mode for pins CK/CKN** Please reference the table for MCR60 [11:10]
5:4	RW	**ODT mode for pins DQS/DQSn** Please reference the table for MCR60 [11:10]
3:2	RW	**ODT mode for pins DM** Please reference the table for MCR60 [11:10]
1:0	RW	**ODT mode for pins DQ** Please reference the table for MCR60 [11:10]

Offset: 64h	**MCR64: DLL Control Register #1**	**Init = 0**
Bit	**R/W**	**Description**
31:25		**Reserved (0)**

to next page

from previous page

24	RW	**DLL3 reference clock selection (for CK/CKn)** 0: Select MCLK as the reference clock of DLL3 1: Select MCLK/2 as the reference clock of DLL3
23	RW	**Reserved**
22	RW	**DLL1 reference clock selection (for DQS1 and DQS0)** 0: Select MCLK as the reference clock of DLL1 1: Select MCLK/2 as the reference clock of DLL1
21	RW	**DLL3 reset control (for CK/CKn and DQS output phase)** 0 : Reset 1 : Normal operation
20	RW	**Reserved, must keep at value 0**
19	RW	**DLL1 reset control (for DQS1 and DQS0 input phase)** 0 : Reset 1 : Normal operation
18	RW	**DLL3 power down control (for CK/CKn and DQS output phase)** 0 : Power down 1 : Normal operation
17	RW	**Reserved, must keep at value 0**
16	RW	**DLL1 power down control (for DQS1 and DQS0 input phase)** 0 : Power down 1 : Normal operation
15:8	RW	**DLL3 output phase value SADJ (for CK/CKn output)** Please reference MCR6C Notes
7:0	RW	**DLL3 output phase value SADJ (for DQS output)** Please reference MCR6C Notes

Offset: 68h — MCR68: DLL Control Register #2 — Init = 0

Bit	R/W	Description
31:16	RW	**Reserved**
15:8	RW	**DLL1 input phase value SADJ (for DQS1)** Please reference MCR6C Notes
7:0	RW	**DLL1 input phase value SADJ (for DQS0)** Please reference MCR6C Notes

Offset: 6Ch — MCR6C: DLL Control Register #3 — Init = 0

Bit	R/W	Description
31:30		**Reserved (0)**
29:24	RW	**Reserved**
23:16	RW	**DLL3 master adjustment value MADJ**
15:8	RW	**Reserved**
7:0	RW	**DLL1 master adjustment value MADJ**

to next page

from previous page

Note :
DLL adjustment algorithm is list below:

Minimum MADJ value = 40

And the value of MADJ will affect the operating frequency, the relation is :

MIN Frequency = MIN_DDR_FREQ(67) * NOM_ADJ(120) / MADJ
MAX Frequency = MAX_DDR_FREQ(347) * NOM_ADJ(120) / MADJ

The adjusted output delay value would be :

delay value (ns) = (SADJ + 24) / MADJ * Tref + 0.1ns

where Tref = period of MCLK

Offset: 70h		**MCR70: Testing Control/Status Register** **Init = 0**
Bit	**R/W**	**Description**
31:16	R	**Testing Fail Count** This value shows the accumulated fail count. It is record once for each 64 bits data unit. This value will not overflow if the error numbers over the maximum count.
15:8		**Reserved (0)**
7	R	**Testing result flag** 0: Pass 1: Fail This flag will be cleared whenever disabling SDRAM tests. AST2050 / AST1100 provides a sequential logic that can effectively test SDRAM memory in a very short period of time. The memory range to be tested is programmable (MCR74 and MCR78). This module can be a good instrument for SDRAM stress tests or SDRAM self tests during boot-up process.
6	R	**Testing finish flag** 0: Busy 1: Finish This flag will be cleared whenever disabling SDRAM tests.
5:3	RW	**Testing data generation mode** 000: Byte Toggled data, 0 → 1 → 0 → 1 001: Byte Rotate left, LSB → MSB 010: Byte Rotate right, MSB → LSB 011: Byte Increment at each 8 bits unit 100: Byte sequence data 101: 4 bit sequence data 110: 2 bit sequence data 111: 1 bit sequence data The initial value of testing data will be from MCR7C. The sequential testing data will be generated based on the above selected mode.
2:1	RW	**Testing mode** 00: Write memory only (testing result flag is always 0 after testing) 01: Read back and compare for each location 10: Write one memory location first then read back the location and compare with the expected value 11: Loop back test (write memory data through SDRAM output/input buffers then compare with expected value)

to next page

from previous page

0	RW	**Enable testing** 0: Disable, reset test function and related status 1: Enable Since SDRAM testing will impact normal graphics display functions, Its not recommended to enable this function after Watchdog Timer reset. Enabling SDRAM testing after power-on reset should be a right time frame.

Offset: 74h	**MCR74: Testing Start Address and Length Register**	**Init = 0**
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:23	RW	**Testing start address base** This value defines the testing base address segment. It is defined at 8MB base.
22:3	RW	**Testing total length** This value defines the testing final address (8 bytes boundary). Testing starts from offset 0 (relative to the base address segment) and ends at this final address. The maximum testing space can up to 8MB.
2:0		**Reserved (0)**

Offset: 78h	**MCR78: Testing Fail DQ Bit Register**	**Init = 0**
Bit	**R/W**	**Description**
31:0	R	**Fail DQ bit position** Bit 0 : DQ0 Bit 1 : DQ1 .... Bit31 : DQ31

Offset: 7Ch		MCR7C: Test Initial Value Register Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Initial value** **When Data Generation Mode = 000, 001, 010, 011** 07:0 : Initial Value for positive edge DQ 15:8 : Initial Value for negative edge DQ The values are the same for all DQS group. **When Data Generation Mode = 100** 07:0, 23:16 = Value for positive edge DQ and turn by sequence. 15:8, 31:24 = Value for negative edge DQ and turn by sequence. The values are the same for all DQS group. **When Data Generation Mode = 101** 3:0, 11:08, 19:16, 27:24 = Value for positive edge DQ and turn by sequence. 7:4, 15:12, 23:20, 31:28 = Value for negative edge DQ and turn by sequence. The values are the same for each 4 bits DQ. **When Data Generation Mode = 110** 1:0, 5:4, 09:08, 13:12, 17:16, 21:20, 25:24, 29:28 = Value for positive edge DQ and turn by sequence. 3:2, 7:6, 11:10, 15:14, 19:18, 23:22, 27:26, 31:30 = Value for negative edge DQ and turn by sequence. The values are the same for each 2 bits DQ. **When Data Generation Mode = 111** 0, 2, 4, 6, 8, 10, 12, 14 = Value for positive edge DQ and turn by sequence. 1, 3, 5, 7, 9, 11, 13, 15 = Value for negative edge DQ and turn by sequence. The values are the same for each 1-bit DQ.

Offset: 100h		MCR100: AST2000 Backward Compatible SCU Password Init = 0x000000A8
Bit	**R/W**	**Description**
31:0	R	**0x000000A8**

Offset: 120h		MCR120: AST2000 Backward Compatible SCU MPLL Parameter Init = 0
Bit	**R/W**	**Description**
31:16		**Reserved (0)**
15:14	RW	**Post Divider**
13:5	RW	**Numerator**
4:0	RW	**Denumerator**

Offset: 170h MCR170: AST2000 Backward Compatible SCU Hardware Strapping Value Init = 0		
Bit	**R/W**	**Description**
31:0	R	**All ’0’.**

17.4 Address Arrangement

17.4.1 Address Translation

The maximum supported addressing space of AST2050 / AST1100 is 256 MB, thus 28 bits address internally. The SDRAM address is coordinated as Row address(Page), Bank address and Column address. The translation from internal address to the SDRAM coordinate is shown below:

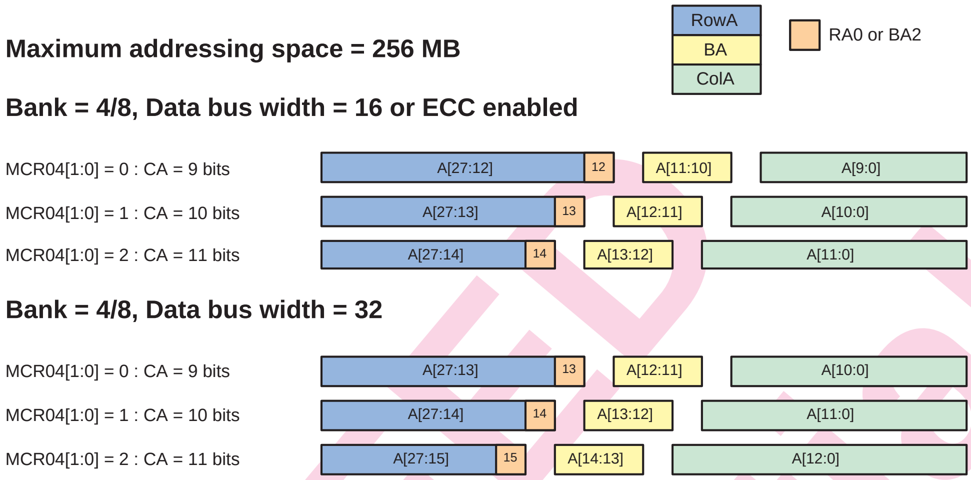


Figure 66: Address Translation Mapping

There is no protection or masking about the out-of-range address accessing. The maximum address outputed to the SDRAM would be 256 MB or constrainted by the setting of CA and BA bits in MCR04. When access to an address over the SDRAM size, the MSB will be ignored by the SDRAM, and targeting to a space with the same low bits address.

17.4.2 Graphics Memory Base Address

The Graphics (VGA) memory segment internal base address is defined as following:

MCR[5:4]=0, 8 MB   Base = 0xF80_0000
MCR[5:4]=1, 16 MB   Base = 0xF00_0000
MCR[5:4]=2, 32 MB   Base = 0xE00_0000
MCR[5:4]=3, 64 MB   Base = 0xC00_0000

For the out-of-range defined address bits, they are ignored by the SDRAM themselves.

17.5 Data Arrangement

The internal data bus width of AST2050 / AST1100 is always 64 bits. This will lead the addressing method of IPs to be 64 bits alignment. The following shows the data mapping between internal and external data bus:

17.5.1 16 Bits Mode

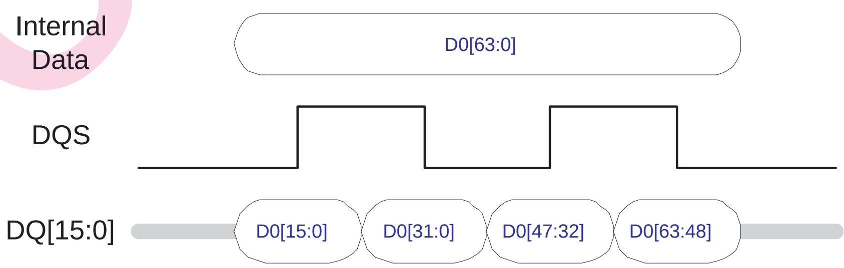


Figure 67: 16 Bits Data Mapping

17.6 Memory Clock Switch Control

17.6.1 Switch from normal speed to low speed with DLL enabled

1. Set MCR14, MCR1C and MCR24 with the low speed AC parameters
2. Set MCR2C with the low speed Mode register
3. Set MCR34 bit[22:21] = 0x3 and bit[19:17] to the slow down ratio, all other bits keep the old value.
4. Set MCR34 bit[20] = 1 and all other bits keep the old value to enable auto clock slow down switch.

17.6.2 Switch from low speed with DLL enabled to normal speed

1. Set MCR2C with the normal speed Mode register
2. Set MCR34 bit[22:21] = 0x2, all other bits keep the old value.
3. Set MCR34 bit[20] = 1 and all other bits keep the old value to enable auto clock switch.

17.7 Self Refresh Command Sequence

17.7.1 Enter Self Refresh

1. Disable all IP working and swap ARM code area to static flash memory
2. Set MCR34 bit[2] = 1 and all other bits keep old value, bit[4:3] is an option for more power saving

17.7.2 Exit Self Refresh

1. Set MCR34 bit[2] = 0 and all other bits keep old value
2. Reset DRAM DLL. Set MCR2C bit[8] = 1 and all other bits keep old value then set MCR28 = 1
3. Disable reset DRAM DLL. Set MCR2C bit[8] = 0 then set MCR28 = 1

18 System Control Unit (SCU)

18.1 Overview

System Control Unit (SCU) implements chip level control registers, which is listed below, to control the various functions supported by AST2050 / AST1100 . Each register has its own specific offset value to derive its physical address location.

Base address of SMC = 0x1E6E_2000
Physical address = (Base address of SCU Controller) + Offset

SCU00: Protection Key Register
SCU04: System Reset Control Register
SCU08: Clock Selection Register
SCU0C: Clock Stop Control Register
SCU10: Frequency Counter Control Register
SCU14: Frequency Counter Measurement Register
SCU18: Interrupt Control and Status Register
SCU1C: 32.768 KHz Error Correction Register
SCU20: M-PLL Parameter Register
SCU24: H-PLL Parameter Register
SCU28: Frequency counter comparison range
SCU2C: Misc. Control Register
SCU30: PCI Configuration Setting Register #1
SCU34: PCI Configuration Setting Register #2
SCU38: PCI Configuration Setting Register #3
SCU3C: System Reset Control Register
SCU40: SOC Scratch Register #1
SCU44: SOC Scratch Register #2
SCU50: VGA Scratch Register #1
SCU54: VGA Scratch Register #2
SCU58: VGA Scratch Register #3
SCU5C: VGA Scratch Register #4
SCU60: VGA Scratch Register #5
SCU64: VGA Scratch Register #6
SCU68: VGA Scratch Register #7
SCU6C: VGA Scratch Register #8
SCU70: Hardware Trapping Register
SCU74: Multi-function Pin Control #1
SCU78: Multi-function Pin Control #2
SCU7C: Silicon Revision ID Register

Changing SCU registers usually results in significant impact on SOC operations. Therefore, all these registers have to be well protected.

18.2 Registers : Base Address = 0x1E6E:2000

Offset: 00h		SCU00: Protection Key Register Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Protection Key** This register is designed to protect SCU registers from unpredictable updates, especially when ARM CPU is out of control; even under such a circumstance, properly protecting SCU registers can make sure that Graphic Display Controller are still able to successfully access SDRAM for displaying graphics correctly. The password of the protection key is **0x1688A8A8**. Reading back SCU registers is irrelevant with this register. **Unlock SCU registers**: Write 0x1688A8A8 to this register **Lock SCU registers**: Write others value to this register Whenever finished the initialization of SCU registers, please always set SCU registers into **locked** mode. The initial state of this register is at locked mode. When this register is unlocked, the read back value of this register is 0x00000001. When this register is locked, the read back value of this register is 0x00000000.

Offset: 04h		SCU04: System Reset Control Register Init = 0x000FFE5C
Bit	**R/W**	**Description**
31:22		**Reserved (0)**
21	RW	**PCI Host Reset Output Enable Control** 0: Disable output, input mode (default) 1: Enable output This bit controls the PCI Reset Pin (BRST#) I/O direction.
20	RW	**Force PCI Host Reset Output High** 0: No operation, PCI Host Reset output is controlled by Bit[19]. (default) 1: Force PCI Host Reset pin output value to high. This bit is used to force reset pin output keeping at high while reset internal host controller by setting Bit[19] = 1.
19	RW	**Reset PCI Host Bus Controller** 0: No operation 1: Reset PCI Host Controller (asynchronous reset) (default) This bit controls the PCI Host Bus reset, both internal controller and BRST# pin output.
18	RW	**Reset MIC Controller** 0: No operation 1: Reset MIC Controller (asynchronous reset) (default)
17	RW	**Reserved, must keep at value "1"**
16	RW	**Reset MDMA Controller** 0: No operation 1: Reset MDMA Controller (asynchronous reset) (default)
15	RW	**Reserved, must keep at value "1"**
14	RW	**Reset USB2.0 Controller** 0: No operation 1: Reset USB2.0 Controller (asynchronous reset) (default)
13	RW	**Reserved, must keep at value "1"**

to next page

from previous page

12	RW	**Reset MAC#2 Controller** 0: No operation 1: Reset MAC#2 Controller (asynchronous reset) (default)
11	RW	**Reset MAC#1 Controller** 0: No operation 1: Reset MAC#1 Controller (asynchronous reset) (default)
10	RW	**Reset PECI controller** 0: No operation 1: Reset PECI Controller (default)
9	RW	**Reset PWM controller** 0: No operation 1: Reset PWM Controller (default)
8	RW	**Reset PCI Salve and VGA Controller** 0: No operation (default) 1: Reset PCI Slave and VGA Controller
7	RW	**Reserved, must keep at value ”0”**
6	RW	**Reset Video Engine** 0: No operation 1: Reset Video Engine (asynchronous reset) (default)
5	RW	**Reset LPC Controller** 0: No operation (default) 1: Reset LPC Controller (asynchronous reset) The reset command will be applied to both LPC Controller and the BMC controller embedded in LPC Controller.
4	RW	**Reset HAC Engine** 0: No operation 1: Reset HAC Engine (asynchronous reset) (default)
3	RW	**Reserved, must keep at value ”1”**
2	RW	**Reset I2C/SMBus Controller** 0: No operation 1: Reset SMBus/I2C Controller (asynchronous reset) (default) Writing ”0” to this register will cause all the 7 sets of SMBus/I2C Controllers to be **asynchronously** reset. Actually each of the 7 controllers has its own control register to **synchronously** reset itself.
1	RW	**Reset AHB Bridges** 0: No operation (default) 1: Reset all the AHB related bridges This register will reset the following three bridges. AHB—to—M-Bus Bridge AHB—to—APB Bridge AHB—to—P-Bus Bridge Writing ”0” to this register is necessary to resume the operations of the three bus bridges.

to next page

from previous page

0	RW	**Reset SDRAM Controller** 0: No operation (default) 1: Reset SDRAM Controller (asynchronous reset) Writing "1" to this bit will directly cause SDRAM controller to be asynchronously reset. Firmware code needs to write "0" to this bit to resume SDRAM Controller. Issuing this reset command should be very careful. It might cause the data stored in SDRAM lost. It might cause SOC malfunction. Firmware code needs to handle all the potential side effects it might introduce after resetting this controller.

Offset: 08h	SCU08: Clock Selection Register	Init = 0xE3F00070
Bit	**R/W**	**Description**
31:29	RW	**LPC Master LHCLK divider selection** 000: LHCLK = H-PLL/2 001: LHCLK = H-PLL/4 010: LHCLK = H-PLL/6 011: LHCLK = H-PLL/8 100: LHCLK = H-PLL/10 101: LHCLK = H-PLL/12 110: LHCLK = H-PLL/14 111: LHCLK = H-PLL/16
28	RW	**LPC Master LHCLK clock generation/output enable control** 0: Disable, LHCLK is from external LCLK pin 1: Enable, LHCLK is generated and output internally
27:26	RW	**Reserved, don't use**
25:23	RW	**APB Bus PCLK divider selection** 000: PCLK = H-PLL/2 001: PCLK = H-PLL/4 010: PCLK = H-PLL/6 011: PCLK = H-PLL/8 100: PCLK = H-PLL/10 101: PCLK = H-PLL/12 110: PCLK = H-PLL/14 111: PCLK = H-PLL/16 There is a limitation on the PCLK frequency allowed. a. For rev. A1 chip, when LPC master APB2LPC bridge function is used: LCLK $\geq$ PCLK. Rev. A2 had fixed this issue. b. When LPC slave channel #3 BT mode is used: PCLK $>$ 0.5*LCLK.
22:20	RW	**PCI Host BHCLK divider selection** 000: BHCLK = H-PLL/2 001: BHCLK = H-PLL/4 010: BHCLK = H-PLL/6 011: BHCLK = H-PLL/8 100: BHCLK = H-PLL/10 101: BHCLK = H-PLL/12 110: BHCLK = H-PLL/14 111: BHCLK = H-PLL/16
19	RW	**PCI Host BHCLK clock generation/output enable control** 0: Disable, BHCLK is from external BCLK pin 1: Enable, BHCLK is generated and output internally
18:17	RW	**Reserved, don't use**

to next page

from previous page

16	RW	**RTC clock source selection (for test only)** 0: The clock source of RTC is from 32.768KHz clock source 1: The clock source of RTC is from 24MHz clock source
15:11	RW	**Reserved, must keep at value ”0”**
10:8	RW	**Video port A output clock delay control bit[3:1]** Include bit[0] from SCU2C[9], 0000: Delay ∼0ns 0001: Delay ∼0.5ns 0010: Delay ∼1ns 0011: Delay ∼1.5ns 0100: Delay ∼2ns 0101: Delay ∼2.5ns 0110: Delay ∼3ns 0111: Delay ∼3.5ns 1000: Clock inversed and delay ∼0ns 1001: Clock inversed and delay ∼0.5ns 1010: Clock inversed and delay ∼1ns 1011: Clock inversed and delay ∼1.5ns 1100: Clock inversed and delay ∼2ns 1101: Clock inversed and delay ∼2.5ns 1110: Clock inversed and delay ∼3ns 1111: Clock inversed and delay ∼3.5ns
7	RW	**ARM CPU clock throttling enable** 0: Disable 1: Enable throttling
6:4	RW	**ARM CPU clock throttling setting** 000: Divided by 2 001: Divided by 4 010: Divided by 6 011: Divided by 8 100: Divided by 10 101: Divided by 12 110: Divided by 14 111: Divided by 16 This register is designed to slow down ARM CPU clock for reducing power consumption in standby mode. The clock divider is embedded with anti-glitch logic to protect CPU operations.
3:2	RW	**ECLK clock source selection** 00: The clock source of ECLK is from M-PLL clock output 01: The clock source of ECLK is from H-PLL clock output 10: The clock source of ECLK is from inverted M-PLL output 11: The clock source of ECLK is from Inverted H-PLL output Before issuing command to change this register, it had better to stop ECLK, and Video Engine must be reset in advance in order to the potential risk in changing this clock.

to next page

from previous page

1:0	RW	**MCLK source selection** 00: The clock source of MCLK is from M-PLL output clock 01: The clock source of MCLK is from H-PLL output clock 10: The clock source of MCLK is from inverted M-PLL output clock 11: The clock source of MCLK is from inverted H-PLL output clock Changing MCLK source is suggested only at the boot-up stage before initializing DRAM controller parameters, and must disable MCLK output at SCU0C[2] first, and re-enable MCLK after this register is setting done. Its not necessary to reset SDRAM controller before updating this register.

Offset: 0Ch	SCU0C: Clock Stop Control Register	Init = 0x000C3E8B
Bit	**R/W**	**Description**
31:20		**Reserved (0)**
19	RW	**Stop BHCLK (For PCI Host Controller) Clock** 0: Enable clock running 1: Stop clock running (default)
18	RW	**Reserved, must keep at value ”1”**
17:16	RW	**Reserved (0)**
15	RW	**Stop UARTCLK (For UART1/UART2 controller)** 0: Enable clock running (default) 1: Stop clock running
14	RW	**Enable USB2.0 clock** 0: Stop USB2.0 clock running, power-down USB2.0 PHY. (default) 1: Enable USB2.0 clock running The procedure to enable USB2.0 controller: 1. Enable USB2.0 clock running, wait **10 ms** for clock stable 2. Disable USB2.0 global reset by setting SCU04[14] = 0 3. Disable USB2.0 PHY reset by setting HUB00[11] = 1 4. Start using USB2.0 controller
13	RW	**Stop YCLK (For HAC)** 0: Enable clock running 1: Stop clock running (default)
12:9	RW	**Reserved, must keep at value ”1111”**
8	RW	**Stop LCLK (for LPC Controller)** 0: Enable clock running (default) 1: Stop clock running
7	RW	**Stop UCLK (For USB1.1)** 0: Enable clock running 1: Stop clock running (default)
6	RW	**REFCLK Stop Enable (For 24MHz)** 0: Enable clock running (default) 1: Stop clock running
5	RW	**Stop DCLK (For VGA)** 0: Enable clock running (default) 1: Stop clock running
4	RW	**Stop BCLK (For PCI Slave)** 0: Enable clock running (default) 1: Stop clock running

to next page

from previous page

3	RW	**Stop V1CLK (For Video Capture #1)** 0: Enable clock running 1: Stop clock running (default)
2	RW	**Stop MCLK (For SDRAM Controller)** 0: Enable clock running (default) 1: Stop clock running
1	RW	**Stop GCLK (For 2D Engine)** 0: Enable clock running 1: Stop clock running (default)
0	RW	**Stop ECLK (For Video Engine)** 0: Enable clock running 1: Stop clock running (default)

Offset: 10h		**SCU10: Frequency Counter Control Register** Init = 0
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	R	**Clock frequency measurement compare result** 0 : Fail 1 : Pass This status flag is the result by comparing the output counter value at SCU14 with the upper and lower limit defined at SCU28, if the lower_limit <= counter <= upper_limit, then the compare result is Pass.
6	R	**Clock frequency measurement finished** 0 : Not finished 1 : Finished This status flag can be cleared by setting SCU10[1] to '0'
5:2	RW	**Clock source selection for clock frequency measurement** 0000: Select delay cell (13 stages) based ring oscillator (with 1/16 clock divider) 0001: Select NAND gate (41 stages) based ring oscillator (with 1/16 clock divider) 0010: Select PCI bus clock 0011: Select D2-PLL 0100: Select M-PLL 0101: Select H-PLL 0110: Select LPC bus clock 0111: Select clock for Video Port B 1011: Select D-PLL 1111: Select clock for Video Port A This register is designed to select the clock source for clock frequency measurement.
1	RW	**Oscillator Counter Enable** 0 : Reset frequency measurement counter 1 : Enable frequency measurement counter
0	RW	**Enable Ring Oscillator** 0 : Disable ring oscillators 1 : Enable ring oscillators Before enabling the measurement of ring oscillator frequency, SW must enable this bit and wait for 1ms to make sure the ring oscillators are stable. After finished the measurement, SW must disable ring oscillators to reduce power consumption.

to next page

from previous page

Note :
The procedure to start counter:

1. Set SCU10 = 0x16
2. Wait until SCU14 = 0
3. Set SCU10[0] = 1 and SCU10[5:2] = clock for measurement
4. Delay 1ms
5. Set SCU10[1] = 1
6. Wait until SCU10[6] = 1
7. Read SCU14 and calculate the result frequency using following equation

Oscillator Counter Algorithm :
When the reference clock CLK24M count from 0 to 512, measure the OSCCLK counting value, then

OSCCLK frequency = CLK24M / 512 * (SCU14 + 1)

Offset: 14h — SCU14: Frequency Counter Measurement Register — Init = 0

Bit	R/W	Description
31:14		**Reserved (0)**
13:0	R	**Value of frequency measurement counter.** Reset to 0 automatically when starts counting Algorithm of frequency measurement: Frequency = (24MHz / 512) * (Value + 1)

Offset: 18h — SCU18: Interrupt Control and Status Register — Init = 0

Bit	R/W	Description
31:18		**Reserved (0)**
17	RW	**VGA scratch register change Interrupt and status** 0 : No interrupt occurs 1 : Interrupt occurs The status flag can be cleared by writing ’1’ to this bit.
16	RW	**VGA cursor change interrupt and status** 0 : No interrupt occurs 1 : Interrupt occurs The status flag can be cleared by writing ’1’ to this bit.
15:2		**Reserved (0)**
1	RW	**Enable VGA scratch register change interrupt** 0 : Disable Interrupt 1 : Enable Interrupt generation
0	RW	**Enable VGA cursor change interrupt** 0 : Disable Interrupt 1 : Enable Interrupt generation

Offset: 1Ch — SCU1C: 32.768 KHz Error Correction Register — Init = 0x0000001B

Bit	R/W	Description
31:8		**Reserved (0)**
7:0	RW	**Error correcting value** This register can be used to fine-tune the precision of 32.768KHz clock source.

to next page

from previous page

Note :
$RTC\ clock = 12MHz * 128/(46848 + Error\_Correction\_Value)$

$RTC\ clock = 12MHz * 128/(46848 + 27) = 32768.0Hz$
$RTC\ clock = 12MHz * 128/(46848 + 28) = 32767.3Hz(-1.8\ seconds\ per\ day = 21\ DPM)$
$RTC\ clock = 12MHz * 128/(46848 + 26) = 32768.7Hz(+1.8\ seconds\ per\ day = 21\ DPM)$

Offset: 20h — SCU20: M-PLL Parameter Register — Init = 0x00004291

Bit	R/W	Description
31:18		**Reserved (0)**
17	RW	**Enable M-PLL bypass mode** 0: No operation 1: Enable M-PLL by pass mode When enabling M-PLL bypass mode, the output clock of M-PLL is directly from the external reference clock input pin. In AST2050 / AST1100 system design, the external reference clock is 24MHz sharing with the clock source for USB devices.
16	RW	**Turn off M-PLL** 0: No operation 1: Turn Off M-PLL When M-PLL is turned off, it will enter power down mode. The output signal is always "0". M-PLL default is OFF if hardware trapping is set to boot-up at low speed (SCU70 [16]).
14:12	RW	**M-PLL Post Divider** 0xx : divide 1 100 : divide 2 101 : divide 4 110 : divide 8 111 : divide 16
10:5	RW	**M-PLL Numerator**
4	RW	**M-PLL Output Divider**
3:0	RW	**M-PLL Denumerator** M-PLL is preliminarily designed to generate the running frequency of memory controller. The output frequency of M-PLL PLL is based on the following question: (Output frequency) = 24MHz * (2-OD) * [(Numerator+2) / (Denumerator+1)] The default frequency of M-PLL settings is always 133MHz.

Offset: 24h — SCU24: H-PLL Parameter Register — Init = 0x00004291

Bit	R/W	Description
31:19		**Reserved (0)**
18	RW	**H-PLL parameter selection** 0: Select H-PLL parameters by trapping resistors 1: Select H-PLL parameters by the programmed registers (SCU24[17:0])
17	RW	**Enable H-PLL bypass mode** 0: No operation 1: Enable H-PLL bypass mode When enabling H-PLL bypass mode, the output clock of H-PLL is directly from the external reference clock input pin. In AST2050 / AST1100 system design, the external reference clock is 24MHz sharing with the clock source for USB devices.

to next page

from previous page

16	RW	**Turn off H-PLL** 0: No operation 1: Turn Off H-PLL When H-PLL is turned off, it will enter power down mode. The output signal is always ”0”.
14:12	RW	**H-PLL Post Divider** 0xx : divide 1 100 : divide 2 101 : divide 4 110 : divide 8 111 : divide 16
10:5	RW	**Numerator**
4	RW	**H-PLL Output Divider**
3:0	RW	**H-PLL Denumerator** H-PLL is preliminarily designed to generate the running frequency of ARM CPU. The output frequency of H-PLL PLL is based on the following question: (Output frequency) = 24MHz * (2-OD) * [(Numerator+2) / (Denumerator+1)] The default frequency of H-PLL settings depends on the related trapping resistors. The available options include 100/133/166/200 MHz

Offset: 28h		SCU28: Frequency counter comparison range Init = 0
Bit	**R/W**	**Description**
31:30		**Reserved (0)**
29:16	RW	**Upper Limit**
15:14		**Reserved (0)**
13:0	RW	**Lower Limit**

Offset: 2Ch		SCU2C: Misc. Control Register Init = 0
Bit	**R/W**	**Description**
31:16		**Reserved (0)**
15	RW	**Enable internal link function between UART1 and UART2** When set to ’1’, UART1 (NCTS1, NDSR1, NDTR1, NRTS1, TXD1, and RXD1) is connected to UART2 (NRTS2, NDTR2, NDSR2, NCTS2, RXD2, and TXD2). Also, NDCD1 is connected to NDTR2; NDCD2 is connected to NDTR1. Both NRI1 and NRI2 are in the idle state. Besides, UART1 still drives NRTS1, NDTR1, and TXD1. UART2 still drives NRTS2, NDTR2, and TXD2 if SCU74[24]=1.
14	RW	**Enable MUX function of UART1 pins** When set to ’1’, UART1 and UART2 can share the same UART1 pins and the connector. In the meantime, UART1 still monitor the UART1 chip input pins.
13	RW	**Timeout control bit for VUART** For the detail, reference VUART20 and VUART24.
12	RW	**Enable the reference clock divider (div13) for UART1 and UART2** 0: baud rate = 24MHz / (16*divisor) 1: baud rate = (24MHz/13) / (16*divisor)
11	RW	**Enable inverting YCLK** 0: YCLK from MCLK 1: YCLK from inverted MCLK

to next page

from previous page

10	RW	**Reserved**
9	RW	**Video port A output clock delay control bit[0]** Reference the description of SCU08[10:8
8	RW	**Disable PCI slave to AHB bus bridge** 0: Enable bridge 1: Disable bridge
7	RW	**Reserved, must keep at value "0"**
6	RW	**Disable VGA CRT display when using Video Direct Fetch mode** 0: Enable VGA CRT display 1: Disable VGA CRT display
5	RW	**Enable VGA registers access when not trapping in VGA mode** 0: VGA registers access is controlled by VGA mode trapping 1: Force enables VGA registers access
4	RW	**Reserved, must keep at value "1"**
3	RW	**Disable video DAC** 0: Enable video DAC 1: Disable video DAC
2	RW	**Disable D1-PLL** 0 : Enable D1-PLL 1 : Disable D1-PLL
1	RW	**OSC clock output pin selection (For test mode only)** 0: No OSC clock output 1: OSC clock will output from DDCACLK pin
0	RW	**Disable SMC output buffers** 0 : Enable SMC output buffers 1 : Disable SMC output buffers

Offset: 30h　　SCU30: PCI Configuration Setting Register #1　　Init = 0x20001A03

Bit	R/W	Description
31:16	RW	**PCI Device ID** The register can support firmware code the flexibility to change PCI Device ID of AST2050 / AST1100 . But changing the ID is usually not recommended.
15:0	RW	**PCI Vendor ID** The register can support firmware code the flexibility to change PCI Vendor ID of AST2050 / AST1100 . But changing the ID is usually not recommended.

Offset: 34h　　SCU34: PCI Configuration Setting Register #2　　Init = 0x20001A03

Bit	R/W	Description
31:16	RW	**PCI Sub-System ID** The register can support firmware code the flexibility to change PCI Sub-System ID. Customers may change the ID according to their requirements.
15:0	RW	**PCI Sub-Vendor ID** The register can support firmware code the flexibility to change PCI Sub-Vendor ID. Customers may change the ID according to their requirements.

Offset: 38h		SCU38: PCI Configuration Setting Register #3 Init = 0x03000000
Bit	**R/W**	**Description**
31:8	RW	**Class Code** The register can support firmware code the flexibility to change PCI Class Code ID of AST2050 / AST1100 . But changing the ID is usually not recommended.
7:0	RW	**PCI Revision ID** The register can support firmware code the flexibility to change PCI Revision ID of AST2050 / AST1100 . But changing the ID is usually not recommended.

Offset: 3Ch		SCU3C: System Reset Control Register Init = 0x00000001
Bit	**R/W**	**Description**
31:4		**Reserved (0)**
3	RW	**Enable external SOC reset function (GPIOB7)** The register will control the GPIOB7 to be an external reset signal pin (EXTRST#). The EXTRST# can reset SOC modules except DRAM controller. The EXTRST# is active low. System must keep EXTRST# in high level when software enables this pin. This register bit was cleared by external power reset pin, SRST#. It can be set by software after Soc boot-up.
2	RW	**External reset flag** This register bit was cleared by external power reset pin, SRST#, and set by external reset signal EXTRST#. It can be cleared by software after software checked the bit status.
1	RW	**Watch dog reset flag** This register bit was cleared by external power reset pin, SRST#, and set by internal watchdog reset signal. It can be cleared by software after software checked the bit status.
0	RW	**Power on reset flag** This register bit was set by external power reset pin, SRST#. It can be cleared by software after software checked the bit status.

Offset: 40h–44h			SCU40 ~ SCU44: SOC Scratch Register Init = 0
Offset	**Bit**	**Attr.**	**Description**
40h	31:0	RW	**SOC scratch register bit[31:0]**
44h	31:0	RW	**SOC scratch register bit[63:32]**

Note :
SOC scratch registers are designed for ARM CPU to pass the necessary information to host CPU, especially for the needs of graphics display drivers. All these registers can be read back by host CPU by way of PCI read cycles. The meaning of each bit is defined by software. This register implements the first 32 bits.

Offset: 40h		SCU40: ASPEED Defined for VGA Function Handshake Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Scratch for ASPEED SDK and SLT** 0x5A: Embedded Linux boot to Linux Properly others: Not defined
23:16	RW	**Reserved**

to next page

from previous page

15:14	RW	**MAC#1 PHY Mode Configuration** 00: Dedicate PHY 01: NCSI Mode 10: Intel NCSI EVB Mode 11: Reserved
13:12	RW	**MAC#2 PHY Mode Configuration** 00: Dedicate PHY 01: NCSI Mode 10: Intel NCSI EVB Mode 11: Reserved
11:8	RW	**Reserved**
7	RW	**DRAM Initial Selection (see note 1)** 0:VBIOS Initial the DRAM 1: SOC Firmware Initial the DRAM
6	RW	**SOC Firmware Initial DRAM Status (see note 1)** 0: DRAM Initial is not ready 1: DRAM Initial is Ready
5	RW	**Reserved (AST2000 use only)**
4	RW	**KVM Virtual EDID Function Selection (see note 2)** 0: disable 1: enable
3	RW	**Reserved (AST2000 use only)**
2	RW	**Reserved (AST2000 use only)**
1:0	RW	**Reserved**

Note :
1.
if (0x1e6e2040 D[7] == 0)
VBIOS initial the DRAM
Else
SOC Firmware initial the DRAM
SOC Firmware set 0x1e6e2040 D[6] to 1 if DRAM initial is ready
VBIOS POST will wait until 0x1e6e2040 D[6] set by SOC Firmware
End if

2.
if (0x1e6e2040 D[4] == 0)
VBIOS get EDID from DDC
Else
If the Monitor Attached
Get EDID from DDC
Else
Use Virtual EDID as EDID
End if
End if

Offset: 44h	SCU44: ASPEED Defined for VGA Function Handshake	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**The last service IRQ number**

Offset: 50h–6Ch			SCU50 ~ SCU6C:VGA Scratch Register Init = 0
Offset	**Bit**	**Attr.**	**Description**
50h	31:0	R	**VGA scratch register bit[31:0]**
54h	31:0	R	**VGA scratch register bit[63:32]**
58h	31:0	R	**VGA scratch register bit[95:64]**
5Ch	31:0	R	**VGA scratch register bit[127:96]**
60h	31:0	R	**VGA scratch register bit[159:128]**
64h	31:0	R	**VGA scratch register bit[191:160]**
68h	31:0	R	**VGA scratch register bit[223:192]**
6Ch	31:0	R	**VGA scratch register bit[255:224]**

Note :
VGA scratch registers are designed for Host CPU to pass the necessary information to ARM CPU, especially for the needs of embedded firmware. All these registers can be read back by ARM CPU. The meaning of each bit is defined by software.

Offset: 70h		SCU70: Hardware Trapping Register Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Software defined trapping registers**
23	RW	**Enable LPC dedicated reset pin function** 0 : LPC reset is shared with PCI reset pin 1 : LPC reset is located at pin number B10
22	RW	**Enable test mode** 0 : Enable normal mode 1 : Enable test mode
21	RW	**Reverse PCI AD[31:0] pin sequence** 0 : Normal pin sequence 1 : Reversed pin sequence This option is designed to optimize the trace routing when PCB layout. AST2050 / AST1100 can be applied to either on-board or add-on applications. But the optimal pin sequence for the two kinds of application may not the same. So when this bit is set, the following pin shuffle will be applied immediately. AD[31] pin replaces AD[0] pin AD[30] pin replaces AD[1] pin ..... AD[0] pin replace AD[31] pin CBE[3] pin replace CBE[0] pin CBE[2] pin replace CBE[1] pin CBE[1] pin replace CBE[2] pin CBE[0] pin replace CBE[3] pin
20	RW	**Disable ARM CPU to M-bus bridge** 0 : No operation 1 : Disable ARM CPU to M-Bus bridge When this bit is set, ARM CPU can only access memory data through AHB bus. The direct path to M-bus will be disabled. This is for insurance policy only.
19	RW	**Bypass all PLL** 0 : No operation 1 : Bypass all PLL (for test mode only)
18	RW	**Reserved, must keep at ’0’**

to next page

from previous page

17	RW	**PCI VGA Config Space Prefetch bit setting** 0: PCI VGA Config prefetch bit return 0 1: PCI VGA Config prefetch bit return 1 This bit setting controls the PCI Config Space Prefetch bit return value.
16	RW	**SOC Boot Up Full Speed Mode** 0 : ARM CPU will boot up at low speed mode (1/16 of full speed) 1 : ARM CPU will boot up at full speed mode The said low speed mode is implemented by CPU throttling logic setting at 1/16 throttling ratio. And MPLL will be turned off and MCLK is source from 24MHz reference clock. **When boot up at low speed mode, software must set this bit to 1 for full speed operation, else it will always operates at low speed mode.**
15	RW	**PCI Class Code selection** 0 : Select the Class Code for video device 1 : Select the Class Code for VGA device
14	RW	**Bypass VGA DAC** 0 : Normal DAC function 1 : Bypass DAC Mode(for test mode only)
13:12	RW	**CPU/AHB clock frequency ratio selection** 00 : Select CPU:AHB = 1:1 01 : Select CPU:AHB = 2:1 10 : Select CPU:AHB = 4:1 11 : Select CPU:AHB = 3:1
11:9	RW	**H-PLL default clock frequency selection** 00x: Reserved 010: Select 200 MHz 011: Select 166 MHz 100: Select 133 MHz 101: Select 100 MHz 110: Reserved 111: Select 24 MHz (by enabling H-PLL bypass mode)
8:6	RW	**MAC interface mode selection** 000: Reserved 001: Reserved 010: Reserved 011: Select MII(MAC#1) only 100: Select RMII(MAC#1) only 101: Reserved 110: Select RMII(MAC#1) and RMII(MAC#2) 111: Disable MAC
5	RW	**Enable VGA BIOS ROM** 0: No VGA BISO ROM (for on-board applications) 1: Enable VGA BIOS ROM (for add-on applications)
4	RW	**Reserved, must keep at '0'**
3:2	RW	**VGA memory size selection** 00 : Select 8 MB VGA memory 01 : Select 16 MB VGA memory 10 : Select 32 MB VGA memory 11 : Select 64 MB VGA memory VGA memory will share with SOC memory from SDRAM Controller.

to next page

from previous page

1:0	RW	**ARM CPU boot code selection** 0x : Reserved 10 : Boot from SPI flash memory 11 : Disable ARM CPU operation

Offset: 74h		SCU74: Multi-function Pin Control #1 Init = 0x40048000
Bit	**R/W**	**Description**
31	RW	**Enable HCLK output**
30	RW	**Enable VGA external DAC sense pin**
29	RW	**Reserved, must keep at value ”0”**
28	RW	**Reserved, must keep at value ”0”**
27	RW	**Enable GPIOE group pins shared with MAC pins** Valid for SCU70[8:6] = 2, 4, 7
26	RW	**Reserved, must keep at value ”0”**
25	RW	**Enable MAC PHY #1 PHYLINK and PHYPD# pins**
24	RW	**Enable full UART2 pins**
23	RW	**Enable VP[17:12] input/output**
22	RW	**Enable VP[11:0] input/output**
21	RW	**Reserved, must keep at value ”0”**
20	RW	**Enable MAC #2 MDC/MDIO pins**
19	RW	**Reserved, must keep at value ”0”**
18	RW	**Enable primary DDC pins**
17	RW	**Reserved, must keep at value ”0”**
16	RW	**Enable Video port A output control pins**
15	RW	**Enable VGA pins**
14	RW	**Enable I2C #7 pins**
13	RW	**Enable I2C #6 pins**
12	RW	**Enable I2C #5 pins**
11	RW	**Enable PWM 4**
10	RW	**Enable PWM 3**
9	RW	**Enable PWM 2**
8	RW	**Enable PWM 1**
7	RW	**Enable PECI pins**
6	RW	**Enable PCI host controller REQ4#/GNT4# pins**
5	RW	**Enable PCI host controller REQ3#/GNT3# pins**
4	RW	**Enable PCI host controller REQ2#/GNT2# pins**
3	RW	**Enable PCI host controller REQ1#/GNT1# pins**
2	RW	**Enable flash FLBUSY# and FLWP# pins**
1	RW	**Enable NOR flash ACK Control pin**
0	RW	**Enable NOR flash ROMA24 pin**

to next page

from previous page

Note :
The pin multiplexing function of AST2050 / AST1100 is total determined by this register. All the multiplexed pins are with 2 to 4 (at most) pin assignments. The default value of this register will assign most of the multiplex pins as GPIO pins (output buffers are in tri-state mode). Therefore, weak pull-up or pull-down resistors are required if initial high level or low level is necessary.
For more detailed multi-function pins control, reference Section 7.

Offset: 78h	SCU78: Multi-function Pin Control #2	Init = 0
Bit	**R/W**	**Description**
31:5		**Reserved (0)**
4	RW	**Disable PCI INTA# output function**
3	RW	**Enable Watchdog reset event output**
2	RW	**Enable Video port A RGB666 18 bits output mode**
1	RW	**Reserved, must keep at value "0"**
0	RW	**Enable Video port A single edge input/output mode**

Offset: 7Ch	SCU7C: Silicon Revision ID Register	Init = 0x00000202
Bit	**R/W**	**Description**
31:10		**Reserved (0)**
9:8	R	**Chip bounding option** The read back value of this register will reflect the status of the chip bonding option which is designed for product differentiation.
7:0	R	**Silicon revision ID** 0: Represent A0 silicon 1: Represent A1 silicon 2: Represent A2/A3 silicon .... And so forth.

The following table shows a list of silicon revision ID.

AST1100-A0	0x00000200
AST1100-A1	0x00000201
AST1100-A2	0x00000202
AST1100-A3	0x00000202
AST2050-A0	0x00000200
AST2050-A1	0x00000201
AST2050-A2	0x00000202
AST2050-A3	0x00000202
AST2100-A0	0x00000300
AST2100-A1	0x00000301
AST2100-A2	0x00000302
AST2100-A3	0x00000302

19 Hash & Crypto Engine (HACE)

19.1 Overview

Hash and Crypto Engine (HACE) is designed to accelerate the throughput of **hash data digest**, **encryption**, and **decryption**. Basically, HACE can be divided into two independently engines — Hash Engine and Crypto Engine. Each of which can work independently. The two engines can also be programmed to work at cascaded mode, either hash first or crypto first. Working at cascaded mode can significantly reduce memory bandwidth requirement. HACE can directly fetch data through memory bus. Therefore, HACE will not result in AHB bus congestions.

HACE only implements 11 sets of 32-bit registers to program the various supported functions. The physical address of these registers can be derived as the following:

Base address of HACE = 0x1E6E_3000
Physical address = (Base address of HACE) + Offset

HACE00: Base Address of Crypto Data Source Register
HACE04: Base Address of Crypto Data Destination Register
HACE08: Base Address of Crypto Context Buffer Register
HACE0C: Crypto Data Length Register
HACE10: Crypto Engine Command Register
HACE1C: Engine Status Register
HACE20: Base Address of Hash Data Source Register
HACE24: Base Address of Hash Digest Write Buffer Register
HACE28: Base Address of HMAC Key Buffer Register
HACE2C: Hash Data Length Register
HACE30: Hash Engine Command Register

19.2 Features

- Directly connected to APB bus
- Register programming through APB bus interface
- Supports Advanced Encryption Standard (AES) with options:
  - Electronic Code Book (ECB)
  - Cipher Block Chaining (CBC)
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
  - Counter (CTR)
  - Support Three Different Key Sizes : 128, 192 or 256 bits
- Support RC4 Encryption Standard
- Support AES/RC4 key expansion by software and pre-loading into DRAM memory
- Support multiple message digest standards: MD5/SHA1/SHA224/SHA256, HMAC-MD5/HMAC-SHA1/HMAC-SHA224/HMAC-SHA256
- Hash function support length up to 256 MByte.
- Support 4 types of engine trigger modes:
  - Encryption/decryption only
  - Message digest only

  - Encryption/decryption first, message digest second
  - Message digest first, encryption/decryption second
- Crypto and hash engine support independent and cascaded mode.
- Engine fired by directly writing command into command register
- Support CPU Interrupt option
- Programmable AES/RC4 encryption or decryption mode with programmable context buffer location for fast context switching.
- Internal Key context memory.
- Programmable key context management.
- Programmable address of source buffer & destination buffer
- Programmable address of expanded key buffer
- Direct DRAM memory access for:
  - Expanded key loading
  - Hash input data read-in
  - Hash digest write-back
  - Plaintext/Ciphertext read-in
  - Ciphertext/Plaintext write-back
- Performance Target (200MHz memory clock)
  - RC4 : throughput up to 355 Mbps.
  - AES-128 : throughput up to 225 Mbps.
  - AES-192 : throughput up to 190 Mbps.
  - AES-256 : throughput up to 165 Mbps.
  - MD5/SHA-1/SHA-224/SHA-256 : throughput up to 320 Mbps.
  - HMAC-MD5/HMAC-SHA-1/HMAC-SHA-224/HMAC-SHA-256 : throughput up to 320 Mbps.

19.3 Registers : Base Address = 0x1E6E:3000

Offset: 00h — HACE00: Base Address of Crypto Data Source Register — Init = X

Bit	R/W	Description
31:28		**Reserved (0)**
27:3	RW	**Base address of crypto data source** The base address of crypto data source must be 8-byte aligned.
2 :0		**Reserved (0)**

Offset: 04h — HACE04: Base Address of Crypto Data Destination Register — Init = X

Bit	R/W	Description
31:28		**Reserved (0)**
27:3	RW	**Base address of crypto data destination** The base address of crypto data destination must be 8-byte aligned.
2 :0		**Reserved (0)**

Offset: 08h		HACE08: Base Address of Crypto Context Buffer Register Init = X
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:3	RW	**Base address of crypto context buffer** The base address of crypto context buffer must be 8-byte aligned.
2 :0		**Reserved (0)**

Offset: 0Ch		HACE0C: Crypto Data Length Register Init = X
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:0	RW	**Crypto data length (bytes)** 0: **Invalid** 1: 1 byte 2: 2 bytes ... The register determines the data length to be encrypted or decrypted. The data length for a **RC4** crypto command is **byte aligned**. The data length for a **AES** crypto command is **16-byte aligned**. The maximum data length is up to **(256MB-1) bytes** for a **RC4/AES** crypto command. The minimum data length can be **1 byte** for a **RC4** crypto command. The minimum data length can be **16 bytes** for a **AES** crypto command.
Note : When crypto engine works in cascaded mode, HACE0C [27:0] MUST equal to HACE2C [27:0].		

Offset: 10h		HACE10: Crypto Engine Command Register Init = X
Bit	**R/W**	**Description**
31:13		**Reserved (0)**
12	RW	**Enable crypto interrupt** 0: Disable crypto interrupt 1: Enable crypto interrupt when crypto command finished
11	RW	**Disable crypto engine read-in & write-out data control** 0: Enable crypto engine read-in & write-out datat 1: Disable crypto engine read-in & write-out data
10	RW	**Disable loading context data from context buffer** 0: Enable loading context data from context buffer before running crypto algorithm 1: Disable loading context data from context buffer before running crypto algorithm
9	RW	**Disable saving context data into context buffer** 0: Enable saving context data into context buffer when finished crypto command 1: Disable saving context data into context buffer when finished crypto command
8	RW	**Crypto algorithm selection** 0: Select AES crypto algorithm 1: Select RC4 crypto algorithm
7	RW	**Crypto mode selection** 0: Decryption mode (ciphertext in, plaintext out) 1: Encryption mode (plaintext in, ciphertext out)

to next page

from previous page

6 :4	RW	**AES operation mode selection** 000: ECB mode (Initial Vector is **NOT** required) 001: CBC mode (Initial Vector is required) 010: CFB mode (Initial Vector is required) 011: OFB mode (Initial Vector is required) 100: CTR mode (Initial Vector is required) 101: **Invalid** 110: **Invalid** 111: **Invalid** Initial Vector is presented in the context buffer. This register in only applied to AES crypto algorithm. When RC4 crypto algorithm is selected, this register will be ignored.
3 :2	RW	**Key length of AES crypto algorithm** 00: 128-bit key length 01: 192-bit key length 10: 256-bit key length 11: **Invalid** This register is only applied to AES crypto algorithm. When RC4 crypto algorithm is selected, this register will be ignored.
1 :0	RW	**Crypto engine operation mode control** 00: Crypto engine works in independent mode 01: Crypto engine works in independent mode 10: Crypto engine works in cascaded mode (Crypto first, hash second) 11: Crypto engine works in cascaded mode (Hash first, crypto second) In cascaded mode, the programming of HACE10 [1:0] MUST be consistent with the programming of HACE30 [1:0]. Otherwise, HACE may be trapped in a dead lock.

Offset: 1Ch		**HACE1C: Engine Status Register** **Init = 0**
Bit	**R/W**	**Description**
31:13		**Reserved (0)**
12	RW	**Crypto interrupt flag** 0: No interrupt 1: Interrupt is pending When crypto interrupt is enabled, this bit will be set to "1" when crypto command has been finished. Writing "1" to this bit will clear this register.
11:10		**Reserved (0)**
9	RW	**Hash interrupt flag** 0: No interrupt 1: Interrupt is pending When hash interrupt is enabled, this bit will be set to "1" when hash command has been finished. Writing "1" to this bit will clear this register.
8 :2		**Reserved (0)**

to next page

from previous page

1	R	**Crypto engine status flag** 0: Crypto engine is idle 1: Crypto engine is busy
0	R	**Hash engine status flag** 0: Hash engine is idle 1: Hash engine is busy

Offset: 20h	**HACE20: Base Address of Hash Data Source Register**	**Init = X**
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:3	RW	**Base address of hash data source** The base address of hash source data must be 8-byte aligned.
2 :0		**Reserved (0)**

Offset: 24h	**HACE24: Base Address of Hash Digest Write Buffer Register**	**Init = X**
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:3	RW	**Base address of hash digest write buffer** The base address of hash digest write buffer must be 8-byte aligned. <table><tr><td>Algorithm</td><td>Digest</td><td>Digest write buffer</td></tr><tr><td>MD5</td><td>16 bytes</td><td>16 bytes</td></tr><tr><td>SHA-1</td><td>20 bytes</td><td>20 bytes</td></tr><tr><td>SHA-224</td><td>28 bytes</td><td>32 bytes</td></tr><tr><td>SHA-256</td><td>32 bytes</td><td>32 bytes</td></tr></table>
2 :0		**Reserved (0)**

Offset: 28h	**HACE28: Base Address of HMAC Key Buffer Register**	**Init = X**
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:6	RW	**Base address of HMAC key buffer** The base address of HMAC key buffer must be 64-byte aligned.
5 :0		**Reserved (0)**
Note : HMAC Key Buffer store the result of calculate HMAC key command (HACE30 [8] = 1). See "Hash Function Programming Sequence" for detail information.		

Offset: 2Ch	**HACE2C: Hash Data Length Register**	**Init = X**
Bit	**R/W**	**Description**
31:28		**Reserved (0)**

to next page

from previous page

27:0	RW	**Hash data length** 0: 0 byte 1: 1 byte 2: 2 bytes ... The register determines the data length to be hashed. When HACE30 [8:7] = 2, the data length for a hash command is **64 byte aligned**. When HACE30 [8:7] != 2, the data length for a hash command is **byte aligned**. The maximum data length is up to **(256MB-1) bytes** for a hash command. When HACE30 [8:7] = 2, the minimum data length can be **64 byte** for a hash command. When HACE30 [8:7] != 2, the minimum data length can be **0 byte** for a hash command.
Note : When hash engine works in cascaded mode, HACE2C [27:0] MUST equal to HACE0C [27:0].		

Offset: 30h		**HACE30: Hash Engine Command Register** **Init = X**
Bit	**R/W**	**Description**
31:10		**Reserved (0)**
9	RW	**Enable hash interrupt** 0: Disable hash interrupt 1: Enable hash interrupt when hash command finished
8 :7	RW	**HMAC engine command mode** 00: Calculate digest without HMAC 01: Calculate digest with HMAC 10: Calculate digest with Accumulative Mode 11: Calculate HMAC key (Hash engine must be programmed to be working at the independent mode)
6 :4	RW	**Hash algorithm selection** 000: Select MD5 algorithm 001: **Invalid** 010: Select SHA-1 algorithm 011: **Invalid** 100: Select SHA-224 algorithm 101: Select SHA-256 algorithm 110: **Invalid** 111: **Invalid**
3 :2	RW	**Byte swapping control** 00: **Invalid** 01: Byte swapping control for all MD5 hash commands (little-endian) 10: Byte swapping control for all SHA-1/SHA-224/SHA-256 hash commands (big-endian) 11: **Invalid**

to next page

from previous page

1 :0	RW	**Hash engine operation mode control** 00: Hash engine works in independent mode 01: Hash engine works in independent mode 10: Hash engine works in cascaded mode (Crypto first, hash second) 11: Hash engine works in cascaded mode (Hash first, crypto second) In cascaded mode, the programming of HACE10 [1:0] MUST be consistent with the programming of HACE30 [1:0]. Otherwise, HACE may be trapped in a dead lock.

19.4 Crypto Context Buffer Format

19.4.1 RC4 (272 Bytes)

Byte Range	Description
000 - 007	Reserved (0)
008	Index I (With initial value 1)
009	Index J (With initial value 0)
00A - 00F	Reserved (0)
010 - 10F	RC4 Key Byte 0 ~ 255

19.4.2 AES-128 (192 Bytes)

Byte Range	Description
000 - 00F	Initial Vector Byte 0 ~ 15 (Initial Vector is NOT required in **ECB** mode)
010 - 0BF	AES SW Expanded Key Byte 0 ~ 175

19.4.3 AES-192 (224 Bytes)

Byte Range	Description
000 - 00F	Initial Vector Byte 0 ~ 15 (Initial Vector is NOT required in **ECB** mode)
010 - 0DF	AES SW Expanded Key Byte 0 ~ 207

19.4.4 AES-256 (256 Bytes)

Byte Range	Description
000 - 00F	Initial Vector Byte 0 ~ 15 (Initial Vector is NOT required in **ECB** mode)
010 - 0FF	AES SW Expanded Key Byte 0 ~ 239

19.5 Hash Function Programming Sequence

19.5.1 Parameter Definition

- $Hash\_Input\_Data\_Base\_Adr$ (8-byte aligned): Base address of data buffer which want to calculate hash digest.

- $Hash\_Digest\_Base\_Adr$ (8-byte aligned): Base address of hash digest write buffer.
  - MD5 : Digest is 16 bytes, digest write buffer is 16 bytes
  - SHA1 : Digest is 20 bytes, digest write buffer is 20 bytes
  - SHA224 : Digest is 28 bytes, digest write buffer is 32 bytes
  - SHA256 : Digest is 32 bytes, digest write buffer is 32 bytes
- $Hash\_Acc\_Digest\_Base\_Adr$ (64-byte aligned): Base address of accumulative hash digest write buffer.
  - MD5 : Accumulative digest is 16 bytes, accumulative digest write buffer is 16 bytes
  - SHA1 : Accumulative digest is 20 bytes, accumulative digest write buffer is 20 bytes
  - SHA224 : Accumulative digest is 28 bytes, accumulative digest write buffer is 32 bytes
  - SHA256 : Accumulative digest is 32 bytes, accumulative digest write buffer is 32 bytes

- $Hash\_Input\_Size$ (byte aligned): Byte size of data buffer which want to calculate hash digest.
- $Hash\_Acc\_Input\_Size$ (64-byte aligned): Byte size of accumulative data buffer which want to calculate accumulative hash digest.

- $K_0\_Buffer\_Base\_Adr$ (8-byte aligned): Base address of 64 byte K_0 buffer.
- $HMAC\_Key\_Buffer\_Base\_Adr$ (64-byte aligned): Base address of 64 byte buffer which is used to store the result of calculate HMAC key command (HACE30 [8:7] = 3).

19.5.2 MD5/SHA1/SHA224/SHA256

- $Hash\_Input\_Data\_Base\_Adr$ (8-byte aligned)
- $Hash\_Digest\_Base\_Adr$ (8-byte aligned)
- $Hash\_Input\_Size$ (byte aligned)

1. Calculating Hash Digest:
   (a) HACE20 = $Hash\_Input\_Data\_Base\_Adr$ (8-byte aligned)
   (b) HACE24 = $Hash\_Digest\_Base\_Adr$ (8-byte aligned)
   (c) HACE2C = $Hash\_Input\_Size$ (byte aligned)
   (d) HACE30:
      - MD5 : 04h or 204h
      - SHA1 : 28h or 228h
      - SHA224 : 48h or 248h
      - SHA256 : 58h or 258h

19.5.3 HMAC MD5/SHA1/SHA224/SHA256

Programming sequence "Preparing K_0 Buffer" & "Calculating HMAC Key Buffer" are needed ONLY when secret key are changed.

- $K_0\_Buffer\_Base\_Adr$ (8-byte aligned)
- $HMAC\_Key\_Buffer\_Base\_Adr$ **(64-byte aligned)**
- $Hash\_Input\_Data\_Base\_Adr$ (8-byte aligned)
- $Hash\_Digest\_Base\_Adr$ (8-byte aligned)
- $Hash\_Input\_Size$ (byte aligned)

1. Preparing K_0 Buffer:
   Software need to prepare 64 byte K_0 buffer as described in APPENDIX A of "FIPS PUB 198: The Keyed-Hash Message Authentication Code (HMAC)"
2. Calculating HMAC Key Buffer:
   (a) HACE20 = $K_0\_Buffer\_Base\_Adr$ (8-byte aligned)
   (b) HACE28 = $HMAC\_Key\_Buffer\_Base\_Adr$ **(64-byte aligned)**
   (c) HACE2C = 40h
   (d) HACE30:
      - HMAC-MD5 : 184h or 384h
      - HMAC-SHA1 : 1A8h or 3A8h
      - HMAC-SHA224 : 1C8h or 3C8h
      - HMAC-SHA256 : 1D8h or 3D8h
3. Calculating HMAC Hash Digest:
   (a) HACE20 = $Hash\_Input\_Data\_Base\_Adr$ (8-byte aligned)
   (b) HACE24 = $Hash\_Digest\_Base\_Adr$ (8-byte aligned)
   (c) HACE28 = $HMAC\_Key\_Buffer\_Base\_Adr$ **(64-byte aligned)**
   (d) HACE2C = $Hash\_Input\_Size$ (byte aligned)
   (e) HACE30:
      - HMAC-MD5 : 84h or 284h
      - HMAC-SHA1 : A8h or 2A8h
      - HMAC-SHA224 : C8h or 2C8h
      - HMAC-SHA256 : D8h or 2D8h

19.5.4 Accumulative Mode

- $HMAC\_Key\_Buffer\_Base\_Adr$ **(64-byte aligned)**
- $Hash\_Input\_Data\_Base\_Adr$ (8-byte aligned)
- $Hash\_Acc\_Digest\_Base\_Adr$ **(64-byte aligned)**
- $Hash\_Acc\_Input\_Size$ **(64-byte aligned)**

1. Allocating & Initiating Accumulative Digest Write Buffer:
   This sequence is neeed ONLY before processing first accumulative data.

- MD5/SHA1/SHA224/SHA256:

Byte Range	MD5	SHA-1	SHA-224	SHA-256
00 - 03	67452301h	01234567h	D89E05C1h	67E6096Ah
04 - 07	EFCDAB89h	89ABCDEFh	07D57C36h	85AE67BBh
08 - 0B	98BADCFEh	FEDCBA98h	17DD7030h	72F36E3Ch
0C - 0F	10325476h	76543210h	39590EF7h	3AF54FA5h
10 - 13	Reserved	F0E1D2C3h	310BC0FFh	7F520E51h
14 - 17	Reserved	Reserved	11155868h	8C68059Bh
18 - 1B	Reserved	Reserved	A78FF964h	ABD9831Fh
1C - 1F	Reserved	Reserved	A44FFABEh	19CDE05Bh

- Phase 1 of HMAC MD5/SHA1/SHA224/SHA256:
  Phase 1 equal to step 5 ~ 6 in Table 1 of "FIPS PUB 198: The Keyed-Hash Message Authentication Code (HMAC)".

  HMAC_Key_Buffer store the outcome of programming sequence "Calculating HMAC Key Buffer". The HMAC_Key_Buffer must be ready before running this sequence.

Byte Range	HMAC MD5/SHA1/SHA224/SHA256
00 - 03	HMAC_Key_Buffer[00:03] (MD5/SHA1/SHA224/SHA256)
04 - 07	HMAC_Key_Buffer[04:07] (MD5/SHA1/SHA224/SHA256)
08 - 0B	HMAC_Key_Buffer[08:0B] (MD5/SHA1/SHA224/SHA256)
0C - 0F	HMAC_Key_Buffer[0C:0F] (MD5/SHA1/SHA224/SHA256)
10 - 13	HMAC_Key_Buffer[10:13] (SHA1/SHA224/SHA256)
14 - 17	HMAC_Key_Buffer[14:17] (SHA224/SHA256)
18 - 1B	HMAC_Key_Buffer[18:1B] (SHA224/SHA256)
1C - 1F	HMAC_Key_Buffer[1C:1F] (SHA224/SHA256)

- Phase 2 of HMAC MD5/SHA1/SHA224/SHA256:
  Phase 2 equal to step 8 ~ 9 in Table 1 of "FIPS PUB 198: The Keyed-Hash Message Authentication Code (HMAC)"

  HMAC_Key_Buffer store the outcome of programming sequence "Calculating HMAC Key Buffer". The HMAC_Key_Buffer must be ready before running this sequence.

Byte Range	HMAC MD5/SHA1/SHA224/SHA256
00 - 03	HMAC_Key_Buffer[20:23] (MD5/SHA1/SHA224/SHA256)
04 - 07	HMAC_Key_Buffer[24:27] (MD5/SHA1/SHA224/SHA256)
08 - 0B	HMAC_Key_Buffer[28:2B] (MD5/SHA1/SHA224/SHA256)
0C - 0F	HMAC_Key_Buffer[2C:2F] (MD5/SHA1/SHA224/SHA256)
10 - 13	HMAC_Key_Buffer[30:33] (SHA1/SHA224/SHA256)
14 - 17	HMAC_Key_Buffer[34:37] (SHA224/SHA256)
18 - 1B	HMAC_Key_Buffer[38:3B] (SHA224/SHA256)
1C - 1F	HMAC_Key_Buffer[3C:3F] (SHA224/SHA256)

2. Calculating Accumulative Hash Digest:
   Running this sequence repeatedly until receiving the last accumulative data.

   (a) When receiving the last accumulative data, software need to add Padding Message at the end of the accumulative data. Padding Message is described in the specific of MD5 and SHA1/SHA224/SHA256.

   Let N be the totally byte size of accumulative data, the 64 bit length-column of Padding Message is:

Hash Algorithm	Non-HMAC	HMAC Phase 1	HMAC Phase 2
MD5	N * 8	(64 + N) * 8	(64 + 16) * 8
SHA1	N * 8	(64 + N) * 8	(64 + 20) * 8
SHA224	N * 8	(64 + N) * 8	(64 + 28) * 8
SHA256	N * 8	(64 + N) * 8	(64 + 32) * 8

   (b) HACE20 = $Hash\_Input\_Data\_Base\_Adr$ (8-byte aligned)
   (c) HACE24 = $Hash\_Acc\_Digest\_Base\_Adr$ **(64-byte aligned)**
   (d) HACE28 = $Hash\_Acc\_Digest\_Base\_Adr$ **(64-byte aligned)**
   (e) HACE2C = $Hash\_Acc\_Input\_Size$ **(64-byte aligned)**
   (f) HACE30:
      - MD5 or HMAC-MD5 : 104h or 304h
      - SHA1 or HMAC-SHA1 : 128h or 328h
      - SHA224 or HMAC-SHA224 : 148h or 348h
      - SHA256 or HMAC-SHA256 : 158h or 358h

20 Video Engine

20.1 Overview

Video Engine supports high performance video compressions with a wide range of video quality and compression ratio options. The adopted compressing algorithm is a mixed one including JPEG and Vector Quantization (VQ). To enable video compression engine, the following memory buffers are required by allocating from DRAM memory for each of them.

- Video Source Buffer #1
- Video Source Buffer #2
- CRC Buffer (optional, for quick scene change detection)
- Block Change Detection (BCD) Flag Buffer (for scene change detection buffer)
- Compressed Video Stream Buffer

Video Engine implements many registers to program the various supported functions. The physical address of these registers can be derived as the following:

Base address of Video Engine = 0x1E70_0000
Physical address = (Base address of Video Engine) + Offset

20.2 Features

- Directly connected to AHB bus interface for register programming
- Directly access video data through M-Bus
- Maximum operation frequency: 266MHz
- Video source can be from internal VGA output or from external DVO input
- Engine clock can be from CPU clock or memory clock
- Engine clock can be turned off when engine is idle
- Support two video compression formats
  - YUV420: for lower video quality but higher compression ratio
  - YUV444: for higher video quality but lower compression ratio
- Support high resolution video compression up to 1920x1200x32bpp@60Hz
- Target frame rate: 30 frame/sec for 1280x1024@60Hz under YUV420 compression format
- Support intelligent scene change detection by comparing CRC code of each video scan line, significantly reducing memory bandwidth requirement
- Support direct video data fetch for video compression
  - Only enabled for high resolution modes (enhanced high color and true color modes)
  - Quick Cursor must be enabled (cursor overlay will be done in client site)
  - Regular VGA display refresh can be turned off when feasible
  - Significantly reducing memory requirements for video compression
  - 16 bits of DRAM data bus width is enough for 1600x1200x16bpp video compression
- Support arbitrary video down scaling with horizontal & vertical video filtering option (4x2 spatial filter)

- Integrate one RC4 encryption engine for video stream encryption
  - 1 set of loadable 256x8 SRAM for expanded key buffers
  - Key expansion is done by firmware
  - Provide enable/disable option
- Support smart video mode detection functions
- Support video mode watch dog interrupt when source video mode change
- Support programmable bit resolution truncation to input video data
- Support 12 selectable JPEG quality levels
- Support VQ compression mode
- Support video auto stream mode and single frame trigger mode

20.3 Registers : Base Address = 0x1E70:0000

VR000: Protection Key Register		
Offset: 000h		**Init = 0**
Bit	**Attr.**	**Description**
31:0	RW	**Protection key** This register is designed to protect all the registers designed for Video Engine from unpredictable updates. All the registers are still readable even locked. The password of the protection key is **0x1A03_8AA8**. **Unlock registers**: Write 0x1A03_8AA8 to this register **Lock registers**: Write other values to this register When this register is unlocked, the read back value of this register is 0x0000_0001. When this register is locked, the read back value of this register is 0x0000_0000. This register will be reset by power on reset, watch dog reset and SCU software reset. Software must wait minimum 1us to unlock the key after reset signal de-asserted.

VR004: Video Engine Sequence Control Register		
Offset: 004h		**Init = 0**
Bit	**Attr.**	**Description**
31:19	R	**Reserved (0)**
18	R	**Video compression engine status** 0: Video compression engine is in busy state 1: Video compression engine is in idle state
17	R	**Reserved (0)**
16	R	**Video capture engine status** 0: Video capture engine is in busy state 1: Video capture engine is in idle state
15:12	R	**Reserved (0)**
11:10	RW	**Video data format conversion for video compression** 00: YUV444 01: YUV420 10: Reserved 11: Reserved Video capture engine always converts the input RGB data stream to YUV444 data format and write out to the video memory, and this register can select the expected video data format that video compression engine has to convert firstly before doing video compression.
9	RW	**Reserved** This register must be always "0"
8	RW	**Reserved** This register must be always "0"

to next page

from previous page

7	RW	**Enable watchdog for detecting input video resolution mode change** 0: Disable watchdog 1: Enable watchdog The display resolution of an input video source may change anytime. Therefore, building a watchdog is necessary to dedicated monitor the change and generate an interrupt to notice S/W handler when a mode change does happen. This register works only when VR004[0] is "1" and a stable video resolution has been detected. Set mode detection trigger first and then set this bit when mode detection ready.
6	RW	**Trigger Video to insert full frame compression** Write 0: No operation. Write 1: Trigger engine to insert signle full frame compression for stream mode encode. Read 0: The insertion is completed. Read 1: The insertion is not completed.
5	RW	**Enable automatic video compression** 0: Compress a single frame for each trigger command 1: Compress multiple frames for each trigger command When this register is enabled, video capturing engine and video compression engine can work together to automatically capture and compress continuous frames without S/W intervention. Allocating double buffer is required to enable this register. Software must set this bit first before set trigger register VR004[4].
4	RW	**Enable or Trigger Video compression** 0: No operation 0→1: Trigger video compression when compress single frame mode 1: Enable video compression Setting this register from 0 to 1 will trigger video compression engine to compress the captured video data stored in the frame buffer. Video compression can compress single or multiple frames depending on the setting of the register VR004[5]. Software must insert at least one read cycle or 1us delay time between continuous trigger register setting. Software must make sure that the VR004[18] is '1' before trigger this bit.
3	RW	**Enable capturing multiple frames** 0: Capturing single frame 1: Capture multiple frames AST2050 / AST1100 can allocate double buffers for the video capture engine to continuously capture multiple frames. Capturing multiple frames can improve video performance. Allocating double buffers is required before enabling this register. Software must set this bit first before set trigger register VR004[1].
2	RW	**Force Video compression engine Idle** 0: No operation 1: Force compression engine to enter idle state This register is used by software to force compression engine to enter idle state only when capture engine is idle and compression engine hangs up.

to next page

from previous page

1	RW	**Enable or Trigger Video capture** 0: When capture is idle: no operation 0: When capture is not idle and in capture multiple frames mode (VR004[3]=1): Capture engine will stop when completely capture last video frame. 0→1: Trigger video capture when capture single frame mode Setting this register from 0 to 1 will trigger video capture engine to capture either single or multiple video frames, depending on the setting of VR004[3]. Video capture engine will stop capturing video at the end of a frame whenever this register is reset to 0. Software must insert at least one read cycle or 1us delay time between continuous trigger register setting. Software must make sure that the VR004[16] is ’1’ before trigger this bit.
0	RW	**Trigger Video mode detection hardware** 0: No operation 0→1: Trigger video mode detection 1: Enable mode dectection hardware Setting this register from 0 to 1 will trigger the video mode detection hardware to detect a video mode based on the input video source. When a stable video mode has been detected, the hardware will set the corresponding flag for status read back. And the related video parameters generated by the hardware can also be read back from the related registers. An optional interrupt is also available. Software must insert at least one read cycle or 1us delay time between continuous trigger register setting.

Note :
About auto and trigger mode, please reference following table:

VR004[3]	VR004[5]	Capture	Compression	Buffer mode
0	0	Single frame	Software trigger	Frame buffer mode
0	1	Single frame	Hardware auto trigger	Frame buffer mode
1	0	NA	NA	NA
1	1	Multiple frames	Hardware auto trigger	Stream buffer mode

VR008: Video Control Register

Offset: 008h **Init = 0**

Bit	Attr.	Description
31:24	R	**Reserved (0)**
23:16	RW	**Maximum frame rate control for video capture** When this register is reset to 0x00, capture engine will try to capture all the input frames, if memory and network bandwidth is sufficient for doing that. When this register is set to a non-zero value, video capture engine will skip some frames to reduce memory and network bandwidth. The maximum frame rate will be: Maximum frame rate = (VR008[23:16]) * (Source frame rate) / 60.
15:14	RW	**Reserved** This register must be always ”0”
13	RW	**Clock mode of digital video input port** 0: Single edge clock mode 1: Dual edge clock mode This register only impacts the external digital video input.

to next page

from previous page

12	RW	**Video input port bit number** 0: 24 bits 1: 18 bit for single edge only (VR008[13]=0)
11:10	RW	**Clock delay control for digital video input port** 00: No delay 01: Delay by around 1 ns 10: Inversed clock but no delay 11: Inversed clock and delay by around 1 ns This register can adjust the delay of the input video clock for video capture engine to precisely capture video data.
9	RW	**Reserved** This register must be always "0"
8	RW	**Disable hardware cursor overlay for internal VGA (VR008[5]=0)** 0: With VGA hardware cursor overlay image 1: Without VGA hardware cursor overlay image This register can be set by ARM CPU to inform internal VGA controller to generate video data without hardware cursor overlay image. When this register is enabled, the hardware cursor overlay has to be done in clients by Quick Cursor algorithm. The DAC output of internal VGA controller is, if necessary, with hardware cursor overlay image even this register is set to 1.
8	RW	**Auto mode for direct fetch mode for VGA frame buffer (VR008[5]=1)** 0: normal operation 1: auto mode for register VR008[3], VR008[4], VR00C and VR010 This bit is used only for direct fetch mode for VGA frame buffer when VR004[5]=1. When this register are set to '1', software dose't require to set the registers VR008[3], VR008[4], VR00C and VR010. The video hardware will automatically reference to VGA hardware setting for these registers.
7 :6	RW	**Data format for video capture** 00: CCIR601-2 compliant YUV format 01: Full range YUV format 10: RGB format 11: Invalid AST2050 / AST1100 supports two kinds of YUV formats. One is CCIR601-2 compatible; the other is a proprietary format to support higher video quality. RGB format is for debugging purpose only.
5	RW	**Fetch video data directly from VGA frame buffer (internal VGA only)** 0: From internal VGA input or external digital video input 1: Direct fetch from VGA display frame buffer This register is designed for capturing high resolution video modes under low memory bandwidth requirement. It only supports high color and true color display modes. And there is no hardware cursor overlay from the capture data. Therefore, hardware cursor overlay has to be done in clients by Quick Cursor algorithm.
4	RW	**Use the internal timing generator to generate Display Enable (DE) signal (VR008[5]=0)** 0: Use the external DE signal (DVI only) 1: Use the internal DE signal DE signal is for video capture engine to precisely sampling effective video data.
4	RW	**VGA frame buffer bpp mode (VR008[5]=1)** 0: 32 bpp mode 1: 16 bpp mode This register set the number of bits per pixel in VGA frame buffer.

to next page

from previous page

3	RW	**Attribute of the external video source (VR008[5]=0)** 0: Form a pure digital video source 1: From an external Analog-to-Digital Converter (ADC) If a digital video source is from an external video ADC, using internal timing generator will be automatically enabled no matter the register setting of VR004[4].
3	RW	**VGA frame buffer 16 bpp color mode (VR008[5]=1)** 0: RGB565 1: RGB555 This register set the 16 bpp color format in VGA frame buffer
2	RW	**Video source selection** 0: Video source is from the integrated VGA controller 1:Video source is from an external video source
1	RW	**Video source VSYNC polarity selection** 0: Internal VSYNC polarity is same as source VSYNC 1: Internal VSYNC polarity is inversed of source VSYNC Note: This register should be reset to 0 before triggering video mode detection hardware.
0	RW	**Video source HSYNC polarity selection** 0: Internal HSYNC polarity is same as source HSYNC 1: Internal HSYNC polarity is inversed of source HSYNC Note: This register should be reset to 0 before triggering video mode detection hardware.

VR00C: Video Timing Generation Setting Register (VR008[5]=0)

Offset: 00Ch **Init = X**

Bit	Attr.	Description
31:29	R	**Reserved (0)**
28:16	RW	**Number of pixels to the first active pixel** This register defines the number of pixels from the rising edge of HSYNC to the first active pixel
15:13	RW	**Reserved (0)**
12:0	RW	**Number of pixels to the last active pixel** This register defines the number of pixels from the rising edge of HSYNC to the last active pixel.

Note :
Timing generator is primarily designed for video source, like the one form ADC, going without Horizontal Display Enable (HDE) signal. Timing generator will generate one for video capture engine to precisely capture active pixels for the first active pixel to the last active pixel in a scan line.

VR00C: Video Direct Frame Buffer Mode Control Register (VR008[5]=1)

Bit	Attr.	Description
31:28	RW	**Reserved (0)**
27:3	RW	**Direct frame buffer fetch mode base address bit [27:3]** When direct frame buffer fetch mode is enabled (VR008[5]=1), this register set the base address of source video data.
2 :0	RW	**Reserved (0)**

VR010: Video Timing Generation Setting Register (VR008[5]=0)		
Offset: 010h		**Init = X**
Bit	**Attr.**	**Description**
31:28	RW	**Reserved (0)**
27:16	RW	**Number of scan lines to the first active scan line** This register defines that number scan lines from the rising edge of VSYNC to the first active scan line.
15:12	RW	**Reserved (0)**
11:0	RW	**Number of scan lines to the last active scan line** This register defines the number scan lines from the rising edge of VSYNC to the last active scan line.
Note : Timing generator is primarily designed for video source, like the one form ADC, going without Vertical Display Enable (VDE) signal. Timing generator will try to generate one for video capture engine to precisely capture scan lines from the first active scan line to the last active scan line in a frame.		
VR010: Video Direct Frame Buffer Mode Control Register (VR008[5]=1)		
31:16	RW	**Direct frame buffer fetch timing control bit[15:0]** When direct frame buffer fetch mode is enabled (VR008[5]=1), it controlled the 64-pixel segment minimum fetch time = VR010[31:15] * MCLK_cycle_time
13:3	RW	**Direct frame buffer fetch mode line offset bit [27:3]** When direct frame buffer fetch mode is enabled (VR008[5]=1), this register set the line offset of source video data.
2 :0	RW	**Reserved (0)**

VR014: Video Scaling Factor Register		
Offset: 014h		**Init = X**
Bit	**Attr.**	**Description**
31:16	RW	**Vertical down scaling factor** The setting value of this register must be equal or larger than 4096. All the active scan lines will be kept. When the setting value is getting larger, the output video window size will be getting smaller. The formula is as the fowling: (Vertical window size of output video) = (Veridical window size of input video) * 4096 / (Vertical scaling factor)
15:0	RW	**Horizontal down scaling factor** The setting value of this register must be equal or larger than 4096. All the active pixels will be kept. When the setting value is getting larger, the output video window size will be getting smaller. The formula is as the fowling: (Horizontal window size of output video) = (Horizontal window size of input video) * 4096 / (Horizontal scaling factor)

VR018: Video Scaling Filter Parameter Register #0		
Offset: 018h		**Init = X**
Bit	**Attr.**	**Description**
31:0	RW	**Scaling parameters F00, F01, F02, F03** F03: Bit[31:24] F02: Bit[23:24] F01: Bit[15:8] F00: Bit[7:0] The data format of the scaling parameters is 2'comlement format by S2.5.

Note :
The video scalar integrated in AST2050 / AST1100 can support a 2x4 scaling filter. It means that the video output of each down-scaled pixel will be generated by the weighting sum of the surrounding 8 pixels, avoiding graphics information loose. There are three alowed parameter settings.
when scaling down factor = 1.0, VR018, VR01C, VR020, VR024 = 00200000h,
when 0.5 <= scaling down factor < 1.0, VR018, VR01C, VR020, VR024 = 00101000h,
when scaling down factor < 0.5, VR018, VR01C, VR020, VR024 = 08080808h

VR01C: Video Scaling Filter Parameter Register #1		
Offset: 01Ch		**Init = X**
Bit	**Attr.**	**Description**
31:0	RW	**Scaling Parameters F10, F11, F12, F13** F13: Bit[31:24] F12: Bit[23:24] F11: Bit[15:8] F10: Bit[7:0] The data format of the scaling parameters is 2'comlement format by S2.5.

VR020: Video Scaling Filter Parameter Register #2		
Offset: 020h		**Init = X**
Bit	**Attr.**	**Description**
31:0	RW	**Scaling Parameters F20, F21, F22, F23** F23: Bit[31:24] F22: Bit[23:24] F21: Bit[15:8] F20: Bit[7:0] The data format of the scaling parameters is 2'comlement format by S2.5.

VR024: Video Scaling Filter Parameter Register #3		
Offset: 024h		**Init = X**
Bit	**Attr.**	**Description**
31:0	RW	**Scaling Parameters F30, F31, F32, F33** F33: Bit[31:24] F32: Bit[23:24] F31: Bit[15:8] F30: Bit[7:0] The data format of the scaling parameters is 2'comlement format by S2.5.

VR02C: Video BCD Control Register		
Offset: 02Ch		**Init = 0**
Bit	**Attr.**	**Description**
31:24	R	**Reserved (0)**
23:16	RW	**BCD tolerance value** This register defines the tolerance for block change detection (BCD). Any blocks with maximum changing difference in its own block more than this tolerance value will be taken as a non-changing blocks. This function is designed to reduce memory and network bandwidth, especially for the noisy video source from ADC.
15:2	R	**Reserved (0)**
1	RW	**Delay block change update by one frame** 0: Disable 1: Enable When this register is enabled, BCD can purposely delay the block change update for each changed block by one frame. The changed block will be updated in the next frame. This function is designed to reduce memory and network bandwidth.
0	RW	**Enable block change detection (BCD)** 0: Disable 1: Enable When BCD is disabled, video compression engine will compress all the input frames. When BCD is enabled, video compression engine will detect and compress changing blocks only. That will significantly reduce memory and network bandwidth. Higher video frame rate can be achieved as well.

VR030: Video Capturing Window Setting Register		
Offset: 030h		**Init = X**
Bit	**Attr.**	**Description**
31:27	RW	**Reserved (0)**
27:16	RW	**Horizontal total pixels to be captured**
15:11	RW	**Reserved (0)**
10:0	RW	**Vertical total scan lines to be captured**

Note :
The register provides the possibility of doing video capture only for a partial display window in a frame.

VR034: Video Compression Window Setting Register		
Offset: 034h		**Init = X**
Bit	**Attr.**	**Description**
31:27	RW	**Reserved (0)**
27:16	RW	**Horizontal total pixels to be compressed**
15:11	RW	**Reserved (0)**
10:0	RW	**Vertical total scan lines to be compressed**

Note :
The register provides the possibility of doing video compression only for a partial display window in a frame.

VR038: Video Compression Stream Buffer Processing Offset Register		
Offset: 038h		**Init = X**
Bit	**Attr.**	**Description**
31:22	RW	**Reserved (0)**
21:7	RW	**Video 1 compression stream buffer process offset** The video compression stream data processing address which tell the stream interrupt controller the data have been accepted by ISR, but still occupy the buffer.
6 :0	RW	**Reserved (0)**

VR03C: Video Compression Stream Buffer Read Offset Register		
Offset: 03Ch		**Init = X**
Bit	**Attr.**	**Description**
31:22	R	**Reserved (0)**
21:7	RW	**Video 1 compression stream buffer read offset** The video compression stream data read pointer which will tell the stream buffer controller the current software read pointer.
6 :0	R	**Reserved (0)**

VR040: Video Base Address of CRC Buffer Register		
Offset: 040h		**Init = X**
Bit	**Attr.**	**Description**
31:28	R	**Reserved (0)**
27:3	RW	**CRC buffer base address Bit [27:3]** Cyclic Redundancy Code (CRC) comparison is an option to reduce the memory bandwidth requirement for block change detection (BCD). When enabling this option, a CRC buffer allocated from SDRAM memory is required to store the CRC code for each video segment which is a small scan line with 64 pixels. This register determines the base address of the CRC buffer. The address bit [2:0] should always be 0.
2 :0	R	**Reserved (0)**

VR044: Video Based Address of Video Source Buffer #1 Register		
Offset: 044h		**Init = X**
Bit	**Attr.**	**Description**
31:28	R	**Reserved (0)**
27:8	RW	**Base address of video source buffer #1 Bit [27:8]** In order to support the double-buffer mode for video capturing, two video source buffers, allocated from SDRAM memory, are required. This register determines the base address of the first video source buffer. The address bit [7:0] should be 0.
7 :0	RW	**Reserved (0)**

VR048: Video Scan Line Offset of Video Source Buffer Register		
Offset: 048h		**Init = X**
Bit	**Attr.**	**Description**
31:14	R	**Reserved (0)**

to next page

from previous page

27:3	RW	**Scan line offset of video source buffer Bit [13:8]** This register determines the scan line offset (memory address distance) of video source buffer #1 and buffer #2 from one scan line to the next scan line. The address bit [2:0] should be 0.
2 :0	R	**Reserved (0)**

Note :
This is offset address from line to line. The address bit [2:0] should be 0. The buffer offset 0 can be calculated from horizontal total pixel number * 4bpp. The horizontal pixel number must be multiplier of 8. If the real pixel number is not multiplier of 8, please select a least number which is multiplier of 8 and greater than the pixel number.

VR04C: Video Base Address of Video Source Buffer #2 Register

Offset: 04Ch **Init = X**

Bit	Attr.	Description
31:14	R	**Reserved (0)**
27:8	RW	**Base address of video source buffer #2 Bit [27:8]** In order to support the double-buffer mode for video capturing, two video source buffers, allocated from SDRAM memory, are required. This register determines the base address of the second video source buffer. The address bit [7:0] should be 0.
7 :0	RW	**Reserved (0)**

VR050: Video Base Address of BCD Flag Buffer Register

Offset: 050h **Init = X**

Bit	Attr.	Description
31:28	R	**Reserved (0)**
27:3	RW	**Base address of BCD flag buffer Bit [27:3]** BCD flag buffer, allocated from SDRAM memory, records the BCD flag for each block. It requires 4 bits per block to store the necessary status information. BCD flag buffer needs to be initialized for the first time. This register determines the base address of BCD flag buffer. The address bit [2:0] should be 0.
2 :0	R	**Reserved (0)**

VR054: Video Base Address of Compressed Video Stream Buffer Register

Offset: 054h **Init = X**

Bit	Attr.	Description
31:28	R	**Reserved (0)**
27:7	RW	**Base address of compressed video stream buffer Bit [27:7]** The register determines the base address of the video buffer for storing compressed video stream. The address bit [6:0] should be 0.
6 :0	RW	**Reserved (0)**

VR058: Video Stream Buffer Size Register

Offset: 058h **Init = X**

Bit	Attr.	Description
31:5	R	**Reserved (0)**

to next page

from previous page

4 :3	RW	**Stream buffer packet number** 00: 4 packets 01: 8 packets 10: 16 packets 11: 32 packets
2 :0	RW	**Stream buffer packet size** 000: 1 KB 001: 2 KB 010: 4 KB 011: 8 KB 100: 16 KB 101: 32 KB 110: 64 KB 111: 128 KB

VR05C: Video Compression Stream Buffer Write Offset Read Back

Offset: 05Ch **Init = X**

Bit	Attr.	Description
31:22	R	**Reserved (0)**
21:7	R	**Video 1 compression stream buffer write offset** his read back register can return the offset address of current video write offset address for compressed data in the stream buffer.
6 :0	R	**Reserved (0)**

VR060: Video Compression Control Register

Offset: 060h **Init = 0**

Bit	Attr.	Description
31:22	RW	**Reserved 0**
21:20	RW	**JPEG Huffman encoding table selection** 00: Select Y and UV tables 01: Select Y table only 1x: Select UV table only In most of the cases, ”00” is recommended.
19	RW	**Reserved** This register must be always ”1”.
18:17	RW	**JPEG engine hardware test control** For testing hardware only. Writing 0 for a normal condition.
16	RW	**Reserved** This register must be always ”0”.

to next page

from previous page

15:11	RW	**DCT luminance quantization table selection** This register determines how JPEG engine executes DCT quantization for the luminance DCT coefficients of each block. There are 12 DCT luminance quantization tables and 12 DCT chrominance quantization tables implemented in JEPG engine. The first bit of this register (VR060[15]) determines whether the 12 DCT luminance quantization tables or the 12 DCT chrominance quantization tables will be referenced. 0: Select one of the 12 DCT luminance quantization tables (Table #0 ∼ Table #11) 1: Select one of the 12 DCT chrominance quantization tables (Table #0 ∼ Table #11) The left 4 bits of this register (VR060[14:11]) will determine which one of the selected 12 tables will be used for quantizing the luminance DCT coefficients. 0000: Table #0 0001: Table #1 ... 1011: Table #11 Others: Invalid These tables can be applied to both YUV420 and YUV444 video compression.
10:6	RW	**DCT chrominance quantization table selection** This register determines how JPEG engine executes DCT quantization for the chrominance DCT coefficients of each block. There are 12 DCT luminance quantization tables and 12 DCT chrominance quantization tables implemented in JEPG engine. The first bit of this register (VR060[10]) determines whether the 12 DCT luminance quantization tables or the 12 DCT chrominance quantization tables will be referenced. 0: Select one of the 12 DCT luminance quantization tables (Table #0 ∼ Table #11) 1: Select one of the 12 DCT chrominance quantization tables (Table #0 ∼ Table #11) The left 4 bits of this register (VR060[9:6]) will determine which one of the selected 12 tables will be used for quantizing the chrominance DCT coefficients. 0000: Table #0 0001: Table #1 ... 1011: Table #11 Others: Invalid These tables can be applied to both YUV420 and YUV444 video compression.
5	RW	**Enable RC4 encryption** 0: Disable 1: Enable This register will determine the compressed video stream will be RC4-encrypted or not. When enabled, video engine will encrypt the compressed video stream before writing into compressed video stream buffer.
4	RW	**Reserved** This register must be always ”0”.
3	RW	**Reserved** This register must be always ”0”.
2	RW	**Reserved** This register must be always ”0”.
1	RW	**Enable 4-color VQ encoding** 0: Enable 2-color VQ encoding 1: Enable 4-color VQ encoding

to next page

from previous page

0	RW	**JPEG only encoding** 0: Enable JPEG/VQ mixed mode video encoding 1: Enable JPEG only video encoding mode Video engine provides two video encoding modes. One is a pure JPEG video encoding. The other is a mixed video encoding mode, which can automatically encode each video block by either JEPG or VQ, depending on the number of colors detected from the video block. Comparing to JPEG compression algorithm, VQ compression algorithm can provide a much higher compression ratio and without any color loose.

VR064: reserved		
Offset: 064h		**Init = X**
Bit	**Attr.**	**Description**

VR06C: reserved		
Offset: 06Ch		**Init = X**
Bit	**Attr.**	**Description**

VR070: Video Total Size of Compressed Video Stream Read Back Register		
Offset: 070h		**Init = X**
Bit	**Attr.**	**Description**
31:20	R	**Reserved (0)**
19:0	R	**Total size of compressed video stream** This register reports the total length of compressed video stream already stored in the compressed video stream buffer for a video frame. The unit is one double word.

VR074: Video Total Number of Compressed Video Blocks Read Back Register		
Offset: 074h		**Init = X**
Bit	**Attr.**	**Description**
31:30	R	**Reserved (0)**
29:16	R	**Compressed block counter read back (number of blocks)** This register reports the number of video blocks having been compressed into the video compressed stream buffer for a video frame.
15:14	R	**Reserved (0)**
13:0	R	**Processed total block counter read back (number of blocks)** This register reports the total number of video blocks having been processed by video engine
Note : This register is applicable only at YUV420 mode.		

VR078: Video Frame-End Offset of Compressed Video Stream Buffer Read Back Register		
Offset: 078h		**Init = X**
Bit	**Attr.**	**Description**
31:22	R	**Reserved (0)**

to next page

from previous page

21:3	R	**Frame-end offset of compressed video stream buffer Bit [21:3]** This register reports the frame-end offset of the compressed video stream buffer. Adding the value of this register with the base address of the compressed video stream buffer (VR054) can derive the last address of the last video stream data for the last compressed frame. The bit [2:0] should be 0.
2 :0	R	**Reserved (0)**

VR07C: Video Compressed Frame Counter Read Back Register

Offset: 07Ch **Init = X**

Bit	Attr.	Description
31:0	R	**Compressed frame counter Bit [31:0]** This register reports the value of the frame counter which is designed to count the number of compressed frames up to the read back moment.

VR090: Video Source Left/Right Edge Detection Read Back Register

Offset: 090h **Init = X**

Bit	Attr.	Description
31:28	R	**Reserved (0)**
27:16	R	**Video source right edge location from the rising edge of HSYNC Bit [11:0]** The unit of this register is one pixel.
15	R	**No display clock detected** 0: No display clock detected 1: Display clock detected
14	R	**No active display detected** 0: No active display detected 1: Active display detected
13	R	**No HSYNC detected** 0: No HSYNC detected 1: HSYNC detected
12	R	**No VSYNC detected** 0: No VSYNC detected 1: VSYNC detected
11:0	R	**Video source left edge location from the rising edge of HSYNC Bit [11:0]** The unit of this register is one pixel.

Note :
This register is primarily designed to detect the video timing of an external video source. If the video source is from the internal VGA controller, then the video timing can directly be from the information, recorded in VGA scratch registers, provided by VGA BIOS.

VR094: Video Source Top/Bottom Edge Detection Read Back Register

Offset: 094h **Init = X**

Bit	Attr.	Description
31:29	R	**Reserved (0)**
27:16	R	**Video source bottom edge location from the rising edge of VSYNC Bit [11:0]** The unit of this register is one scan line.
15:12	R	**Reserved (0)**

to next page

from previous page

11:0	R	**Video source top edge location from the rising edge of VSYNC Bit [11:0]** The unit of this register is one scan line.
Note : This register is primarily designed to detect the video timing of an external video source. If the video source is from the internal VGA controller, then the video timing can directly be from the information, recorded in VGA scratch registers, provided by VGA BIOS.		

VR098: Video Mode Detection Status Read Back Register		
Offset: 098h		**Init = X**
Bit	**Attr.**	**Description**
31	R	**Mode detection HSYNC ready** 0: HSYNC is not yet ready 1: HSYNC is ready HSYNC being ready just means that the signal has been detected, but not necessary stable.
30	R	**Mode detection VSYNC ready** 0: VSYNC is not yet ready 1: VSYNC is ready VSYNC being ready just means that the signal has been detected, but not necessary stable.
29	R	**Mode detection HSYNC polarity** 0: Source HSYNC polarity is positive 1: Source HSYNC polarity is negative
28	R	**Mode detection VSYNC polarity** 0: Source VSYNC polarity is positive 1: Source VSYNC polarity is negative
27:16	R	**Mode detection vertical scan lines Bit [11:0]** This register reports the number of scan lines detected between two continuous VSYNC. When there is no VSYNC signal detected, all the bits of this register will be ”1”.
15	R	**Video source is out of synchronization** 0: Video source is still stable 1: Video source is out of synchronization This status register, designed to report any mode changes, is effective only when mode detection watchdog is enabled (VR004[7] = 1). Whenever video source is out of synchronization, S/W needs to trig mode detection again.
14	R	**Mode detection vertical signal stable** 0: Vertical signal detection is not stable 1: Vertical signal detection is stable
13	R	**Mode detection horizontal signal stable** 0: Horizontal signal detection is not stable 1: Horizontal signal detection is stable
12	R	**Auto detection of external digital video source type** 0: Video source is from DVI receiver 1: Video source is from ADC output The major difference is the video source from DVI receiver goes with Display Enable signal, the video source from ADC output goes without Display Enable signal.
11:0	R	**Mode detection horizontal period Bit [11:0]** This register reports the period of the detected HSYNC signal after horizontal mode detection is stable (VR098[13] = 1). If there is no HSYNC signal detected, all bits of this register will be 1. The measurement clock is 24MHz.

VR300: Video Control Register

Offset: 300h		**Init = 0**
Bit	**Attr.**	**Description**
31:16	R	**Reserved (0)**
15	RW	**RC4 non-auto reset mode** This register recommend to be always ”1”
14	RW	**RC4 save mode** This register recommend to be always ”1”
13:10	RW	**reserved** This register must be always ”0”
9	RW	**RC4 test mode** This register must be always ”0”
8	RW	**RC4 initial reset** This register can be set only when engine is idle. Set to ’1’ can reset RC4 states, then set to ’0’ to go back normal state.
7 :6	RW	**Reserved** This register must be always ”0”
5 :4	RW	**Enable video vertical down scaling line buffer** 00: Disable down scaling line buffer 01: Enable down scaling line buffer for video 10: invalid 11: invalid To support video scaling filter, vertical down scaling line buffer must be enabled. Otherwise, line dropping algorithm will be applied for instead.
3	RW	**Reserved (0)**
2	RW	**Delay Internal VSYNC** 0: no delay 1: delay internal VSYNC by 12 periods of HSYNC cycle time This is used for video capture auto mode and anti-flicter enabled to avoid frame dropped.
1	RW	**Video stream buffer controller save mode** 0: for internal test only 1: recommended
0	RW	**Reserved** This register must be always ”0”

VR304: Video Interrupt Control Register

Offset: 304h		**Init = 0**
Bit	**Attr.**	**Description**
31:7	RW	**Reserved (0)**
7	RW	**Reserved (0)**
6	RW	**Reserved (0)**
5	RW	**Enable Video frame complete interrupt** 0: Disable 1: Enable
4	RW	**Enable Video mode detection ready interrupt** 0: Disable 1: Enable

to next page

from previous page

3	RW	**Enable Video compression complete interrupt** 0: Disable 1: Enable
2	RW	**Enable Video compression packet ready interrupt** 0: Disable 1: Enable
1	RW	**Enable Video frame capture complete interrupt** 0: Disable 1: Enable
0	RW	**Enable Video mode detection watchdog out of lock interrupt** 0: Disable 1: Enable

VR308: Video Interrupt Control Register		
Offset: 308h		**Init = X**
Bit	**Attr.**	**Description**
31:8	RW	**Reserved (X)**
7	RW	**Reserved (0)**
6	RW	**Reserved (0)**
5	RW	**Video frame complete interrupt status** 0: No interrupt 1: Interrupt is pending Clear this register by writing 1.
4	RW	**Video mode detection ready interrupt status** 0: No interrupt 1: Interrupt is pending Clear this register by writing 1.
3	RW	**Video compression complete interrupt status** 0: No interrupt 1: Interrupt is pending. Clear this register by writing 1.
2	RW	**Video compression packet ready interrupt status** 0: No interrupt 1: Interrupt is pending Clear this register by writing 1.
1	RW	**Video capture complete interrupt status** 0: No interrupt 1: Interrupt is pending Clear this register by writing 1.
0	RW	**Video Mode detection watchdog out of lock interrupt status** 0: No interrupt 1: Interrupt is pending Clear this register by writing 1.

VR30C: Mode Detection Parameter Register		
Offset: 30Ch		**Init = X**
Bit	**Attr.**	**Description**
31:28	RW	**Mode detection horizontal stable detection tolerance Bit [3:0]** This register defines the tolerance in detecting horizontal signal stable. The sampling clock to detect horizontal signal is 24MHz. The unit of this register is one clock period of 24MHz clock source.
27:24	RW	**Mode detection vertical stable detection tolerance Bit [3:0]** This register defines the tolerance in detecting vertical signal stable. The unit of this register is one period of a scan line.
23:20	RW	**Mode detection horizontal stable minimum Bit [3:0]** This register defines the required minimum count of detecting stable HSYNC signal to set mode detection horizontal signal stable. The minimum acceptable value of this register is 3.
19:16	RW	**Mode detection vertical stable minimum Bit [3:0]** This register defines the required minimum count of detecting stable VSYNC signal to set mode detection vertical signal stable. The minimum acceptable value of this register is 3.
15:8	RW	**Mode detection edge pixel threshold [7:0]** This register defines the minimum RGB value of effective pixels in detecting left or right edge. Due to video signal sources from ADC are very easy to be coupled with analog noise, the setting of this register can help the mode detector to screen out noise.
7 :0	RW	**Reserved (0)**

VR310: Video Memory Restriction Area Starting Address Register		
Offset: 310h		**Init = 0x0000_0000**
Bit	**Attr.**	**Description**
31:28	R	**reserved(0)**
27:16	RW	**Video Memory Restriction Area Starting Address** The address must be 64 KB alignment.
15:0	R	**reserved(0)**
Note : Any video memory write access whose address is outside the restriction area will be discarded.		

VR314: Video Memory Restriction Area End Address Register		
Offset: 314h		**Init = 0x0FFF_0000**
Bit	**Attr.**	**Description**
31:28	R	**reserved(0)**
27:16	RW	**Video Memory Restriction Area End Address** The address must be 64 KB alignment.
15:0	R	**reserved(0)**
Note : Any video memory write access whose address is outside the restriction area will be discarded.		

VR320: Primary CRC Parameter Register

Offset: 320h **Init = X**

Bit	Attr.	Description
31:16	RW	**Primary CRC upper 16-bit polynomial** This register defines the upper 16-bit coefficients of primary CRC polynomial used for scene change detection. Primary CRC polynomial is designed to support the video capture for video source buffer #1.
15:8	RW	**Primary CRC lower 8-bit polynomial** This register defines the lower 8-bit coefficients of primary CRC polynomial used for scene change detection. Primary CRC polynomial is designed to support the video capture for video source buffer #1.
7 :2	RW	**Maximum frame skip count for CRC comparison scheme** This register, effective only when CRC comparison scheme (VR320[0]) is selected, defines the maximum number of still frames that can be skipped. Over this maximum frame count, all the video blocks in the next frame will be captured and updated in the corresponding video buffer. This function is designed to workaround the potential risk of aliasing error when adopting CRC comparison scheme.
1	RW	**Reserved** This register must be always "0"
0	RW	**Scene change detection scheme selection** 0: Select pixel-by-pixel comparison scheme 1: Select CRC comparison scheme Scene change detection on CRC comparison basis will consumes lower memory bandwidth requirements than scene change detection on pixel-by-pixel comparison basis, but at the risk of aliasing error. VR320[7:2] provides a workaround solution to fix the potential error. Selecting CRC comparison scheme is recommended.

Note :
The adopted primary CRC check is based on a 24-bit polynomial.

VR324: Secondary CRC Parameter Register

Offset: 324h **Init = X**

Bit	Attr.	Description
31:16	RW	**Secondary CRC upper 16-bit polynomial** This register defines the upper 16-bit coefficients of secondary CRC polynomial used for scene change detection. Secondary CRC polynomial is designed to support the video capture for video source buffer #2.
15:8	RW	**Secondary CRC lower 8-bit polynomial** This register defines the upper 8-bit coefficients of secondary CRC polynomial used for scene change detection. Secondary CRC polynomial is designed to support the video capture for video source buffer #2.
7 :0	R	**Reserved (0)**

Note :
The adopted primary CRC check is based on a 24-bit polynomial.

VR328: Video Data Truncation Register

Offset: 328h **Init = X**

Bit	Attr.	Description
31:16	R	**Reserved (0)**

to next page

from previous page

8 :6	RW	**R channel reduction bit number** 000: no reduction 001: recude 1 bit 010: recude 2 bits 011: recude 3 bits 100: recude 4 bits 101: recude 5 bits 110: recude 6 bits 111: recude 7 bits
5 :3	RW	**G channel reduction bit number** 000: no reduction 001: recude 1 bit 010: recude 2 bits 011: recude 3 bits 100: recude 4 bits 101: recude 5 bits 110: recude 6 bits 111: recude 7 bits
2 :0	RW	**B channel reduction bit number** 000: no reduction 001: recude 1 bit 010: recude 2 bits 011: recude 3 bits 100: recude 4 bits 101: recude 5 bits 110: recude 6 bits 111: recude 7 bits

VR340: VGA Scratch Remap Read Back Register

Offset: 340h **Init = X**

Bit	Attr.	Description
31:30	R	**Reserved(0)**
29:24	R	**VGA hardware cursor X position offset bit[5:0]**
23:22	R	**Reserved(0)**
21:16	R	**VGA hardware cursor Y position offset bit[5:0]**
15:10	R	**Reserved(0)**
9	R	**VGA hardware cursor type** 0: Monochrome cursor type. 1: Color cursor type.
8	R	**VGA hardware cursor is enabled** 0: VGA hardware cursor is disabled. 1: VGA hardware cursor is enabled.
7 :0	R	**Remap to VGA CR80 register**

VR344: VGA Scratch Remap Read Back Register

Offset: 344h **Init = X**

Bit	Attr.	Description
31:27	R	**Reserved(0)**
26:16	R	**Hardware cursor Y position bit[10:0]**

to next page

from previous page

15:12	R	**Reserved(0)**
11:0	R	**Hardware cursor X position bit[11:0]**

VR348: VGA Scratch Remap Read Back Register		
Offset: 348h		**Init = X**
Bit	**Attr.**	**Description**
31:26	R	**Reserved(0)**
25:3	R	**Hardware cursor pattern memory address bit [25:3]**
2 :0	R	**Reserved(0)**

VR34C: VGA Scratch Remap Read Back Register		
Offset: 34Ch		**Init = X**
Bit	**Attr.**	**Description**
31:24	R	**Remap to VGA CR8F register**
23:16	R	**Remap to VGA CR8E register**
15:8	R	**Remap to VGA CR8D register**
7 :0	R	**Remap to VGA CR8C register**

VR350: VGA Scratch Remap Read Back Register		
Offset: 350h		**Init = X**
Bit	**Attr.**	**Description**
31:24	R	**Remap to VGA CR93 register**
23:16	R	**Remap to VGA CR92 register**
15:8	R	**Remap to VGA CR91 register**
7 :0	R	**Remap to VGA CR90 register**

VR354: VGA Scratch Remap Read Back Register		
Offset: 354h		**Init = X**
Bit	**Attr.**	**Description**
31:24	R	**Remap to VGA CR97 register**
23:16	R	**Remap to VGA CR96 register**
15:8	R	**Remap to VGA CR95 register**
7 :0	R	**Remap to VGA CR94 register**

VR358: VGA Scratch Remap Read Back Register		
Offset: 358h		**Init = X**
Bit	**Attr.**	**Description**
31:24	R	**Remap to VGA CR9B register**
23:16	R	**Remap to VGA CR9A register**
15:8	R	**Remap to VGA CR99 register**
7 :0	R	**Remap to VGA CR98 register**

VR35C: VGA Scratch Remap Read Back Register		
Offset: 35Ch		**Init = X**
Bit	**Attr.**	**Description**
31:30	R	**VGA power state**
29	R	**VGA attribute index register bit 5**
28	R	**VGA mask register not zero**
27	R	**VGA CRT reset**
26	R	**VGA screen off**
25	R	**VGA reset**
24	R	**VGA enable**
23:16	R	**Remap to VGA CR9E register**
15:8	R	**Remap to VGA CR9D register**
7 :0	R	**Remap to VGA CR9C register**

VR400~VR4FC: RC4 Encryption Key Register #0 ~ #63		
Offset: 400~4FCh		**Init = X**
Bit	**Attr.**	**Description**
31:0	RW	**RC4 key data SRAM** There are total 256 bytes of embedded SRAM (64 double words in total) designed to store RC4 encryption keys. Initializing the SRAM is necessary when enabling RC4 encryption for compressed video stream.

21 AHB to P-Bus Bridge

21.1 Overview

AHB-to-P Bus Bridge (A2P) is an interface controller bridging two internal buses:

AHB: The internal system bus supporting ARM SOC subsystem
P-Bus: The internal expansion bus supporting bus commands from PCI slave controller

A2P is a one way bus bridge providing a path for ARM to access all the IP modules located on the P-Bus.

21.2 Operation

The bridge will be auto enabled when set to PCI master mode (SCU70.bit[4])
AHB to P-bus bridge control registers address = 0x1E72_0000+OFFSET

OFFSET = 00000-0007F Address for relocated I/O on P-bus
OFFSET = 00080-0FFFF reserved
OFFSET = 10000-1FFFF Address for MMIO space on P-bus

22 MDMA Engine

22.1 Overview

MDMA Controller (MDMA) provides hardware logic to speed up the throughput of memory data copy or data filling. At least 4X ∼ 8X speed up factor can be expected, comparing to the throughput by way of a series of CPU read/write cycles. Additionally, MDMA supports 16 double words of MDMA command queue to reduce CPU waiting time. A bunch of MDMA commands can be fired continuously without waiting the idle state of MDMA controller. Interrupt option is available as well.

MDMA only implements 6 sets of 32-bit registers to program the various supported functions. The physical address of these registers can be derived as the following:

Base address of MDMA = 0x1E74_0000
Physical address = (Base address of MDMA) + Offset

MDMA00: Base Address of Source Data Register
MDMA04: Base Address of Destination Data Register
MDMA08: Buffer Filling Data Register
MDMA0C: MDMA Command Register
MDMA10: Interrupt Control Register
MDMA14: Interrupt Status Register

22.2 Features

- Directly connected to AHB bus
- Support direct data transfer through internal memory bus
- 16 stages command queue buffer
- Accelerate memory-to-memory data movement throughput by at least 4 times
- Support fast memory data fill operation (for ECC memory initialization)

22.3 Registers : Base Address = 0x1E74:0000

Registers with offset 00h, 04h, 08h, 0Ch will write to command queue buffer.
Registers with offset 10h, 14h will directly write to register.

Offset: 00h		MDMA00: Base Address of Source Data Register Init = X
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:0	RW	**Base address of source data [27:0]** MDMA can support DMA function up to 256MB with byte-aligned precision.

Note :
When clock ratio MCLK/H-PLL > 2 or H-PLL/MCLK > 2, writing data to this register is legal untill Status of MDMA IDLE (MDMA14 Bit [3]) is ”1”.

Before writing data to this register, please make sure that the available MDMA command queue length (MDMA14 Bit [8:4]) is not ”0”.

Otherwise, MDMA command overflow will happen.

Offset: 04h	MDMA04: Base Address of Destination Data Register	Init = X
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:0	RW	**Base address of destination data [27:0]** MDMA can support DMA function up to 256MB with byte-aligned precision.

Note :
When clock ratio MCLK/H-PLL > 2 or H-PLL/MCLK > 2, writing data to this register is legal untill Status of MDMA IDLE (MDMA14 Bit [3]) is "1".

Before writing data to this register, please make sure that the available MDMA command queue length (MDMA14 Bit [8:4]) is not "0".

Otherwise, MDMA command overflow will happen.

Offset: 08h	MDMA08: Buffer Filling Data Register	Init = X
Bit	**R/W**	**Description**
31:0	RW	**Buffer filling data** This double word register specifies the data to be filled into a buffer with a specified address range. Buffer filling only supports double-word aligned address ranges.

Note :
When clock ratio MCLK/H-PLL > 2 or H-PLL/MCLK > 2, writing data to this register is legal untill Status of MDMA IDLE (MDMA14 Bit [3]) is "1".

Before writing data to this register, please make sure that the available MDMA command queue length (MDMA14 Bit [8:4]) is not "0".

Otherwise, MDMA command overflow will happen.

Offset: 0Ch	MDMA0C: MDMA Command Register	Init = 0xxx_xxxxh
Bit	**R/W**	**Description**
31	RW	**Update status of MDMA command ID** 0: Disable 1: Enable When writing "1" to this bit, MDMA will set the status of MDMA command ID, specified by Bit [30:28], to "1" when finishing this MDMA command.
30:28	RW	**MDMA command ID number (#0∼#7)** This ID number is assigned by S/W. It will be used as a reference to generate the interrupt corresponding to the ID number.
27:26		**Reserved (0)**
25:24	RW	**MDMA command type** 00: MDMA command 01: Reserved 10: Buffer filling command 11: Reserved Buffer filling command can be used to initialize ECC memory. When this type of command is fired, MDMA will only issue memory write requests, no memory read request.

to next page

from previous page

23:0	RW	**Data length (byte) of MDMA** 0: Invalid 1: 1 byte 2: 2 bytes ... MDMA can support up to (16M-1) bytes of data movement or filling for one set of MDMA command.
Note : Any write cycles for this register (MDMA0C) will imply one MDMA command has been fired into MDMA command queue. MDMA Controller will start to execute the specified MDMA function. When clock ratio MCLK/H-PLL > 2 or H-PLL/MCLK > 2, writing data to this register is legal untill Status of MDMA IDLE (MDMA14 Bit [3]) is "1". Before writing data to this register, please make sure that the available MDMA command queue length (MDMA14 Bit [8:4]) is not "0". Otherwise, MDMA command overflow will happen.		

Offset: 10h		**MDMA10: Interrupt Control Register** **Init = 0**
Bit	**R/W**	**Description**
31:24		**Reserved (0)**
23	RW	**Interrupt mask of MDMA command ID #7** 0: Disable interrupt 1: Enable interrupt This bit will determine whether to generate interrupt or not when finished a MDMA command with ID #7.
22	RW	**Interrupt mask of MDMA command ID #6** 0: Disable interrupt 1: Enable interrupt This bit will determine whether to generate interrupt or not when finished a MDMA command with ID #6.
21	RW	**Interrupt mask of MDMA command ID #5** 0: Disable interrupt 1: Enable interrupt This bit will determine whether to generate interrupt or not when finished a MDMA command with ID #5.
20	RW	**Interrupt mask of MDMA command ID #4** 0: Disable interrupt 1: Enable interrupt This bit will determine whether to generate interrupt or not when finished a MDMA command ID #4.
19	RW	**Interrupt mask of MDMA command ID #3** 0: Disable interrupt 1: Enable interrupt This bit will determine whether to generate interrupt or not when finished a MDMA command with ID #3.

to next page

from previous page

18	RW	**Interrupt mask of MDMA command ID #2** 0: Disable interrupt 1: Enable interrupt This bit will determine whether to generate interrupt or not when finished a MDMA command with ID #2.
17	RW	**Interrupt mask of MDMA command ID #1** 0: Disable interrupt 1: Enable interrupt This bit will determine whether to generate interrupt or not when finished a MDMA command with ID #1.
16	RW	**Interrupt mask of MDMA command ID #0** 0: Disable interrupt 1: Enable interrupt This bit will determine whether to generate interrupt or not when finished a MDMA command with ID #0.
15:4		**Reserved (0)**
3	RW	**Enable interrupt when MDMA controller is IDLE** 0: Disable interrupt 1: Enable interrupt when MDMA controller is IDLE
2		**Reserved (0)**
1	RW	**Interrupt mask of MDMA command overflow ("1" is Recommended setting)** 0: Disable interrupt 1: Enable interrupt when MDMA command queue is overflow
0		**Reserved (0)**

Offset: 14h — MDMA14: Interrupt Status Register — Init = 0000_0100h

Bit	R/W	Description
31:24		**Reserved (0)**
23	RW	**Status of MDMA command ID #7** 0: Not yet finished 1: MDMA command ID #7 has been finished Writing "1" to this bit will clear this register.
22	RW	**Status of MDMA command ID #6** 0: Not yet finished 1: MDMA command ID #7 has been finished Writing "1" to this bit will clear this register.
21	RW	**Status of MDMA command ID #5** 0: Not yet finished 1: MDMA command ID #7 has been finished Writing "1" to this bit will clear this register.
20	RW	**Status of MDMA command ID #4** 0: Not yet finished 1: MDMA command ID #7 has been finished Writing "1" to this bit will clear this register.

to next page

from previous page

19	RW	**Status of MDMA command ID #3** 0: Not yet finished 1: MDMA command ID #7 has been finished Writing "1" to this bit will clear this register.
18	RW	**Status of MDMA command ID #2** 0: Not yet finished 1: MDMA command ID #7 has been finished Writing "1" to this bit will clear this register.
17	RW	**Status of MDMA command ID #1** 0: Not yet finished 1: MDMA command ID #7 has been finished Writing "1" to this bit will clear this register.
16	RW	**Status of MDMA command ID #0** 0: Not yet finished 1: MDMA command ID #7 has been finished Writing "1" to this bit will clear this register.
15:9		**Reserved (0)**
8 :4	R	**Available MDMA command queue length** 00000: MDMA command queue is full (no free space) 00001: MDMA command queue has 1 double word of free space 00010: MDMA command queue has 2 double words of free space ... 10000: MDMA command queue has 16 double words of free space Others: Reserved MDMA is equipped with 16 double words of FIFO to implement MDMA command queue. Therefore, the maximum available queue length is 16.
3	RW	**Status of MDMA IDLE** 0: Not yet idle or MDMA command queue is not empty 1: MDMA is idle and MDMA command queue is empty Writing "1" to this bit will clear this register.
2		**Reserved (0)**
1	RW	**MDMA command queue is overflow** 0: Not overflow 1: MDMA command queue is overflow MDMA command queue overflow means ARM CPU continuously writes out new MDMA commands (MDMA00, MDMA04, MDMA08 or MDMA0C) even the available MDMA command queue length (MDMA14 [8:4]) is 0. Writing "1" to this bit will clear this status flag.
0	R	**MDMA Controller status** 0: MDMA Controller is idle and MDMA command queue is empty 1: MDMA Controller is busy or MDMA command queue is not empty

23 GPIO Controller

23.1 Overview

AST2050 / AST1100 Integrates one set of GPIO Controller with maximum 64 control pins to provide general-purpose input/output functions. All the I/O buffers are 3.3V with 5V tolerance capability, and all the GPIO pins can be categorized into 7 groups. Please reference Section 3.5.

This is a superset of registers definition. For AST2050/AST1100 chip, only partial GPIO bits are supported.

Each GPIO pin can be programmed to support the following options:

- Input or output option (input mode or output mode)
- Interrupt generation option (enabled or disabled interrupt generation)
- Interrupt sensitivity option (level-high, level-low, rising-edge, falling-edge or both-edge trigger mode)
- WDT reset tolerance (for non-interrupted related registers only)
- De-bouncing option (0ms, 1ms, 5ms or 10ms de-bouncing)

GPIO implements 16 sets of 32-bit registers, which are listed below, to program the various supported functions including input/output mode, interrupt sensitivity, WDT tolerance, and de-bouncing options. Each register has its own specific offset value, ranging from 0x00 to 0x3Ch, to derive its physical address location.

Base Address of GPIO = 0x1E78_0000
Register Address of GPIO = (Base Address of GPIO) + Offset

GPIO00: GPIO Data Value Register
GPIO04: GPIO Direction Register
GPIO08: GPIO Interrupt Enable Register
GPIO0C: GPIO Interrupt Sensitivity Type 0 Register
GPIO10: GPIO Interrupt Sensitivity Type 1 Register
GPIO14: GPIO Interrupt Sensitivity Type 2 Register
GPIO18: GPIO Interrupt Status Register
GPIO1C: GPIO Reset Tolerant Register
GPIO20: Extended GPIO Data Value Register
GPIO24: Extended GPIO Direction Register
GPIO28: Extended GPIO Interrupt Enable Register
GPIO2C: Extended GPIO Interrupt Sensitivity Type 0 Register
GPIO30: Extended GPIO Interrupt Sensitivity Type 1 Register
GPIO34: Extended GPIO Interrupt Sensitivity Type 2 Register
GPIO38: Extended GPIO Interrupt Status Register
GPIO3C: Extended GPIO Reset Tolerant Register
GPIO40: GPIO Debounce Setting #1
GPIO44: GPIO Debounce Setting #2
GPIO48: Extended GPIO Debounce Setting #1
GPIO4C: Extended GPIO Debounce Setting #2
GPIO50: Debounce Time Setting #1
GPIO54: Debounce Time Setting #2
GPIO58: Debounce Time Setting #3

23.2 Features

- Directly connected to APB bus
- Support 8 dedicated and 56 shared GPIO pins
- Programmable reset tolerance option for each GIPO pin
- Support interrupt triggered by all the 64 GPIO pins
- Each input pin is with 0ms/1us/1ms/5ms/10ms de-bouncing logic option
- Default internal pull-down resistors for each GPIO pins
- 8 out of the 64 GPIO pins are with 16mA driving current, others are with 8mA.
- Need external pull-up resistors

23.3 Registers : Base Address = 0x1E78:0000

Offset: 00h		GPIO00: GPIO Data Value Register — Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOD[7:0] data register**
23:16	RW	**Port GPIOC[7:0] data register**
15:8	RW	**Port GPIOB[7:0] data register**
7 :0	RW	**Port GPIOA[7:0] data register**

Offset: 04h		GPIO04: GPIO Direction Register — Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOD[7:0] direction control** 0: Select input mode 1: Select output mode
23:16	RW	**Port GPIOC[7:0] direction control** 0: Select input mode 1: Select output mode
15:8	RW	**Port GPIOB[7:0] direction control** 0: Select input mode 1: Select output mode
7 :0	RW	**Port GPIOA[7:0] direction control** 0: Select input mode 1: Select output mode

Offset: 08h		GPIO08: GPIO Interrupt Enable Register — Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOD[7:0] interrupt enable** 0: Disable interrupt 1: Enable interrupt
23:16	RW	**Port GPIOC[7:0] interrupt enable** 0: Disable interrupt 1: Enable interrupt
15:8	RW	**Port GPIOB[7:0] interrupt enable** 0: Disable interrupt 1: Enable interrupt

to next page

from previous page

7 :0	RW	**Port GPIOA[7:0] interrupt enable** 0: Disable interrupt 1: Enable interrupt

Offset: 0Ch	GPIO0C: GPIO Interrupt Sensitivity Type 0 Register	Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOD[7:0] interrupt sensitivity type 0 selection** 0: Select falling-edge or level-low trigger mode 1: Select rising-edge or level-high trigger mode
23:16	RW	**Port GPIOC[7:0] interrupt sensitivity type 0 selection** 0: Select falling-edge or level-low trigger mode 1: Select rising-edge or level-high trigger mode
15:8	RW	**Port GPIOB[7:0] interrupt sensitivity type 0 selection** 0: Select falling-edge or level-low trigger mode 1: Select rising-edge or level-high trigger mode
7 :0	RW	**Port GPIOA[7:0] interrupt sensitivity type 0 selection** 0: Select falling-edge or level-low trigger mode 1: Select rising-edge or level-high trigger mode

Offset: 10h	GPIO10: GPIO Interrupt Sensitivity Type 1 Register	Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOD[7:0] interrupt sensitivity type 1 selection** 0: Select edge trigger mode 1: Select level trigger mode
23:16	RW	**Port GPIOC[7:0] interrupt sensitivity type 1 selection** 0: Select edge trigger mode 1: Select level trigger mode
15:8	RW	**Port GPIOB[7:0] interrupt sensitivity type 1 selection** 0: Select edge trigger mode 1: Select level trigger mode
7 :0	RW	**Port GPIOA[7:0] interrupt sensitivity type 1 selection** 0: Select edge trigger mode 1: Select level trigger mode

Offset: 14h	GPIO14: GPIO Interrupt Sensitivity Type 2 Register	Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOD[7:0] interrupt sensitivity type 2 selection** 0: Select edge or level trigger mode 1: Select dual-edge trigger mode
23:16	RW	**Port GPIOC[7:0] interrupt sensitivity type 2 selection** 0: Select edge or level trigger mode 1: Select dual-edge trigger mode
15:8	RW	**Port GPIOB[7:0] interrupt sensitivity type 2 selection** 0: Select edge or level trigger mode 1: Select dual-edge trigger mode
7 :0	RW	**Port GPIOA[7:0] interrupt sensitivity type 2 selection** 0: Select edge or level trigger mode 1: Select dual-edge trigger mode

Offset: 18h		GPIO18: GPIO Interrupt Status Register Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOD[7:0] interrupt status register** Read 0: No interrupt pending Read 1: interrupt pending Write 0: No operation Write 1: Clear interrupt status flag
23:16	RW	**Port GPIOC[7:0] interrupt status register** Read 0: No interrupt pending Read 1: interrupt pending Write 0: No operation Write 1: Clear interrupt status flag
15:8	RW	**Port GPIOB[7:0] interrupt status register** Read 0: No interrupt pending Read 1: interrupt pending Write 0: No operation Write 1: Clear interrupt status flag
7 :0	RW	**Port GPIOA[7:0] interrupt status register** Read 0: No interrupt pending Read 1: interrupt pending Write 0: No operation Write 1: Clear interrupt status flag

Offset: 1Ch		GPIO1C: GPIO Reset Tolerant Register Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOD[7:0] WDT reset tolerance enable** 0: GPIO00 and GPIO04 registers will be reset by WDT reset 1: GPIO00 and GPIO04 registers will not be reset by WDT reset Each GPIO pin can be individually programmed to be WDT reset tolerant or not.
23:16	RW	**Port GPIOC[7:0] WDT reset tolerance enable** 0: GPIO00 and GPIO04 registers will be reset by WDT reset 1: GPIO00 and GPIO04 registers will not be reset by WDT reset Each GPIO pin can be individually programmed to be WDT reset tolerant or not.
15:8	RW	**Port GPIOB[7:0] WDT reset tolerance enable** 0: GPIO00 and GPIO04 registers will be reset by WDT reset 1: GPIO00 and GPIO04 registers will not be reset by WDT reset Each GPIO pin can be individually programmed to be WDT reset tolerant or not.
7 :0	RW	**Port GPIOA[7:0] WDT reset tolerance enable** 0: GPIO00 and GPIO04 registers will be reset by WDT reset 1: GPIO00 and GPIO04 registers will not be reset by WDT reset Each GPIO pin can be individually programmed to be WDT reset tolerant or not.

Offset: 20h		GPIO20: Extended GPIO Data Value Register Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOH[7:0] data register**
23:16	RW	**Port GPIOG[7:0] data register**
15:8	RW	**Port GPIOF[7:0] data register**
7 :0	RW	**Port GPIOE[7:0] data register**

Offset: 24h		GPIO24: Extended GPIO Direction Register Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOH[7:0] direction control** 0: Select input mode 1: Select output mode
23:16	RW	**Port GPIOG[7:0] direction control** 0: Select input mode 1: Select output mode
15:8	RW	**Port GPIOF[7:0] direction control** 0: Select input mode 1: Select output mode
7 :0	RW	**Port GPIOE[7:0] direction control** 0: Select input mode 1: Select output mode

Offset: 28h		GPIO28: Extended GPIO Interrupt Enable Register Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOH[7:0] interrupt enable** 0: Disable interrupt 1: Enable interrupt
23:16	RW	**Port GPIOG[7:0] interrupt enable** 0: Disable interrupt 1: Enable interrupt
15:8	RW	**Port GPIOF[7:0] interrupt enable** 0: Disable interrupt 1: Enable interrupt
7 :0	RW	**Port GPIOE[7:0] interrupt enable** 0: Disable interrupt 1: Enable interrupt

Offset: 2Ch		GPIO2C: Extended GPIO Interrupt Sensitivity Type 0 Register Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOH[7:0] interrupt sensitivity type 0 selection** 0: Select falling-edge or level-low trigger mode 1: Select rising-edge or level-high trigger mode
23:16	RW	**Port GPIOG[7:0] interrupt sensitivity type 0 selection** 0: Select falling-edge or level-low trigger mode 1: Select rising-edge or level-high trigger mode
15:8	RW	**Port GPIOF[7:0] interrupt sensitivity type 0 selection** 0: Select falling-edge or level-low trigger mode 1: Select rising-edge or level-high trigger mode
7 :0	RW	**Port GPIOE[7:0] interrupt sensitivity type 0 selection** 0: Select falling-edge or level-low trigger mode 1: Select rising-edge or level-high trigger mode

Offset: 30h		GPIO30: Extended GPIO Interrupt Sensitivity Type 1 Register Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOH[7:0] interrupt sensitivity type 1 selection** 0: Select edge trigger mode 1: Select level trigger mode
23:16	RW	**Port GPIOG[7:0] interrupt sensitivity type 1 selection** 0: Select edge trigger mode 1: Select level trigger mode
15:8	RW	**Port GPIOF[7:0] interrupt sensitivity type 1 selection** 0: Select edge trigger mode 1: Select level trigger mode
7 :0	RW	**Port GPIOE[7:0] interrupt sensitivity type 1 selection** 0: Select edge trigger mode 1: Select level trigger mode

Offset: 34h		GPIO34: Extended GPIO Interrupt Sensitivity Type 2 Register Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOH[7:0] interrupt sensitivity type 2 selection** 0: Select edge or level trigger mode 1: Select dual-edge trigger mode
23:16	RW	**Port GPIOG[7:0] interrupt sensitivity type 2 selection** 0: Select edge or level trigger mode 1: Select dual-edge trigger mode
15:8	RW	**Port GPIOF[7:0] interrupt sensitivity type 2 selection** 0: Select edge or level trigger mode 1: Select dual-edge trigger mode
7 :0	RW	**Port GPIOE[7:0] interrupt sensitivity type 2 selection** 0: Select edge or level trigger mode 1: Select dual-edge trigger mode

Offset: 38h		GPIO38: Extended GPIO Interrupt Status Register Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOH[7:0] interrupt status register** Read 0: No interrupt pending Read 1: interrupt pending Write 0: No operation Write 1: Clear interrupt status flag
23:16	RW	**Port GPIOG[7:0] interrupt status register** Read 0: No interrupt pending Read 1: interrupt pending Write 0: No operation Write 1: Clear interrupt status flag
15:8	RW	**Port GPIOF[7:0] interrupt status register** Read 0: No interrupt pending Read 1: interrupt pending Write 0: No operation Write 1: Clear interrupt status flag

to next page

from previous page

7 :0	RW	**Port GPIOE[7:0] interrupt status register** Read 0: No interrupt pending Read 1: interrupt pending Write 0: No operation Write 1: Clear interrupt status flag

Offset: 3Ch	GPIO3C: Extended GPIO Reset Tolerant Register	Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOH[7:0] WDT reset tolerance enable** 0: GPIO20 and GPIO24 registers will be reset by WDT reset 1: GPIO20 and GPIO24 registers will not be reset by WDT reset Each GPIO pin can be individually programmed to be WDT reset tolerant or not.
23:16	RW	**Port GPIOG[7:0] WDT reset tolerance enable** 0: GPIO20 and GPIO24 registers will be reset by WDT reset 1: GPIO20 and GPIO24 registers will not be reset by WDT reset Each GPIO pin can be individually programmed to be WDT reset tolerant or not.
15:8	RW	**Port GPIOF[7:0] WDT reset tolerance enable** 0: GPIO20 and GPIO24 registers will be reset by WDT reset 1: GPIO20 and GPIO24 registers will not be reset by WDT reset Each GPIO pin can be individually programmed to be WDT reset tolerant or not.
7 :0	RW	**Port GPIOE[7:0] WDT reset tolerance enable** 0: GPIO20 and GPIO24 registers will be reset by WDT reset 1: GPIO20 and GPIO24 registers will not be reset by WDT reset Each GPIO pin can be individually programmed to be WDT reset tolerant or not.

Offset: 40h	GPIO40: GPIO Debounce Setting Register #1	Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOD[7:0] debounce setting register #1**
23:16	RW	**Port GPIOC[7:0] debounce setting register #1**
15:8	RW	**Port GPIOB[7:0] debounce setting register #1**
7 :0	RW	**Port GPIOA[7:0] debounce setting register #1**

Offset: 44h	GPIO44: GPIO Debounce Setting Register #2	Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOD[7:0] debounce setting register #2**
23:16	RW	**Port GPIOC[7:0] debounce setting register #2**
15:8	RW	**Port GPIOB[7:0] debounce setting register #2**
7 :0	RW	**Port GPIOA[7:0] debounce setting register #2**

Offset: 48h	GPIO48: Extended GPIO Debounce Setting Register #1	Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOH[7:0] debounce setting register #1**
23:16	RW	**Port GPIOG[7:0] debounce setting register #1**
15:8	RW	**Port GPIOF[7:0] debounce setting register #1**
7 :0	RW	**Port GPIOE[7:0] debounce setting register #1**

Offset: 4Ch		GPIO4C: Extended GPIO Debounce Setting Register #2 Init = 0
Bit	**R/W**	**Description**
31:24	RW	**Port GPIOH[7:0] debounce setting register #2**
23:16	RW	**Port GPIOG[7:0] debounce setting register #2**
15:8	RW	**Port GPIOF[7:0] debounce setting register #2**
7 :0	RW	**Port GPIOE[7:0] debounce setting register #2**

The definition of interrupt trigger mode registers:
GPIO0C ~ GPIO14, GPIO2C ~ GPIO34 are as follows:

Type 2	Type 1	Type 0	Interrupt Trigger Mode
(14/34h)	(10/30h)	(0C/2Ch)	
0	0	0	falling-edge trigger mode
0	0	1	rising-edge trigger mode
0	1	0	level-low trigger mode
0	1	1	level-high trigger mode
1	x	x	dual-edge trigger mode

The definition of debounce setting registers GPIO40 ~ GPIO4C are as follows:

Debounce Setting #2	Debounce Setting #1	Function
0	0	No Debounce
0	1	Select GPIO50 as debounce timer
1	0	Select GPIO54 as debounce timer
1	1	Select GPIO58 as debounce timer

Offset: 50h Offset: 54h Offset: 58h		GPIO50: Debounce Timer Setting Register #1 Init = 0 GPIO54: Debounce Timer Setting Register #2 Init = 0 GPIO58: Debounce Timer Setting Register #3 Init = 0
Bit	**Attr.**	**Description**
31:24		**Reserved (0)**
23:0	RW	**Debounce Timer Value** This register defines the timer period for GPIO input sampling. The sampling timer period is : Debounce time = PCLK cycle time * Debounce timer value

24 Real Time Clock (RTC)

24.1 Overview

Real Time Clock (RTC) is a flexible real time clock. When the system enters sleeping mode, the PCLK clock of APB bus can be gated and the RTC keeps on counting. This mechanism promises the lowest power consumption when the system is asleep.

Furthermore, RTC provides separated second, minute, hour and day counters. The second counter is toggled once every second, the minute counter is toggled once every minute, the hour counter is toggled once every hour and the day counter is toggled once every day. The separated counter mechanism reduces the complexity of software. The software doesnt need to calculate the second, minute, or hour information. The software only needs to read counter values and calculate the current time.

RTC provides second, minute, hour, day, and clock alarm function. When turned on the second alarm function, the RTC will auto trigger an interrupt each second. Also, the auto minute, hour alarm can be turned on. The function is useful for implementing a clock.

RTC totally implements 6 sets of 32-bit registers, which are listed below, to program the various supported functions. Each register has its own specific offset value, ranging from 0x00 to 0x14h, to derive its physical address location.

Base address of RTC = 0x1E78_1000
Physical address = (Base address of RTC) + Offset

RTC00: Counter Status Register
RTC04: Clock Alarm Register
RTC08: Reload Value Register
RTC0C: Control Register
RTC10: Restart Register
RTC14: Reset Register

24.2 Features

- Directly connected to APB bus
- Clock source is divided from 24MHz clock input
- 24-Hour timer mode with highest precision of a second
- Programmable alarm with interrupt generation
- Maskable interrupt
- No battery backup supported
- Precision ≈ 50ppm (24MHz input precision), approximately 1 second deviation for each 12 hours. So it is recommended to sync the time with the time server for couple days or couple weeks.

24.3 Registers : Base Address = 0x1E78:1000

Offset: 00h		RTC00: Counter Status Register Init = X
Bit	**R/W**	**Description**
31:17	R	**DayCnt: Status of Day Counter.** The DayCnt register is the RTC day counter register. Before the RTC can be enabled, user can set the counter value by setting restart register to reload value. When RTC is enabled, the DayCnt value always increases by day. This register can calculate the correct date when each system was woke up.
16:12	R	**HourCnt: Status of Hour Counter.** The HourCnt register is the RTC hour counter register. After RTC is enabled, the HourCnt value increases by hour. When the HourCnt value exceeds 23, the value is reset to zero. The HourCnt range is 0 ~ 23. If the RTC is disabled, the HourCnt will hold the value.
11:6	R	**MinuCnt: Status of Minute Counter.** The MinuCnt register is the RTC minute counter register. After RTC is enabled, the MinuCnt value increases by minute. When the MinuCnt value exceeds 59, the value is reset to zero. The MinuCnt range is 0 ~ 59. If the RTC is disabled, the MinuCnt will hold the value.
5 :0	R	**SecCnt: Status of Second Counter.** The SecCnt register is the RTC second counter register. After RTC is enabled, the SecCnt value increases by second. When the SecCnt value exceeds 59, the value is reset to zero. The SecCnt range is 0 ~ 59. If the RTC is disabled, the SecCnt will hold the value.

Offset: 04h		RTC04: Clock Alarm Register Init = X
Bit	**R/W**	**Description**
31:13		**Reserved (0)**
16:12	RW	**The hour alarm register.** The register is an RTC hour alarm register. If user wants to trigger RTC alarm interrupt at 12:10:15, the register needs to set 0xC. If the register value exceeds 0x17, RTC alarm will never be triggered. But the RTC counter keeps on counting.
11:6	RW	**The minute alarm register.** The register is an RTC minute alarm register. If user wants to trigger rtc alarm interrupt at 12:10:15, the register needs to set 0xA. If the register value exceeds 0x3B, RTC alarm will never be triggered. But the RTC counter keeps on counting.
5 :0	RW	**The second alarm register.** The register is an RTC second alarm register. If user wants to trigger rtc alarm interrupt at 12:10:15, the register needs to set 0xF. If the register value exceeds 0x3B, RTC alarm will never be triggered. But the RTC counter keeps on counting.

Offset: 08h		RTC08: Reload Value Register Init = X
Bit	**R/W**	**Description**
31:17	RW	**The reload value of day.** The register is the RTC reload value of day. User can adjust the clock by setting reload value, and restart RTC. After restart RTC, the reload value will be reload into day counter. The method of restart is described on restart register.
16:12	RW	**The reload value of hour.** The register is the RTC reload value of day. User can adjust the clock by setting reload value, and restart RTC. After restart RTC, the reload value will be reload into hour counter. The method of restart is described on restart register.

to next page

from previous page

11:6	RW	**The reload value of minute.** The register is the RTC reload value of day. User can adjust the clock by setting reload value, and restart RTC. After restart RTC, the reload value will be reload into minute counter. The method of restart is described on restart register.
5 :0	RW	**The reload value of second.** The register is the RTC reload value of day. User can adjust the clock by setting reload value, and restart RTC. After restart RTC, the reload value will be reload into second counter. The method of restart is described on restart register.

Offset: 0Ch — RTC0C: Control Register — Init = X

Bit	R/W	Description
31:6		**Reserved (0)**
5	R	**The restart status** 1: Now, RTC is reloading the reload value into counter. 0: not restart period
4	RW	**Enable day alarm** 1: enable 0: disable
3	RW	**Enable hour alarm** 1: enable 0: disable
2	RW	**Enable minute alarm** 1: enable 0: disable
1	RW	**Enable second alarm** 1: enable 0: disable
0	RW	**RTC enable** 1: enable 0: disable Default setting is disabled

Offset: 10h — RTC10: Restart Register — Init = X

Bit	R/W	Description
31:8		**Reserved (0)**
7 :0	W	**Restart Enable** When 0x5A value is written to this register, the reload value register will be loaded into counter of RTC whenever RTC function is enabled (RTC0C [0]) or not. After write cycle finish, RTC0C [5] will auto reset to zero.

Offset: 14h — RTC14: Reset Register — Init = X

Bit	R/W	Description
31:8		**Reserved (0)**
7 :0	RW	**Reset Enable** Writing data 0x99 to this register will reset RTC immediately.

24.4 Operation

There are 3 programming methods to use the RTC. The major difference is whether to wait for the reload status or not.

Update the RTC whenever the RTC is under reload busy state may cause RTC dead lock. It needs a long reset procedure to recover the dead lock condition.

After enabling the RTC, the programmer can program the HourArm, MinArm, SecArm registers and the Control register to enable the RTC alarm interrupt and auto alarm function, if you need.

24.4.1 Programming Mode 1: No waiting restart status

Initial Sequence

1. Enable reset RTC, write RTC14 = 0x99
2. Set current time to reload register RTC08
3. Delay 1 second
4. Clear reset RTC, write RTC14 = 0x0
5. Enable Restart, write RTC10 = 0x5A
6. Enable RTC, write RTC0C.bit[0] = 1

Update Sequence

1. Enable reset RTC, write RTC14 = 0x99
2. Set current time to reload register RTC08
3. Delay 1 second
4. Clear reset RTC, write RTC14 = 0x0
5. Enable Restart, write RTC10 = 0x5A

24.4.2 Programming Mode 2: Waiting restart status at the start

Initial Sequence

1. Enable reset RTC, write RTC14 = 0x99
2. Delay 1 second
3. Clear reset RTC, write RTC14 = 0x0
4. Set current time to reload register RTC08
5. Enable Restart, write RTC10 = 0x5A
6. Enable RTC, write RTC0C.bit[0] = 1

Update Sequence

1. Waiting Restart finished, wait until RTC0C.bit[5] = 0
2. Set current time to reload register RTC08
3. Enable Restart, write RTC10 = 0x5A

24.4.3 Programming Mode 3: Waiting restart status at the end

Initial Sequence

1. Enable reset RTC, write RTC14 = 0x99
2. Delay 1 second
3. Clear reset RTC, write RTC14 = 0x0
4. Set current time to reload register RTC08
5. Enable Restart, write RTC10 = 0x5A
6. Enable RTC, write RTC0C.bit[0] = 1
7. Waiting Restart finished, wait until RTC0C.bit[5] = 0, it needs about 0 $\sim$ 3 seconds

Update Sequence

1. Set current time to reload register RTC08
2. Enable Restart, write RTC10 = 0x5A
3. Waiting Restart finished, wait until RTC0C.bit[5] = 0, it needs about 0 $\sim$ 3 seconds

25 Timer Controller

25.1 Overview

Timer Controller (TMC) includes 3 sets of 32-bit decrement counters, based on either APB clock or external clock regarding to the definition of the clock section. Each counter is equipped with are two sets of matching registers. When any one of the Match registers equal to the corresponding counter value, a timer interrupt will be triggered. Each counter also can be programmed to trigger an interrupt or not whenever overflow occurs. Furthermore, all the counter values can be read back at any time.

TMC totally implements 13 sets of 32-bit registers, which are listed below, to program the various supported functions. Each register has its own specific offset value, ranging from 0x00 to 0x30h, to derive its physical address location.

Base address of Timer = 0x1E78_2000
Physical address = (Base address of Timer) + Offset

TMC00: Counter #1 Status Register
TMC04: Counter #1 Reload Value Register
TMC08: Counter #1 First Matching Register
TMC0C: Counter #1 Second Matching Register
TMC10: Counter #2 Status Register
TMC14: Counter #2 Reload Value Register
TMC18: Counter #2 First Matching Register
TMC1C: Counter #2 Second Matching Register
TMC20: Counter #3 Status Register
TMC24: Counter #3 Reload Value Register
TMC28: Counter #3 First Matching Register
TMC2C: Counter #3 Second Matching Register
TMC30: Control Register

25.2 Features

- Directly connected to APB Bus
- Built-in 3 sets of 32-bit timer modules
- Free-running or periodic mode
- Maskable interrupts

25.3 Registers : Base Address = 0x1E78:2000

Offset: 00h		TMC00: Counter #1 Status Register Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Counter #1 Counter status** This register stores the current status of counter #1. When timer enable bit TMC30 [0] is set, the counter will start to decrement. CPU can modify the register value at any time.

Offset: 04h	TMC04: Counter #1 Reload Value Register	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Counter #1 reload value register** When counter #1 decrease to zero, the reload value will be loaded to counter #1 automatically.

Offset: 08h	TMC08: Counter #1 First Matching Register	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**First set match register** When counter #1 match this register, the timer will generate an edge triggered interrupt to CPU.

Offset: 0Ch	TMC0C: Counter #1 Second Matching Register	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Secondary match register** When counter #1 match this register, the timer will generate an edge triggered interrupt to CPU.

Offset: 10h	TMC10: Counter #2 Status Register	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Counter #2 Counter status** This register stores the current status of counter #2. When timer enable bit TMC30 [4] is set, the counter will start to decrement. CPU can modify the register value at any time.

Offset: 14h	TMC14: Counter #2 Reload Value Register	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Counter #1 reload value register** When counter #2 decrease to zero, the reload value will be reload to counter #2 automatically.

Offset: 18h	TMC18: Counter #2 First Matching Register	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**First set match register** When counter #2 match this register, the timer will generate an edge trigger interrupt to CPU.

Offset: 1Ch	TMC1C: Counter #2 Second Matching Register	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Secondary match register** When counter #2 match this register, the timer will generate an edge trigger interrupt to CPU.

Offset: 20h	TMC20: Counter #3 Status Register	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Counter #3 Counter status** This register stores the current status of counter #3. When timer enable bit TMC30 [8] is set, the counter will start to decrement. CPU can modify the register value at any time.

Offset: 24h	TMC24: Counter #3 Reload Value Register	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Counter #1 reload value register** When counter #3 decrease to zero, the reload value will be reload to counter #3 automatically.

Offset: 28h	TMC28: Counter #3 First Matching Register	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**First set match register** When counter #3 match this register, the timer will generate an edge trigger interrupt to CPU.

Offset: 2Ch	TMC2C: Counter #3 Second Matching Register	Init = 0
Bit	**R/W**	**Description**
31:0	RW	**Secondary match register** When counter #3 match this register, the timer will generate an edge trigger interrupt to CPU.

Offset: 30h	TMC30: Control Register	Init = 0
Bit	**R/W**	**Description**
31:11		**Reserved (0)**
10	RW	**Enable Interrupt for Timer/Counter #3** 0: when overflow occur, timer dont generate interrupt 1: when overflow occur, interrupt will be generated
9	RW	**Clock selection for Timer/Counter #3** Counter is base on selected clock to down count 0: APB clock (PCLK) 1: External clock (1 MHz)
8	RW	**Timer enable for Timer/Counter #3** 0: disable 1: enable When timer is disabled, all action for counter, reload, and interrupt will be gated.
7		**Reserved (0)**
6	RW	**Enable Interrupt for Timer/Counter #2** 0: when overflow occur, timer dont generate interrupt 1: when overflow occur, interrupt will be generated
5	RW	**Clock selection for Timer/Counter #2** Counter is base on selected clock to down count 0: APB clock (PCLK) 1: External clock (1 MHz)

to next page

from previous page

4	RW	**Timer enable for Timer/Counter #2** 0: disable 1: enable When timer is disabled, all action for counter, reload, and interrupt will be gated.
3		**Reserved (0)**
2	RW	**Enable Interrupt for Timer/Counter #1** 0: when overflow occur, timer dont generate interrupt 1: when overflow occur, interrupt will be generated
1	RW	**Clock selection for Timer/Counter #1** Counter is base on selected clock to down count 0: APB clock (PCLK) 1: External clock (1 MHz)
0	RW	**Timer enable for Timer/Counter #1** 0: disable 1: enable When timer is disabled, all action for counter, reload, and interrupt will be gated.

25.4 Operation

Reload, Match1, Match2 and Control[Interrupt] must be set when timer is used. Reload controls the period between twice overflow. For example, if 0x02 value be set to Reload and then enable timer Control[Enable], the sequence of counter is 2,1,0,2,1,....
A interrupt can be generated when timer counter reach zero, if Control[Interrupt] be set.

Sequence :

1. Set Reload
2. Set Control[Interrupt]
3. Enable Timer, Control[Enable]

25.5 Programming Note

TMC30 bit[2] in timer controller could not enable/disable the timer match register function to issue interrupt. So, if you don't want to use the match register function, please set the match register values to 0xFFFFFFFF.

26 UART (16550)

26.1 Overview

AST2050 / AST1100 integrates two sets of UART (Universal Asynchronous Receiver/Transmitter) providing serial communication capabilities with other external devices, like another computer using a serial cable based on RS232 protocol. This core is designed to be compatible with the industry defector standard — 16550 UART. The two sets of UART are equipped with a 16x8 FIFO that can be programmed to be enabled or disabled. The supported baud rates are also programmable.

Each unit of UART totally implements 12 sets of 32-bit registers, which are listed below, to program the various supported functions including character length selection, baud rate selection, interrupt generation, and parity generation/checking. Each register has its own specific offset value, ranging from 0x00 to 0x14h, to derive its physical address location.

Base Address of UART1 = 0x1E78_3000
Base Address of UART2 = 0x1E78_4000
Register Address of UART = (Base Address of UART) + Offset

UART_RBR: Receiving Buffer Register (DLAB = 0)
UART_THR: Transmit Holding Register (DLAB = 0)
UART_IER: Interrupt Enable Register (DLAB = 0)
UART_IIR: Interrupt Identity Register
UART_FCR: FIFO Control Register
UART_LCR: Line Control Register
UART_MCR: Modem Control Register
UART_LSR: Line Status Register
UART_MSR: Modem Status Register
UART_SCR: Scratch Register
UART_DLL: Divisor Latch Low Register : (DLAB = 1)
UART_DLH: Divisor Latch High Register : (DLAB = 1)

The UART packet frame format is shown as Figure 68.

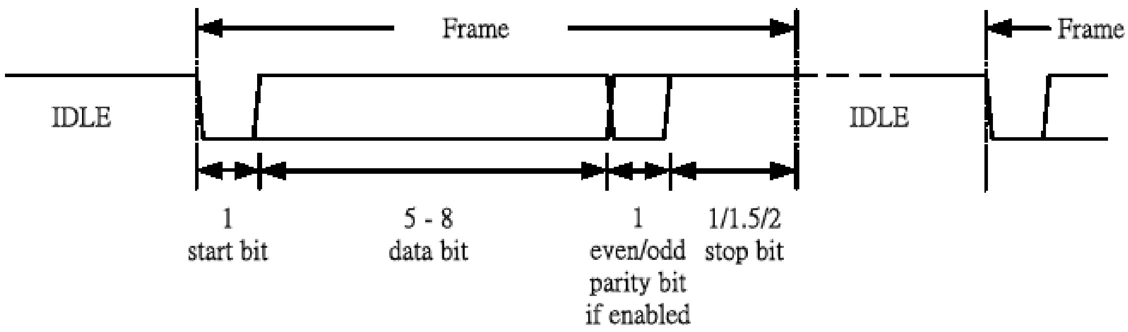


Figure 68: UART Packet Frame

26.2 Features

- Directly connected to APB bus
- Support two UART with full flow control pins (one is with dedicated flow control pins, the other is shared with GPIO pins)
- Separate transmit & receive FIFO buffer (16x8) to reduce CPU interrupts

- Support up to 115.2K baud-rate
- Programmable baud rate generator
- Standard asynchronous communication bits — Stat/Stop/Parity
- Independent masking of transmit FIFO, receive FIFO, receiving timeout and error condition Interrupts
- False start-bit detection
- Line break generation and detection
- Fully programmable serial interface characteristics:
  - 5/6/7/8 data length
  - Even, odd and none parity generation and detection
  - 1/2 stop-bit generation
- Extended diagnostic Loopback Mode allows testing more Modem Control and Auto Flow Control features

26.3 Registers : Base Address = (UART1)0x1E78:3000 or (UART2)0x1E78:4000

Offset: 00h	UART_RBR: Receiving Buffer Register (DLAB = 0)	Init = 0
Offset: 00h	UART_THR: Transmit Holding Register (DLAB = 0)	Init = 0

Bit	Attr.	Description
31:8		**Reserved (0)**
7 :0	R	**UART_RBR: Receiving Buffer Register** The UART_RBR is a read-only register that contains the data byte received on the serial input port. The data in register is valid only if the Data Ready bit in the Line status Register (UART_LSR) is set. When the **FIFOs are programmed OFF**, the data in the UART_RBR must be read before the next data arrives; otherwise it will be overwritten, resulting in an overrun error. When the **FIFOs are programmed ON**, this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.
7:0	W	**UART_THR: Transmit Holding Register** The UART_THR is a write-only register that contains data to be transmitted on the serial output port. Data can be written to the UART_THR any time that the THR Empty (**THRE**) bit of the Line Status Register (UART_LSR) is set. When the **FIFOs are programmed OFF** and THRE is set, writing a single character to the UART_THR clears the THRE. Any additional writes to the UART_THR before the THRE is set again causes the UART_THR data to be overwritten. When the **FIFOs are programmed ON** and THRE is set, 16 bytes of data may be written to the UART_THR before the FIFO is full. Any attempt to write data when the FIFO is full results in the write data being lost.

Offset: 04h	UART_IER: Interrupt Enable Register (DLAB = 0) Init = 0	
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	RW	**PTIME: Enable Programmable THRE Interrupt Mode** 0: Disable THRE interrupt mode 1: Enable THRE interrupt mode
6 :4		**Reserved (0)**
3	RW	**EDSSI: Enable Modem Status Interrupt** 0: Disable interrupt 1: Enable interrupt
2	RW	**ELSI: Enable Receiver Line Status Interrupt** 0: Disable interrupt 1: Enable interrupt
1	RW	**ETBEI: Enable Transmitter Holding Register Empty Interrupt** 0: Disable interrupt 1: Enable interrupt
0	RW	**ERBFI: Enable Received Data Available Interrupt** 0: Disable interrupt 1: Enable interrupt

Offset: 08h	UART_IIR: Interrupt Identity Register Init = 0x01	
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7:6	R	**FIFO-Enabled Bits** 00: FIFOs disabled 11: FIFOs enabled
5:4		**Reserved (0)**
3:1	R	**Interrupt Decoding Table** The content of this register can be used to identify the source of the current interrupt based on the following: 000: Modem Status Changed 001: UART_THR empty 010: Received Data Available 011: Receiver Status 110: Character Time Out For more information about Interrupt Identity, see the following Table for detailed description.
0	R	Indicates that an interrupt is pending when its logic 0. When its 1, no interrupt is pending.
Note : The IIR enables the programmer to retrieve what is the current highest priority pending interrupt.		

UART Interrupt Type Decoding				
Bit [3:1]	Priority	Interrupt Type	Interrupt Source	Interrupt Clear Control
011	Highest	Receiver line status	Parity, Overrun, Framing errors or Break Interrupt.	Reading the Line Status Register.
010	2nd	Received Data available	FIFO OFF: Receiver data available FIFO ON: RX FIFO trigger level reached	FIFO OFF: Reading the RBR FIFO ON: FIFO drops below the trigger level
110	3rd	Character Timeout indication	Theres at least 1 character in the FIFO but no character has been input to the FIFO or read from it for the last 4 Character times.	Reading the RBR.
001	4th	Transmitter Holding register empty	IER[7] = 0 : THR Empty IER[7] = 1 : TX FIFO at or below threshold	Reading the IIR or writing into THR (FIFO or IER[7] disabled) or TX FIFO above threshold (FIFO or IER[7] enabled)
000	5th	Modem status	nCTS (Clear to send), nDSR (data set ready), nRI (Ring indicator), nDCD (Data center detect)	Reading the UART_MSR.

Offset: 08h	**UART_FCR: FIFO Control Register**	**Init = 0**
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7:6	W	**Define the Receiver FIFO Interrupt trigger level.** 00: 1 byte received 01: 4 bytes received 10: 8 bytes received 11: 14 bytes received
5:4	W	**Define the Transmitter FIFO Interrupt trigger level.** 00: FIFO empty 01: 2 bytes in FIFO 10: FIFO 1/4 full 11: FIFO 1/2 full
3		**Reserved (0)**
2	W	**Transmit FIFO Reset** Writing 1 to this bit clears the Transmitter FIFO and resets its logic.
1	W	**Receive FIFO Reset** Writing 1 to this bit clears the Receiver FIFO and resets its logic.
0	W	**Enable UART FIFO** 0: Disable FIFO 1: Enable FIFO Changing the value of this register will always reset UART FIFO immediately.

Note :
The FCR allows selection of the FIFO trigger level (the number of bytes in FIFO required to enable the Received Data Available interrupt). In addition, the FIFOs can be cleared using this register.

Offset: 0Ch		UART_LCR: Line Control Register Init = 0
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	RW	**DLAB: Divisor latch access bit** 0: The normal registers are accessed. 1: The divisor latches can be accessed. Setting this bit will enable the reading and writing of the Divisor Latch register (UART_DLL and UART_DLH) to set the baud rate of the UART. This bit must be cleared after initial baud rate setup in order to access other registers.
6	RW	**Break Control bit.** 0: break is disabled. 1: When not in Loopback Mode, the serial out is forced into logic '0' (break state). When in Loopback Mode, the break condition is internally looped back to the receiver.
5		**Reserved (0)**
4	RW	**EPS: Parity mode selection** 0: Select odd parity mode (odd number of "1" for data and parity combined) 1: Select even parity mode (even number of "1" for data and parity combined)
3	RW	**PEN: Enable parity bit** 0: Disable parity bit 1: Enable parity bit
2	RW	**STOP: Number of stop bits transmitted** 0: 1 stop bit. 1: 1.5 stop bits when 5-bit character length selected and 2 bits otherwise Note that the receiver always checks the first stop bit only.
1:0	RW	**CLS: Select number of bits per character** 00: 5 bits. 01: 6 bits. 10: 7 bits. 11: 8 bits.

Offset: 10h		UART_MCR: Modem Control Register Init = 0
Bit	**R/W**	**Description**
31:5		**Reserved (0)**
4	RW	**Loopback mode.** 0: normal operation. 1: loopback mode. When in loopback mode, the Serial Output Signal (TXD) is set to logic '1'. The signal of the transmitter shift register is internally connected to the input of the receiver shift register. The following connections are made: nDTR → nDSR nRTS → nCTS Out1 → nRI Out2 → nDCD
3	RW	**Out2.** In loopback mode, connected to Data Carrier Detect(nDCD) input.
2	RW	**Out1.** In loopback mode, connected to Ring Indicator (nRI)signal input.
1	RW	**Request To Send (nRTS) signal control.** 0: nRTS is '1' 1: nRTS is '0'

to next page

from previous page

0	RW	**Data Terminal Ready (nDTR) signal control.** 0: nDTR is '1' 1: nDTR is '0'

Offset: 14h	UART_LSR: Line Status Register	Init = 0x60
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	R	**Error in Receiver FIFO** 1: There is at least one parity error, framing error, or break indication in the FIFO. This bit is only active when FIFOs are enabled. This bit is cleared when the UART_LSR is read. 0: Otherwise.
6	R	**Transmitter empty** 1: The UART_THR or FIFO and the Transmitter Shift Register are both empty. 0: Otherwise.
5	R	**THRE: Transmitter holding register empty** 1: When FIFO or THRE mode is disabled, the UART_THR or FIFO is empty. When FIFO and THRE mode is enabled, the transmitter FIFO is full. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. 0: Otherwise.
4	R	**BI: Break interrupt** 1: The serial input is held in a logic 0 state for longer than the sum of start time + data bits + parity + stop bits. A break condition on serial input causes one and only one character, consisting of all zeros, to be received by the UART. 0: No break condition in the current character.
3	R	**FE: Framing error** 1: There is a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data. 0: No framing error in the current character.
2	R	**PE: Parity error** 1: There is a parity error in the receiver if the Parity Enable is set. 0: No parity error in the current character.
1	R	**OE: Overrun error** 1: An overrun error has occurred because a new data character was received before the previous data was read. In the FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver. 0: No overrun state.
0	R	**DR: Data ready** 1: The receiver contains at least one character in the UART_RBR or the receiver FIFO. 0: The UART_RBR is read or the receiver FIFO is empty.

Offset: 18h	UART_MSR: Modem Status Register	Init = 0
Bit	**R/W**	**Description**
31:8	R	**Reserved (0)**
7	R	**Complement of the nDCD input or equals to Out2(MCR[3]) in loopback mode.**
6	R	**Complement of the nRI input or equals to Out1(MCR[2]) in loopback mode.**
5	R	**Complement of the nDSR input or equals to nDTR(MCR[0]) in loopback mode.**
4	R	**Complement of the nCTS input or equals to nRTS(MCR[1]) in loopback mode.**

to next page

from previous page

3	R	**Delta Data Carrier Detect (DDCD) indicator** 1: The nDCD line has changed its state since the last time the CPU read the MSR. In loopback mode, DDCD reflects changes on MCR bit 3 (Out2)
2	R	**Trailing Edge of Ring Indicator (TERI) detector.** The nRI line has changed its state from low to high state since the last time the CPU read the MSR. In loopback mode, TERI reflects when MCR bit 2 (Out1) has changed state from a high to a low.
1	R	**Delta Data Set Ready (DDSR) indicator** 1: The nDSR line has changed its state since the last time the CPU read the MSR. In loopback mode, DDSR reflects changes on MCR bit 0 (DTR)
0	R	**Delta Clear To Send (DCTS) indicator** 1: The nCTS line has changed its state since the last time the CPU read the MSR. In loopback mode, DCTS reflects changes on MCR bit 1 (RTS)
Note : The register displays the current state of the modem control lines. Also, four bits also provide an indication in the state of one of the modem status lines. These bits are set to '1' when a change in corresponding line has been detected and they are reset when the register is being read.		

Offset: 1Ch		UART_SCR: Scratch Register — Init = 0
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7 :0	RW	**Scratch bits** This register can be used as a temporary storage, no specific definition.

26.3.1 UART_DLL/UART_DLH

In addition, there are 2 Clock Divisor registers that together to form one 16-bits, read/write, Divisor Latch register that contains the baud rate.
The registers can be accessed when the 7th (DLAB) bit of the Line Control Register idivisor for the UART s set to '1'. At this time the above registers at addresses 0x0 & 0x4 can't be accessed.

Setting the 7th bit of UART_LCR to 1 can access the divisor latches. You should restore this bit to 0 after setting the divisor latches in order to restore access to the other registers that occupy the same addresses.

The 2 bytes form one 16-bit register, which is internally accessed as a single number. You should therefore set all 2 bytes of the register to ensure normal operation. The register is set to the default value of 0 on reset, which disables all serial I/O operations in order to ensure explicit setup of the register in the software. The value set should be equal to

Baud rate = 24MHz / (16 * divisor).

The internal counter starts to work when the LSB of DL is written, so when setting the divisor, write the MSB first and the LSB last.

Offset: 00h		UART_DLL: Divisor Latch Low Register : (DLAB = 1) — Init = 0
Bit	**R/W**	**Description**
31:8		**Reserved.**
7:0	RW	**The LSB of the baud-rate divisor latch.**

Offset: 04h		UART_DLH: Divisor Latch High Register : (DLAB = 1) Init = 0
Bit	R/W	Description
31:8		**Reserved.**
7:0	RW	**The MSB of the baud-rate divisor latch.**

27 Watchdog Timer

27.1 Overview

Watchdog Timer (WDT) is designed to prevent system deadlock. In general, the WDT must be restarted before WDT timeout. Whenever timeout occurs, WDT will generate 3 signals:

- **System reset signal**: to reset system
- **Interrupt signal**: to interrupt CPU
- **External signal**: **Only work for A1 version chip**

WDT totally implements xx sets of 32-bit registers, which are listed below, to program the various functions supported by WDT. Each register has its own specific offset value to derive its physical address location.

Base address of WDT = 0x1E78_5000
Physical address = (Base address of WDT) + Offset

WDT00: Counter Status Register
WDT04: Counter Reload Value Register
WDT08: Counter Restart Register
WDT0C: Control Register
WDT10: Timeout Status Register
WDT14: Clear Timeout Status Register
WDT18: Reset Width Register

27.2 Features

- Directly connected to APB bus
- Watchdog function
- Built-in 32-bit programmable counter
- Generate either interrupt or reset after counting down to zero (programmable)

27.3 Registers : Base Address = 0x1E78:5000

Offset: 00h		**WDT00: Counter Status Register** — **Init = 0x03EF1480**
Bit	**R/W**	**Description**
31:0	R	**Counter status** This register stores the current status of counter. After HRST_N, this register is set to 0x3EF1480. When the programmer writes 0x4755 to Restart register, Reload register will be loaded into this register. Counter starts to decrease once WDT0C[0] enable bit is set. If watchdog timer is disabled, it will hold the value.

Offset: 04h		WDT04: Counter Reload Value Register Init = 0x03EF1480
Bit	**R/W**	**Description**
31:0	RW	**Counter reload value register** Reload register contains value which will be loaded into WDT00 register. When reset or restart, Reload value will be automatically loaded into WDT00 register.

Offset: 08h		WDT08: Counter Restart Register Init = 0
Bit	**R/W**	**Description**
31:16		**Reserved (0)**
15:0	W	**Restart register** Restart register is used to avoid system deadlock. If the 0x4755 value is written into this register, the Reload register will be loaded into WDT00 register and WDT00 register restarts to decrease if WDT0C[0] register is set.

Offset: 0Ch		WDT0C: Control Register Init = 0
Bit	**R/W**	**Description**
31:5		**Reserved (0)**
4	RW	**Clock select for WDT Counter** The counter will be decreased by selected clock. 0: PCLK 1: 1MHz clock source
3	RW	**wdt_ext: External signal enable after timeout** This signal is connected to external WDTRST output pin D9. If timeout occurs, and this bit is enabled, an active high output will be asserted. 0: disable 1: enable
2	RW	**wdt_intr: Interrupt enable after timeout** 0: disable 1: enable
1	RW	**Reset system after timeout** 0: disable 1: enable
0	RW	**WDT enable signal** 0: disable 1: enable

Offset: 10h		WDT10: Timeout Status Register Init = 0
Bit	**R/W**	**Description**
31:1		**Reserved (0)**
0	R	**Indicate timeout** Timeout register is a record. If the WDT had ever occurred Time out condition, this bit will be set automatically. 0: timeout never occur 1: timeout occur

Offset: 14h		WDT14: Clear Timeout Status Register Init = 0
Bit	**R/W**	**Description**
31:1		**Reserved (0)**

to next page

from previous page

0	W	**Clear timeout status** Write '1' value into this bit to clear Timeout Status register

Offset: 18h	WDT18: Reset Width Register	Init = 0xFF
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7 :0	RW	**Reset width** This register decides the asserting duration of wdt_intr, wdt_ext signal. The default value is 0xFF. It means the default asserting duration of wdt_intr, wdt_ext is 256 PCLK cycles. When WDT0C[1] enabled, the signal width generated only can have 1.25 us long at maximum.

27.4 Operation

To enable watchdog timer, the programmer needs to set Reload register first. Reload decides the period of timeout. The default value of Reload is 0x3EF1480, it means that in a 66MHz system, the period of timeout is one second. The programmer can change this, if you need. For example, setting Reload to 0xEC08CE00, it means the period of timeout is 1 minute and it guarantees the system will be reset after one minute. This can be used to avoid the deadlock of system. The programmer needs to write 0x4755 to Restart register before timeout, when WDT is enable.

After setting the Reload, Restart and WDT0C[4] bit, the programmer can enable WatchDog by write '1' to WDT0C[0] bit. And then watchdog timer starts to count down. The following steps are the summary.

1. disable watch dog timer
2. set Reload register
3. write 0x4755 to Restart register
4. set WDT0C[4] bit
5. enable watch dog timer

28 PWM & Fan Tacho Controller

28.1 Overview

Base Address of PWM & Fan Tach Controller = 0x1E78_6000
Physical address of register = (Base address of PWM & Fan Tach Controller) + Offset

PTCR00: General Control Register
PTCR04: Clock Control Register
PTCR08: Duty Control 0 Register
PTCR0C: Duty Control 1 Register
PTCR10: Type M Control 0 Register
PTCR14: Type M Control 1 Register
PTCR18: Type N Control 0 Register
PTCR1C: Type N Control 1 Register
PTCR20: Tach Source Register
PTCR28: Trigger Register
PTCR2C: Result Register
PTCR30: Interrupt Control Register
PTCR34: Interrupt Status Register
PTCR38: Type M Limit Register
PTCR3C: Type N Limit Register

28.2 Features

PWM Controller

- Support 4 PWM outputs
- Support both low-frequency and high-frequency PWM for fan speed control
- Duty cycle from 0 to 100% with 1/256 resolution incremental
- Support low-frequency PWM pulse stretching for fan speed measurements
- Shared with GPIO pins

Fan Tachometer Controller

- Directly connected to APB bus
- Support 16 tachometer inputs
- Measurement schemes: rising edge, falling edge or both edges
- Support Interrupt trigger when over fan speed limitation setting
- 4 tachometer input pins are dedicated, 12 tachometer input pins are shared with DVO input pins

28.3 Registers : Base Address = 0x1E78:6000

Offset: 00h		PTCR00: General Control Register Init = 0xXXXXX000
Bit	**R/W**	**Description**
31:16	RW	**Enable Fan Tach #15 ~ Fan Tach #0** Bit[31]: enable fan tach #15 0: disable 1: enable Bit[30]: enable fan tach #14 0: disable 1: enable ... Bit[16]: enable fan tach #0 0: disable 1: enable
15	RW	**Type selection of PWM D port** 0: type M 1: type N
14	RW	**Type selection of PWM C port** 0: type M 1: type N
13	RW	**Type selection of PWM B port** 0: type M 1: type N
12	RW	**Type selection of PWM A port** 0: type M 1: type N
11	RW	**Enable PWM D port** 0: disable 1: enable
10	RW	**Enable PWM C port** 0: disable 1: enable
9	RW	**Enable PWM B port** 0: disable 1: enable
8	RW	**Enable PWM A port** 0: disable 1: enable
7 :1	RW	**Reserved**
0	RW	**Enable PWM & Fan Tach clock** 0: disable 1: enable

Offset: 04h		PTCR04: Clock Control Register Init = X
Bit	**R/W**	**Description**
31:24	RW	**Type N PWM period bit [7:0] (in units of type N PWM clock)**
23:20	RW	**Type N PWM clock division H bit [3:0]** 0000: divide 1 0001: divide 2 0010: divide 4 0011: divide 8 ... 1111: divide 32768

to next page

from previous page

19:16	RW	**Type N PWM clock division L bit [3:0]** 0000: divide 1 0001: divide 2 0010: divide 4 0011: divide 6 ... 1111: divide 30
15:8	RW	**Type M PWM period bit [7:0] (in units of type M PWM clock)**
7 :4	RW	**Type M PWM clock division H bit [3:0]** 0000: divide 1 0001: divide 2 0010: divide 4 0011: divide 8 ... 1111: divide 32768
3 :0	RW	**Type M PWM clock division L bit [3:0]** 0000: divide 1 0001: divide 2 0010: divide 4 0011: divide 6 ... 1111: divide 30

Offset: 08h — PTCR08: Duty Control 0 Register — Init = X

Bit	R/W	Description
31:24	RW	**PWM B falling point bit [7:0] of period**
23:16	RW	**PWM B rising point bit [7:0] of period**
15:8	RW	**PWM A falling point bit [7:0] of period**
7 :0	RW	**PWM A rising point bit [7:0] of period**

Offset: 0Ch — PTCR0C: Duty Control 1 Register — Init = X

Bit	R/W	Description
31:24	RW	**PWM D falling point bit [7:0] of period**
23:16	RW	**PWM D rising point bit [7:0] of period**
15:8	RW	**PWM C falling point bit [7:0] of period**
7 :0	RW	**PWM C rising point bit [7:0] of period**

Offset: 10h — PTCR10: Type M Control 0 Register — Init = X

Bit	R/W	Description
31:16	RW	**Type M fan tach period bit [15:0] (in unit of type M PWM clock)**
15:8		**Reserved (0)**
7	RW	**Enable smart fan tach of type M** 0: disable 1: enable
6	RW	**Reserved**

to next page

from previous page

5 :4	RW	**Type M fan tach mode selection bit [1:0]** 00: falling edge 01: rising edge 10: both edges 11: reserved
3 :1	RW	**Type M fan tach clock division bit [1:0]** 000: divide 4 001: divide 16 010: divide 64 011: divide 256 ... 111: divide 65536
0	RW	**Enable fan tach of type M** 0: disable 1: enable

Offset: 14h		**PTCR14: Type M Control 1 Register** **Init = X**
Bit	**R/W**	**Description**
31:16	RW	**Type M fan tach falling point bit [15:0] of period**
15:0	RW	**Type M fan tach rising point bit [15:0] of period**

Offset: 18h		**PTCR18: Type N Control 0 Register** **Init = X**
Bit	**R/W**	**Description**
31:16	RW	**Type N fan tach period bit [15:0] (in unit of type N PWM clock)**
15:8		**Reserved (0)**
7	RW	**Enable smart fan tach of type N** 0: disable 1: enable
6	RW	**Reserved**
5 :4	RW	**Type N fan tach mode selection bit [1:0]** 00: falling edge 01: rising edge 10: both edges 11: reserved
3 :1	RW	**Type N fan tach clock division bit [1:0]** 000: div 4 001: div 16 010: div 64 011: div 256 ... 111: div 65536
0	RW	**Enable fan tach of type N** 0: disable 1: enable

Offset: 1Ch		**PTCR1C: Type N Control 1 Register** **Init = X**
Bit	**R/W**	**Description**
31:16	RW	**Type N fan tach falling point bit [15:0] of period**
15:0	RW	**Type N fan tach rising point bit [15:0] of period**

Offset: 20h	PTCR20: Tach Source Register	Init = X
Bit	**R/W**	**Description**
31:30	RW	**PWM source of fan tach #15 bit [1:0]** 00: PWM A 01: PWM B 10: PWM C 11: PWM D
29:28	RW	**PWM source of fan tach #14 bit [1:0]** 00: PWM A 01: PWM B 10: PWM C 11: PWM D
27:26	RW	**PWM source of fan tach #13 bit [1:0]** 00: PWM A 01: PWM B 10: PWM C 11: PWM D
25:24	RW	**PWM source of fan tach #12 bit [1:0]** 00: PWM A 01: PWM B 10: PWM C 11: PWM D
1 :0	RW	**PWM source of fan tach #0 bit [1:0]** 00: PWM A 01: PWM B 10: PWM C 11: PWM D

Offset: 28h	PTCR28: Trigger Register	Init = X
Bit	**R/W**	**Description**
31:16		**Reserved (0)**
15:0	RW	**Trigger to read fan tach #15 ~ fan tach #0 (0-to-1 trigger)** Bit[15]: 0-to-1 trigger fan tach #15 Bit[14]: 0-to-1 trigger fan tach #14 ... Bit[0]: 0-to-1 trigger fan tach #0

Offset: 2Ch	PTCR2C: Result Register	Init = X
Bit	**R/W**	**Description**
31	R	**Fan tach # full measurement status** 0: partial measurement 1: full measurement
30:20		**Reserved (0)**
19:0	R	**Measured fan tach # value bit [19:0]**

Offset: 30h	PTCR30: Interrupt Control Register	Init = X
Bit	**R/W**	**Description**
31:16		**Reserved (0)**

to next page

from previous page

15:0	RW	**Enable fan tach #15 ~ fan tach #0 interrupt** Bit[15]: enable fan tach #15 interrupt 0: disable 1: enable Bit[14]: enable fan tach #14 interrupt 0: disable 1: enable ... Bit[0]: enable fan tach #0 interrupt 0: disable 1: enable

Offset: 34h		PTCR34: Interrupt Status Register — Init = X
Bit	**R/W**	**Description**
31:16		**Reserved (0)**
15:0	RW	**Fan tach #15 ~ fan tach #0 interrupt status** Bit[15]: fan tach #15 interrupt status 0: no interrupt 1: interrupt pending Bit[14]: fan tach #14 interrupt status 0: no interrupt 1: interrupt pending ... Bit[0]: fan tach #0 interrupt status 0: no interrupt 1: interrupt pending

Offset: 38h		PTCR38: Type M Limit Register — Init = X
Bit	**R/W**	**Description**
31:20		**Reserved (0)**
19:0	RW	**Type M fan tach limit bit [19:0]**

Offset: 3Ch		PTCR3C: Type N Limit Register — Init = X
Bit	**R/W**	**Description**
31:20		**Reserved (0)**
19:0	RW	**Type N fan tach limit bit [19:0]**

†† **RPM = (24000000*60) / (2 * TachoValue * TachoClkDivision)**

29 Virtual UART

29.1 Overview

AST2050 / AST1100 integrates a Virtual UART module providing virtual serial communication capabilities between host CPU and ARM CPU. The virtual UART is equipped with two sets of registers compatible with the industry defector standard - 16550 UART.

One set is for host CPU; the other set is for ARM CPU. Host CPU and ARM CPU can communicate with each other like there is a physical UART link between them, but the related data transfer actually is just through pure register read/write transfers in the chip. The base address for host CPU to access UART registers through LPC bus can be programmed by ARM CPU by the extended related registers (VUART28 and VUART2C)

Base Address of Virtual UART = 0x1E78_7000
Register Address of Virtual UART = (Base Address of VUART) + Offset

The following registers can be access by host CPU through LPC bus.

VUART00 (Host): Receiving Buffer Register (Read, DLAB = 0)
VUART00 (Host): Transmit Holding Register (Write, DLAB = 0)
VUART00 (Host): Divisor Latch Low Register (Read/Write, DLAB = 1)
VUART04 (Host): Interrupt Enable Register (Read/Write, DLAB = 0)
VUART04 (Host): Divisor Latch High Register (Read/Write, DLAB = 1)
VUART08 (Host): FIFO Control Register
VUART0C (Host): Line Control Register
VUART10 (Host): Modem Control Register
VUART14 (Host): Line Status Register
VUART18 (Host): Modem Status Register
VUART1C (Host): Scratch Register

The following registers can be access by ARM CPU through APB bus.

VUART00 (Slave): Receiving Buffer Register (Read, DLAB = 0)
VUART00 (Slave): Transmit Holding Register (Write, DLAB = 0)
VUART00 (Slave): Divisor Latch Low Register (Read/Write, DLAB = 1)
VUART04 (Slave): Interrupt Enable Register (Read/Write, DLAB = 0)
VUART04 (Slave): Divisor Latch High Register (Read/Write, DLAB = 1)
VUART08 (Slave): FIFO Control Register
VUART0C (Slave): Line Control Register
VUART10 (Slave): Modem Control Register
VUART14 (Slave): Line Status Register
VUART18 (Slave): Modem Status Register
VUART1C (Slave): Scratch Register
VUART20 (Slave): General Control Register A
VUART24 (Slave): General Control Register B
VUART28 (Slave): VUART Address Register L
VUART2C (Slave): VUART Address Register H
VUART30 (Slave): General Control Register E
VUART34 (Slave): General Control Register F
VUART38 (Slave): General Control Register G
VUART3C (Slave): General Control Register H

AST2050 / AST1100 also integrates a pass-through mode of UART1 or UART2. It creats a control path from the LPC bus, through AHB/APB, to UART1 or UART2. Host can directly access UART1 or UART2 by LPC I/O cycles without any firmware help. It could be used to replace one COM port of Super I/O on host side.

The base address for host CPU to access UART1 or UART2 registers through LPC bus can be programmed by ARM CPU by the extended related registers (PUART28 and PUART2C)

Base Address of Pass-through PUART = 0x1E78_8000
Register Address of Pass-through UART = (Base Address of PUART) + Offset

The following registers can be accessed by ARM CPU through APB bus.

PUART20: General Control Register A
PUART24: General Control Register B
PUART28: PUART Address Register L
PUART2C: PUART Address Register H
PUART30: General Control Register E
PUART34: General Control Register F
PUART38: General Control Register G
PUART3C: General Control Register H

29.2 Features

- Directly connected to both APB bus and LPC Bus
- Support one Virtual UART interfaces and one Pass-through UART interface
- Separate transmit & receive FIFO buffer (16x8) to reduce CPU overhead
- Programmable base address for host CPU to access UART registers through LPC bus

29.3 VUART Registers : Base Address = 0x1E78:7000

Offset: 00h	VUART00 (Host)	Init = X
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
		RBR: Receiving Buffer Register (DLAB = 0)
7 :0	R	**Receiving Buffer Register** The RBR contains the data byte received on the serial input port (sin). The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (UARTLSR) is set. When the FIFOs are programmed ON, this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.
		THR: Transmit Holding Register (DLAB = 0)
7:0	W	**Transmit Holding Register** The THR contains data to be transmitted on the serial output port (sout). Data can be written to the THR any time that the THR Empty (THRE) bit of the Line Status Register (UARTLSR) is set. If FIFOs are enabled and THRE is set, 16 number of characters of data may be written to the THR before the FIFO is full. Any attempt to write data when the FIFO is full results in the write data being lost.
		DLL: Divisor Latch Low Register (DLAB = 1)

to next page

from previous page

7 :0	RW	**Divisor Latch Low Register** This DLL register is designed for software compatible. The actual output baud rate is equal to LPC clock frequency, 33MHz.

Offset: 00h	VUART00 (Slave)	Init = X
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
		RBR: Receiving Buffer Register (DLAB = 0)
7 :0	R	**Receiving Buffer Register** The RBR contains the data byte received on the serial input port (sin). The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (UARTLSR) is set. When the FIFOs are programmed ON, this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.
		THR: Transmit Holding Register (DLAB = 0)
7:0	W	**Transmit Holding Register** The THR contains data to be transmitted on the serial output port (sout). Data can be written to the THR any time that the THR Empty (THRE) bit of the Line Status Register (UARTLSR) is set. If FIFOs are enabled and THRE is set, 16 number of characters of data may be written to the THR before the FIFO is full. Any attempt to write data when the FIFO is full results in the write data being lost.
		DLL: Divisor Latch Low Register (DLAB = 1)
7 :0	RW	**Divisor Latch Low Register** This DLL register is designed for software compatible. The actual output baud rate is equal to LPC clock frequency, 33MHz.

Offset: 04h	VUART04 (Host)	Init = 0
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
		IER: Interrupt Enable Register (DLAB = 0)
7	RW	**Enable FIFO 1/2 full THRE Interrupt Mode if VUART34[6]=1** 0: Disable FIFO 1/2 full THRE interrupt mode 1: Enable FIFO 1/2 full THRE interrupt mode if VUART34[6]=1
6 :4		**Reserved (0)**
3	RW	**Enable Modem Status Interrupt** 0: Disable interrupt 1: Enable interrupt
2	RW	**Enable Receiver Line Status Interrupt** 0: Disable interrupt 1: Enable interrupt
1	RW	**Enable Transmitter Holding Register Empty Interrupt** 0: Disable interrupt 1: Enable interrupt
0	RW	**Enable Received Data Available Interrupt** 0: Disable interrupt 1: Enable interrupt

to next page

from previous page

DLH: Divisor Latch High Register (DLAB = 1)		
7 :0	RW	**Divisor latch (High)** This DLH register is designed for software compatible. The actual output baud rate is equal to LPC clock frequency, 33MHz.

Offset: 04h	VUART04 (Slave)	Init = 0
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
IER: Interrupt Enable Register (DLAB = 0)		
7 :4		**Reserved (0)**
3	RW	**Enable Modem Status Interrupt** 0: Disable interrupt 1: Enable interrupt
2	RW	**Enable Receiver Line Status Interrupt** 0: Disable interrupt 1: Enable interrupt
1	RW	**Enable Transmitter Holding Register Empty Interrupt** 0: Disable interrupt 1: Enable interrupt
0	RW	**Enable Received Data Available Interrupt** 0: Disable interrupt 1: Enable interrupt
DLH: Divisor Latch High Register (DLAB = 1)		
7 :0	RW	**Divisor latch (High)** This DLH register is designed for software compatible. The actual output baud rate is equal to LPC clock frequency, 33MHz.

Offset: 08h	VUART08 (Host): (IIR) Interrupt Identity Register	Init = 0xC1
Bit	**R/W**	**Description**
31:4		**Reserved (0)**
3:1	R	**Interrupt Decoding Table** The content of this register can be used to identify the source of the current interrupt based on the following: 000: Modem Status Changed 001: THR empty 010: Received Data Available 011: Receiver Status 110: Character Time Out For more information about Interrupt Identity, see the following Table for detailed description.
0	R	Indicates that an interrupt is pending when its logic 0. When its 1, no interrupt is pending.
Note : The IIR enables the programmer to retrieve what is the current highest priority pending interrupt.		

VUART Interrupt Type Decoding				
Bit [3:1]	Priority	Interrupt Type	Interrupt Source	Interrupt Clear Control
011	Highest	Receiver line status	Overrun or Break Interrupt.	Reading the Line Status Register. IER[2](Host)=0
010	2nd	Received Data available	FIFO ON: RX FIFO trigger level reached	FIFO ON: FIFO drops below the trigger level IER[0](Host)=0 FCR[1](Host)=1
110	3rd	Character Timeout indication	Theres at least 1 character in the FIFO but no character has been input to the FIFO or read from it for 1/512, 1/256, 1/128 or 1/64 second.	Reading the RBR. IER[0](Host)=0 FCR[1](Host)=1 VUART34[1](Slave)=1
001	4th	Transmitter Holding register empty	Transmitter Holding register empty	Reading the IIR or writing into THR
000	5th	Modem status	nCTS (Clear to send), nDSR (data set ready)	Reading the MSR.

Offset: 08h	VUART08 (Host): (FCR) FIFO Control Register	Init = 0x01
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7:6	W	**Define the Receiver FIFO Interrupt trigger level.** 00: 1 byte received 01: 4 bytes received 10: 8 bytes received 11: 14 bytes received
5:3		**Reserved (0)**
2	W	**Transmit FIFO Reset** Writing 1 to this bit clears the Transmitter FIFO and resets its logic.
1	W	**Receive FIFO Reset** Writing 1 to this bit clears the Receiver FIFO and resets its logic.
0	W	**Enable UART FIFO** 0: Disable FIFO 1: Enable FIFO The value of this register is always logic '1'.

Note :
The FCR allows selection of the FIFO trigger level (the number of bytes in FIFO required to enable the Received Data Available interrupt). In addition, the FIFOs can be cleared using this register.

Offset: 08h	VUART08 (Slave): (IIR) Interrupt Identity Register	Init = 0xC1
Bit	**R/W**	**Description**
31:4		**Reserved (0)**

to next page

from previous page

3:1	R	**Interrupt Decoding Table** The content of this register can be used to identify the source of the current interrupt based on the following: 000: Modem Status Changed 001: THR empty 010: Received Data Available 011: Receiver Status 110: Character Time Out For more information about Interrupt Identity, see the following Table for detailed description.
0	R	Indicates that an interrupt is pending when its logic 0. When its 1, no interrupt is pending.
Note : The IIR enables the programmer to retrieve what is the current highest priority pending interrupt.		

VUART Interrupt Type Decoding

Bit [3:1]	Priority	Interrupt Type	Interrupt Source	Interrupt Clear Control
011	Highest	Receiver line status	Overrun or Break Interrupt.	Reading the Line Status Register. IER[2](Slave)=0 FCR[1](Slave)=1 if Overrun
010	2nd	Received Data available	FIFO ON: RX FIFO trigger level reached	FIFO ON: FIFO drops below the trigger level IER[0](Slave)=0 FCR[1](Slave)=1
110	3rd	Character Timeout indication	Theres at least 1 character in the FIFO but no character has been input to the FIFO or read from it for 1/512, 1/256, 1/128 or 1/64 second.	Reading the RBR. IER[0](Slave)=0 FCR[1](Slave)=1 VUART34[0](Slave)=1
001	4th	Transmitter Holding register empty	Transmitter Holding register empty	Reading the IIR or writing into THR FCR[2](Slave)=1
000	5th	Modem status	nCTS (Clear to send), nDSR (data set ready)	Reading the MSR.

Offset: 08h — VUART08 (Slave): (FCR) FIFO Control Register — Init = 0x01

Bit	R/W	Description
31:8		**Reserved (0)**
7:6	W	**Define the Receiver FIFO Interrupt trigger level.** 00: 1 byte received 01: 4 bytes received 10: 8 bytes received 11: 14 bytes received
5:3		**Reserved (0)**
2	W	**Transmit FIFO Reset** Writing 1 to this bit clears the Transmitter FIFO and resets its logic.

to next page

from previous page

1	W	**Receive FIFO Reset** Writing 1 to this bit clears the Receiver FIFO and resets its logic.
0	W	**Enable UART FIFO** 0: Disable FIFO 1: Enable FIFO The value of this register is always logic '1'.
Note : The FCR allows selection of the FIFO trigger level (the number of bytes in FIFO required to enable the Received Data Available interrupt). In addition, the FIFOs can be cleared using this register.		

Offset: 0Ch		VUART0C (Host): (LCR) Line Control Register Init = 0x03
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	RW	**DLAB: Divisor latch access bit** 0: The normal registers are accessed. 1: The divisor latches can be accessed. Setting this bit will enable the reading and writing of the Divisor Latch register (DLL and DLH) to set the baud rate of the UART. This bit must be cleared after initial baud rate setup in order to access other registers.
6	RW	**Break Control bit.** 0: break is disabled. 1: break event is transmitted to the Slave side.
5:2	RW	**Reserved**
1:0	RW	**Select number of bits per character** 00: 5 bits. 01: 6 bits. 10: 7 bits. 11: 8 bits.

Offset: 0Ch		VUART0C (Slave): (LCR) Line Control Register Init = 0x03
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	RW	**DLAB: Divisor latch access bit** 0: The normal registers are accessed. 1: The divisor latches can be accessed. Setting this bit will enable the reading and writing of the Divisor Latch register (DLL and DLH) to set the baud rate of the UART. This bit must be cleared after initial baud rate setup in order to access other registers.
6	RW	**Break Control bit.** 0: break is disabled. 1: break event is transmitted to the Host side.
5:2	RW	**Reserved**
1:0	RW	**Select number of bits per character** 00: 5 bits. 01: 6 bits. 10: 7 bits. 11: 8 bits.

Offset: 10h	VUART10 (Host): (MCR) Modem Control Register	Init = 0
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	R	**Transmit FIFO full.** 0: transmit FIFO not full. 1: transmit FIFO full.
6:5		**Reserved (0)**
4	RW	**Loopback mode.** 0: normal operation. 1: loopback mode. The signal of the transmitter shift register is internally connected to the input of the receiver shift register. The following connections are made: nDTR → nDSR nRTS → nCTS Out1 → nRI Out2 → nDCD
3	RW	**Out2.** In loopback mode, connected to Data Carrier Detect(nDCD) input.
2	RW	**Out1.** In loopback mode, connected to Ring Indicator (nRI)signal input.
1	RW	**Request To Send (nRTS) signal control.** 0: nRTS is '1' 1: nRTS is '0'
0	RW	**Data Terminal Ready (nDTR) signal control.** 0: nDTR is '1' 1: nDTR is '0'

Offset: 10h	VUART10 (Slave): (MCR) Modem Control Register	Init = 0
Bit	**R/W**	**Description**
31:5		**Reserved (0)**
4	RW	**Loopback mode.** 0: normal operation. 1: loopback mode. The signal of the transmitter shift register is internally connected to the input of the receiver shift register. The following connections are made: nDTR → nDSR nRTS → nCTS Out1 → nRI Out2 → nDCD
3	RW	**Out2.** In loopback mode, connected to Data Carrier Detect(nDCD) input.
2	RW	**Out1.** In loopback mode, connected to Ring Indicator (nRI)signal input.
1	RW	**Request To Send (nRTS) signal control.** 0: nRTS is '1' 1: nRTS is '0'
0	RW	**Data Terminal Ready (nDTR) signal control.** 0: nDTR is '1' 1: nDTR is '0'

Offset: 14h	VUART14 (Host): (LSR) Line Status Register	Init = 0x60
Bit	**R/W**	**Description**
31:7		**Reserved (0)**
6	R	**Transmitter empty** 1: The THR or FIFO and the Transmitter Shift Register are both empty. 0: Otherwise.
5	R	**Transmitter holding register empty** 1: The THR or FIFO and the Transmitter Shift Register are both empty. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. 0: Otherwise.
4	R	**Break interrupt** 1: The Break Control bit of Line Control Register on the opposite side is set. 0: No break condition in the current character.
3:2		**Reserved (0)**
1	R	**Overrun error** 1: In the FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver. 0: No overrun state.
0	R	**Data ready** 1: The receiver contains at least one character in the RBR or the receiver FIFO. 0: The RBR is read or the receiver FIFO is empty.

Offset: 14h	VUART14 (Slave): (LSR) Line Status Register	Init = 0x60
Bit	**R/W**	**Description**
31:7		**Reserved (0)**
6	R	**Transmitter empty** 1: The THR or FIFO and the Transmitter Shift Register are both empty. 0: Otherwise.
5	R	**Transmitter holding register empty** 1: The THR or FIFO and the Transmitter Shift Register are both empty. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. 0: Otherwise.
4	R	**Break interrupt** 1: The Break Control bit of Line Control Register on the opposite side is set. 0: No break condition in the current character.
3:2		**Reserved (0)**
1	R	**Overrun error** 1: In the FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver. 0: No overrun state.
0	R	**Data ready** 1: The receiver contains at least one character in the RBR or the receiver FIFO. 0: The RBR is read or the receiver FIFO is empty.

Offset: 18h	VUART18 (Host): (MSR) Modem Status Register	Init = X
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	R	**Complement of the nDSR input or equals to Out2(MCR[3]) in loopback mode.**
6	R	**Out1(MCR[2]) in loopback mode.**

to next page

from previous page

5	R	**Complement of the nDSR input or equals to nDTR(MCR[0]) in loopback mode.**
4	R	**Complement of the nCTS input or equals to nRTS(MCR[1]) in loopback mode.**
3	R	**Delta Data Carrier Detect (DDCD) indicator** 1: The nDSR line has changed its state since the last time the CPU read the MSR. In loopback mode, DDCD reflects changes on MCR bit 3 (Out2)
2	R	**Trailing Edge of Ring Indicator (TERI) detector.** In loopback mode, TERI reflects when MCR bit 2 (Out1) has changed state from a high to a low.
1	R	**Delta Data Set Ready (DDSR) indicator** 1: The nDSR line has changed its state since the last time the CPU read the MSR. In loopback mode, DDSR reflects changes on MCR bit 0 (DTR)
0	R	**Delta Clear To Send (DCTS) indicator** 1: The nCTS line has changed its state since the last time the CPU read the MSR. In loopback mode, DCTS reflects changes on MCR bit 1 (RTS)

Offset: 18h — VUART18 (Slave): (MSR) Modem Status Register — Init = X

Bit	R/W	Description
31:8		**Reserved (0)**
7	R	**Complement of the nDSR input or equals to Out2(MCR[3]) in loopback mode.**
6	R	**Out1(MCR[2]) in loopback mode.**
5	R	**Complement of the nDSR input or equals to nDTR(MCR[0]) in loopback mode.**
4	R	**Complement of the nCTS input or equals to nRTS(MCR[1]) in loopback mode.**
3	R	**Delta Data Carrier Detect (DDCD) indicator** 1: The nDSR line has changed its state since the last time the CPU read the MSR.
2	R	**Reserved (0)**
1	R	**Delta Data Set Ready (DDSR) indicator** 1: The nDSR line has changed its state since the last time the CPU read the MSR.
0	R	**Delta Clear To Send (DCTS) indicator** 1: The nCTS line has changed its state since the last time the CPU read the MSR.

Offset: 1Ch — VUART1C (Host): (SCR) Scratch Register — Init = X

Bit	R/W	Description
31:8		**Reserved (0)**
7 :0	RW	**Scratch bits** This register can be used as a temporary storage, no specific definition.

Offset: 1Ch — VUART1C (Slave): (SCR) Scratch Register — Init = X

Bit	R/W	Description
31:8		**Reserved (0)**
7 :0	RW	**Scratch bits** This register can be used as a temporary storage, no specific definition.

Offset: 20h — VUART20 (Slave): General Control Register A — Init = 0b00x0_xx00

Bit	R/W	Description
31:8		**Reserved (0)**

to next page

from previous page

7 :6	R	**Status of host-side Receiver FIFO Trigger**
5	RW	**Disable Host-Tx-discard mode** 0: Enable Host-Tx-discard mode Slave will throw data away automatically when Host Tx FIFO is not empty. 1: Disable Host-Tx-discard mode
4	R	**Status of host-side loopback mode**
3 :2	RW	**Slave-side timeout time width selection bit [1:0]** 00: 1/64 second if PUART34[3]=0 and SCU2C[13]=0 01: 1/128 second if PUART34[3]=0 and SCU2C[13]=0 10: 1/256 second if PUART34[3]=0 and SCU2C[13]=0 11: 1/512 second if PUART34[3]=0 and SCU2C[13]=0 00: 1/3600 second if PUART34[3]=0 and SCU2C[13]=1 01: 1/7200 second if PUART34[3]=0 and SCU2C[13]=1 10: 1/14400 second if PUART34[3]=0 and SCU2C[13]=1 11: 1/28800 second if PUART34[3]=0 and SCU2C[13]=1 00: 512*LCLK if PUART34[3]=1 01: 256*LCLK if PUART34[3]=1 10: 128*LCLK if PUART34[3]=1 11: 64*LCLK if PUART34[3]=1
1	RW	**SIRQ polarity** 0: The output is high impedance when host interrupt has been cleared. The output is low level when host interrupt has been set. 1: The output is low level when host interrupt has been cleared. The output is high impedance when host interrupt has been set.
0	RW	**Enable virtual UART** 0: Disable 1: Enable
Note : This register is defined for ARM CPU only.		

Offset: 24h		**VUART24 (Slave): General Control Register B** — **Init = 0bxxxx_xx11**
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7 :4	RW	**SIRQ number selection bit [3:0]** 0000: IRQ0 0001: IRQ1 0010: SMI ... 1111: IRQ15
3 :2	RW	**Host-side timeout period selection** 00: 1/64 second if PUART34[3]=0 and SCU2C[13]=0 01: 1/128 second if PUART34[3]=0 and SCU2C[13]=0 10: 1/256 second if PUART34[3]=0 and SCU2C[13]=0 11: 1/512 second if PUART34[3]=0 and SCU2C[13]=0 00: 1/3600 second if PUART34[3]=0 and SCU2C[13]=1 01: 1/7200 second if PUART34[3]=0 and SCU2C[13]=1 10: 1/14400 second if PUART34[3]=0 and SCU2C[13]=1 11: 1/28800 second if PUART34[3]=0 and SCU2C[13]=1 00: 512*PCLK if PUART34[3]=1 01: 256*PCLK if PUART34[3]=1 10: 128*PCLK if PUART34[3]=1 11: 64*PCLK if PUART34[3]=1

to next page

from previous page

1 :0	R	**Number of bits per character (host-side)**
Note : This register is defined for ARM CPU only.		

Offset: 28h	VUART28 (Slave): Virtual UART Address Register L	Init = X
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7 :0	RW	**Virtual UART address bit [7:0]** This register defines the base address (the low bytes) for host CPU to access virtual UART registers through LPC bus.
Note : This register is defined for ARM CPU only.		

Offset: 2Ch	VUART2C (Slave): Virtual UART Address Register H	Init = X
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7 :0	RW	**Virtual UART address bit [15:8]** This register defines the base address (the high bytes) for host CPU to access virtual UART registers through LPC bus.
Note : This register is defined for ARM CPU only.		

Offset: 30h	VUART30 (Slave): General Control Register E	Init = 0b0000_1110
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	R	**Transmit FIFO full. (slave-side)** 0: transmit FIFO not full. 1: transmit FIFO full.
6 :4	R	**THR read pointer bit [2:0] (slave-side)**
3 :0	R	**Complement of IIR status bit [3:0] (host-side)**
Note : This register is defined for ARM CPU only.		

Offset: 34h	VUART34 (Slave): General Control Register F	Init = X
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	RW	**Enable FIFO 1/2 full THRE Interrupt Mode (slave-side)** 0: Disable FIFO 1/2 full THRE interrupt mode 1: Enable FIFO 1/2 full THRE interrupt mode
6	RW	**Enable FIFO 1/2 full THRE Interrupt Mode Control (host-side)** 0: Disable FIFO 1/2 full THRE interrupt mode control 1: Enable FIFO 1/2 full THRE interrupt mode control
5	RW	**Trig a THRE interrupt on host side even through it has been empty already**
4 :2	RW	**Reserved**
1	RW	**Disable character time out interrupt (slave-side)**

to next page

from previous page

0	RW	**Disable character time out interrupt (host-side)**
Note : This register is defined for ARM CPU only.		

Offset: 38h	VUART38 (Slave): General Control Register G	Init = X
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7 :0	R	**THR read back data bit [7:0] (slave-side)**
Note : This register is defined for ARM CPU only.		

Offset: 3Ch	VUART3C (Slave): General Control Register H	Init = X
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7 :6	R	**Read back data bit [1:0] of receiver FIFO interrupt trigger level (slave-side)**
5 :0	R	**Reserved (0)**
Note : This register is defined for ARM CPU only.		

29.4 PUART Registers : Base Address = 0x1E78:8000

Offset: 20h	PUART20: General Control Register A	Init = 0b00x0_xx00
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7 :6	R	**Status of host-side Receiver FIFO Trigger**
5		**Reserved (0)**
4	R	**Status of host-side loopback mode**
3 :2		**Reserved (0)**
1	RW	**SIRQ polarity** 0: The output is high impedance when host interrupt has been cleared. The output is low level when host interrupt has been set. 1: The output is low level when host interrupt has been cleared. The output is high impedance when host interrupt has been set.
0	RW	**Enable pass-through UART** 0: Disable 1: Enable
Note : This register is defined for ARM CPU only.		

Offset: 24h	PUART24: General Control Register B	Init = 0bxxxx_xx11
Bit	**R/W**	**Description**
31:8		**Reserved (0)**

to next page

from previous page

7 :4	RW	**SIRQ number selection bit [3:0]** 0000: IRQ0 0001: IRQ1 0010: SMI ... 1111: IRQ15
3 :2		**Reserved (0)**
1 :0	R	**Number of bits per character (host-side)**
Note : This register is defined for ARM CPU only.		

Offset: 28h	PUART28: Pass-through UART Address Register L	Init = X
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7 :0	RW	**Pass-through UART address bit [7:0]** This register defines the base address (the low bytes) for host CPU to access pass-through UART registers through LPC bus.
Note : This register is defined for ARM CPU only.		

Offset: 2Ch	PUART2C: Pass-through UART Address Register H	Init = X
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7 :0	RW	**Pass-through UART address bit [15:8]** This register defines the base address (the high bytes) for host CPU to access pass-through UART registers through LPC bus.
Note : This register is defined for ARM CPU only.		

Offset: 30h	PUART30: General Control Register E	Init = 0
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	R	**Complement of Enable Transmitter Holding Register Empty Interrupt (host-side)**
6	R	**Complement of Enable Received Data Available Interrupt (host-side)**
5 :0	R	**Complement of LCR bit [7:2] (host-side)**
Note : This register is defined for ARM CPU only.		

Offset: 34h	PUART34: General Control Register F	Init = X
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	RW	**Pass-through mode selection** 0: Pass-through mode for UART1 1: Pass-through mode for UART2

to next page

from previous page

6 :4		**Reserved**
3	RW	**VUART timeout time width control bit (slave-side)**
2	RW	**VUART timeout time width control bit (host-side)**
1 :0		**Reserved**
Note : This register is defined for ARM CPU only.		

Offset: 38h		**PUART38: General Control Register G** — **Init = X**
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7 :0		**Reserved**
Note : This register is defined for ARM CPU only.		

Offset: 3Ch		**PUART3C: General Control Register H** — **Init = 0x00**
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7	R	**Complement of Enable Modem Status Interrupt (host-side)**
6	R	**Complement of Enable Receiver Line Status Interrupt (host-side)**
5 :0		**Reserved (0)**
Note : This register is defined for ARM CPU only.		

30 LPC Controller

30.1 Overview

AST2050 / AST1100 integrates both LPC Host Controller and LPC Slave Controller, but only one of the two controllers can be enabled at one time. LPC Slave Controller also integrates IPMI 2.0/1.1 compliant BMC controller. There are totally 49 registers, which is listed below, to control the various functions supported by AST2050 / AST1100 . Each register has its own specific offset value to derive its physical address location.

Base Address of LPC Controller = 0x1E78_9000
Physical address of register = (Base address of LPC Controller) + Offset

HICR0:	Host Interface Control Register 0
HICR1:	Host Interface Control Register 1
HICR2:	Host Interface Control Register 2
HICR3:	Host Interface Control Register 3
HICR4:	Host Interface Control Register 4
LADR3H:	LPC Channel #3 Address register H
LADR3L:	LPC Channel #3 Address Register L
LADR12H:	LPC Channel #1/#2 Address Register H
LADR12L:	LPC Channel #1/#2 Address Register L
IDR1:	Input Data Register 1
IDR2:	Input Data Register 2
IDR3:	Input Data Register 3
ODR1:	Output Data Register 1
ODR2:	Output Data Register 2
ODR3:	Output Data Register 3
STR1:	Status Register 1
STR2:	Status Register 2
STR3:	Status Register 3
BTR0:	BT Status Register 0
BTR1:	BT Status Register 1
BTCSR0:	BT Control Status Register 0
BTCSR1:	BT Control Status Register 1
BTCR:	BT Control Register
BTDTR:	BT Data Buffer
BTIMSR:	BT Interrupt Mask Register
BTFVSR0:	BT FIFO Valid Size Register 0
BTFVSR1:	BT FIFO Valid Size Register 1
SIRQCR0:	SERIRQ Control Register 0
SIRQCR1:	SERIRQ Control Register 1
SIRQCR2:	SERIRQ Control Register 2
SIRQCR3:	SERIRQ Control Register 3
HICR5:	Host Interface Control Register 5
HICR6:	Host Interface Control Register 6
HICR7:	Host Interface Control Register 7
HICR8:	Host Interface Control Register 8
SNPWADR:	LPC Snoop Address Register
SNPWDR:	LPC Snoop Data Register
LHCR0:	LPC Host Control Register 0
LHCR1:	LPC Host Control Register 1
LHCR2:	LPC Host Control Register 2
LHCR3:	LPC Host Control Register 3
LHCR4:	LPC Host Control Register 4
LHCR5:	LPC Host Control Register 5

LHCR6:	LPC Host Control Register 6
LHCR7:	LPC Host Control Register 7
LHCR8:	LPC Host Control Register 8
LHCR9:	LPC Host Control Register 9
LHCRA:	LPC Host Control Register A
LHCRB:	LPC Host Control Register B

The definition of BMC related registers, from offset 0x00 to offset 0x7C, are basically compatible with the popular BMC controller - H8S/2168. Therefore, the software code developed for the chip can be easily ported to AST2050 / AST1100 .

30.2 Features

- Directly connected to APB bus interface
- Dual operation modes
  - Master mode: designed to update system BIOS, TPM or LPC keyboard controller (I/O, memory, firmware read write cycles)
  - Salve mode: designed for BMC functions (I/O read write cycles)
- Support Serial IRQ (reduce polling time)
- Support port 80H/81H (programmable address) snooping registers with interrupt option
- Support two set of Virtual UART (16550) (SIRQ#)
- Compliant with IPMI version 2.0 KCS mode and BT mode
  - Channel #1 supports KCS interface
  - Channel #2 supports KCS interface
  - Channel #3 supports KCS or BT interface
- Three register sets to support three programmable I/O channels. Each register set includes:
  - Input data register (IDR1-IDR3)
  - Output data register (ODR1-ODR3)
  - Status register (STR1-STR3)
- H8S/2168 compliant register definition and programming sequence
- Two sets of 64x8 Embedded SRAM for BT mode support
- Support LPC S/W & H/W power down function
- Support LPC Abort monitoring function

30.3 Registers : Base Address = 0x1E78:9000

Attribute Definition:

Attribute	:	Description
R	:	Readable
W	:	Writable
W0C	:	Write '0' to clear value to 0
W1C	:	Write '1' to clear value to 0
W1T	:	Write '1' to toggle value
W0S	:	Write '0' to set value to 1
W1S	:	Write '1' to set value to 1
U	:	Unknown value

HICR0: Host Interface Control Register 0 — **Offset: 00h**

Bit	Name	Initial	Slave	Host	Description
7	LPC3E	0	RW	-	Enable LPC Channel #3
6	LPC2E	0	RW	-	Enable LPC Channel #2
5	LPC1E	0	RW	-	Enable LPC Channel #1
4		0	R	-	Reserved
3	SDWNE	0	RW	-	Enable LPC software shutdown
2	PMEE	0	RW	-	Enable PME output
1		0	R	-	Reserved
0		0	R	-	Reserved

HICR1: Host Interface Control Register 1 — **Offset: 04h**

Bit	Name	Initial	Slave	Host	Description
7	LPCBSY	0	R	-	LPC busy flag
6	CLKREQ	0	R	-	LCLK request
5	IRQBSY	0	R	-	SERIRQ busy flag
4	LRSTB	0	RW	-	LPC software reset bit
3	SDWNB	0	RW	-	LPC software shutdown bit
2	PMEB	0	RW	-	PME output bit
1		0	R	-	Reserved
0		0	R	-	Reserved

HICR2: Host Interface Control Register 2 — **Offset: 08h**

Bit	Name	Initial	Slave	Host	Description
7		0	R	-	Reserved
6	LRST	0	RW0C	-	LPC reset interrupt status
5	SDWN	0	RW0C	-	LPC shutdown interrupt status
4	ABRT	0	RW0C	-	LPC Abort Interrupt status
3	IBFIF3	0	RW	-	Enable IBFI3 interrupt
2	IBFIF2	0	RW	-	Enable IDR2 receive completion interrupt
1	IBFIE1	0	RW	-	Enable IDR1 receive completion interrupt
0	ERRIE	0	RW	-	Enable error interrupt

HICR3: Host Interface Control Register 3 — **Offset: 0Ch**

Bit	Name	Initial	Slave	Host	Description
7	LFRAME	U	R	-	LFRAME pin monitoring
6	CLKRUN	U	R	-	CLKRUN pin monitoring
5	SERIRQ	U	R	-	SERIRQ pin monitoring
4	LRESET	U	R	-	LRESET pin monitoring
3	LPCPD	U	R	-	LPCPD pin monitoring
2	PME	U	R	-	PME pin monitoring
1		0	R	-	Reserved
0		0	R	-	Reserved

HICR4: Host Interface Control Register 4 — **Offset: 10h**

Bit	Name	Initial	Slave	Host	Description
7	LADR12AS	U	RW	-	Channel address selection (LADR12H or LADRL)
6		U	R	-	Reserved
5		U	R	-	Reserved
4		U	R	-	Reserved
3		U	R	-	Reserved
2	KCSENBL	U	RW	-	Enable KCS interface in Channel #3
1		0	R	-	Reserved
0	BTENBL	0	RW	-	Enable BT interface in Channel #3

LADR3H: LPC Channel #3 Address register H — **Offset: 14h**

Bit	Name	Initial	Slave	Host	Description
7	Bit15	0	RW	-	Channel #3 address Bit[15]
6	Bit14	0	RW	-	Channel #3 address Bit[14]
5	Bit13	0	RW	-	Channel #3 address Bit[13]
4	Bit12	0	RW	-	Channel #3 address Bit[12]
3	Bit11	0	RW	-	Channel #3 address Bit[11]
2	Bit10	0	RW	-	Channel #3 address Bit[10]
1	Bit9	0	RW	-	Channel #3 address Bit[9]
0	Bit8	0	RW	-	Channel #3 address Bit[8]

LADR3L: LPC Channel #3 Address Register L — **Offset: 18h**

Bit	Name	Initial	Slave	Host	Description
7	Bit7	0	RW	-	Channel #3 address Bit[7]
6	Bit6	0	RW	-	Channel #3 address Bit[6]
5	Bit5	0	RW	-	Channel #3 address Bit[5]
4	Bit4	0	RW	-	Channel #3 address Bit[4]
3	Bit3	0	RW	-	Channel #3 address Bit[3]
2		0	R	-	Reserved
1	Bit1	0	RW	-	Channel #3 address Bit[1]
0		0	R	-	Reserved

LADR12H: LPC Channel #1/#2 Address Register H					Offset: 1Ch
Bit	Name	Initial	Slave	Host	Description
7:0	**Bit[15:8]**	0	RW	-	**Channel #1/#2 address Bit[15:8]** (Selected by LADRSEL)

LADR12L: LPC Channel #1/#2 Address Register L					Offset: 20h
Bit	Name	Initial	Slave	Host	Description
7:0	**Bit[7:0]**	60h /62h	RW	-	**Channel #1/#2 address bit[7:0]** (Selected by LADRSEL)

IDR1: Input Data Register 1					Offset: 24h
Bit	Name	Initial	Slave	Host	Description
7:0	**Bit[7:0]**	U	R	W	**Channel #1 input data Bit[7:0]**

IDR2: Input Data Register 2					Offset: 28h
Bit	Name	Initial	Slave	Host	Description
7:0	**Bit[7:0]**	U	R	W	**Channel #2 input data Bit[7:0]**

IDR3: Input Data Register 3					Offset: 2Ch
Bit	Name	Initial	Slave	Host	Description
7:0	**Bit[7:0]**	U	R	W	**Channel #3 input data Bit[7:0]**

ODR1: Output Data Register 1					Offset: 30h
Bit	Name	Initial	Slave	Host	Description
7:0	**Bit[7:0]**	U	RW	R	**Channel #1 output data Bit[7:0]**

ODR2: Output Data Register 1					Offset: 34h
Bit	Name	Initial	Slave	Host	Description
7:0	**Bit[7:0]**	U	RW	R	**Channel #2 output data Bit[7:0]**

ODR3: Output Data Register 1					Offset: 38h
Bit	Name	Initial	Slave	Host	Description
7:0	**Bit[7:0]**	U	RW	R	**Channel #3 output data Bit[7:0]**

STR1: Status Register 1					Offset: 3Ch
Bit	Name	Initial	Slave	Host	Description
7	**DBU17**	0	RW	R	**Defined by user**
6	**DBU16**	0	RW	R	**Defined by user**
5	**DBU15**	0	RW	R	**Defined by user**
4	**DBU14**	0	RW	R	**Defined by user**
3	**C/D1**	0	R	R	**Command/Data**
2	**DBU12**	0	RW	R	**Defined by user**
1	**IBF1**	0	R	R	**Input data register full**
0	**OBF1**	0	RW0C	R	**Output data register full**

STR2: Status Register 2					Offset: 40h
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
7	**DBU27**	0	RW	R	**Defined by user**
6	**DBU26**	0	RW	R	**Defined by user**
5	**DBU25**	0	RW	R	**Defined by user**
4	**DBU24**	0	RW	R	**Defined by user**
3	**C/D2**	0	R	R	**Command/Data**
2	**DBU22**	0	RW	R	**Defined by user**
1	**IBF2**	0	R	R	**Input data register full**
0	**OBF2**	0	RW0C	R	**Output data register full**

STR3: Status Register 3					Offset: 44h
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
7	**DBU37**	0	RW	R	**Defined by user**
6	**DBU36**	0	RW	R	**Defined by user**
5	**DBU35**	0	RW	R	**Defined by user**
4	**DBU34**	0	RW	R	**Defined by user**
3	**C/D3**	0	R	R	**Command/Data**
2	**DBU22**	0	RW	R	**Defined by user**
1	**IBF3**	0	R	R	**Input data register full**
0	**OBF3**	0	RW0C	R	**Output data register full**

BTR0: BT Status Register 0					Offset: 48h
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
7		0	R	-	**Reserved**
6		0	R	-	**Reserved**
5		0	R	-	**Reserved**
4	**FRDI**	0	RW0C	-	**FIFO read request interrupt**
3	**HRDI**	0	RW0C	-	**BT host read interrupt**
2	**HWRI**	0	RW0C	-	**BT host write interrupt**
1	**HBTWI**	0	RW0C	-	**BTDTR host write start interrupt**
0	**HBTRI**	0	RW0C	-	**BTDTR host read end interrupt**

BTR1: BT Status Register 1					Offset: 4Ch
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
7		0	R	-	**Reserved**
6	**HRSTI**	0	RW0C	-	**BT reset interrupt**
5		0	R	-	**Reserved**
4	**BEVTI**	0	RW0C	-	**BEVT_ATN clear interrupt**
3	**B2HI**	0	RW0C	-	**Read-end interrupt**
2	**H2BI**	0	RW0C	-	**Write-end interrupt**
1	**CRRPI**	0	RW0C	-	**Read pointer clear interrupt**
0	**CRWPI**	0	RW0C	-	**Write pointer clear interrupt**

BTCSR0: BT Control Status Register 0					Offset: 50h
Bit	Name	Initial	Slave	Host	Description
7		0	R	-	Reserved
6	FSEL1	0	RW	-	BT Transfer FIFO selection bit 1
5	FSEL0	0	RW	-	T Transfer FIFO selection bit 0
4	FRDIE	0	RW	-	Enable FIFO read request interrupt
3	HRDIE	0	RW	-	Enable BT host read interrupt
2	HWRIE	0	RW	-	Enable BT host write interrupt
1	HBTWIE	0	RW	-	Enable BTDTR host write start interrupt
0	HBTRIE	0	RW	-	Enable BTDTR host read end interrupt

BTCSR1: BT Control Status Register 1					Offset: 54h
Bit	Name	Initial	Slave	Host	Description
7	RSTREN	0	RW	-	Enable slave reset read
6	HRSTIE	0	RW	-	Enable BT reset interrupt
5		0	R	-	Reserved
4	BEVTIE	0	RW	-	Enable BEVT_ATN clear interrupt
3	B2HIE	0	RW	-	Enable read end interrupt
2	H2BIE	0	RW	-	Enable write end interrupt
1	CRRPIE	0	RW	-	Enable read pointer clear interrupt
0	CRWPIE	0	RW	-	Enable write pointer clear interrupt

BTCR: BT Control Register					Offset: 58h
Bit	Name	Initial	Slave	Host	Description
7	B_BUSY	0	RW	R	BT write transfer busy flag
6	H_BUSY	0	R	W1T	BT read transfer busy flag
5	OEM0	0	RW	RW0S	User defined bit
4	BEVT_ATN	0	RW1S	RW1C	Event interrupt
3	B2H_ATN	0	RW1S	RW1C	Slave buffer write end indication flag
2	H2B_ATN	0	RW0C	RW1S	Host buffer write end indication flag
1	CLR_RD_PTR	0	RW0C	W1S	Read pointer clear
0	CLR_WR_PTR	0	RW0C	W1S	Write pointer clear

BTDTR: BT Data Buffer					Offset: 5Ch
Bit	Name	Initial	Slave	Host	Description
7:0	Bit [7:0]	U	RW	RW	BT mode buffer read/write data

BTIMSR: BT Interrupt Mask Register					Offset: 60h
Bit	Name	Initial	Slave	Host	Description
7	BMC_HWRST	0	RW0C	RW1S	Slave reset
6		0	R	R	Reserved
5		0	R	R	Reserved
4	OEM3	0	RW	RW0S	User defined bit

to next page

from previous page

3	**OEM2**	0	RW	RW0S	**User defined bit**
2	**OEM1**	0	RW	RW0S	**User defined bit**
1		0	R	R	**Reserved**
0		0	R	R	**Reserved**

BTFVSR0: BT FIFO Valid Size Register 0 **Offset: 64h**

Bit	Name	Initial	Slave	Host	Description
7:0	**N7 to N0**	0	R	-	**Valid bytes in the FIFO for host write transfer**

BTFVSR1: BT FIFO Valid Size Register 1 **Offset: 68h**

Bit	Name	Initial	Slave	Host	Description
7:0	**N7 to N0**	0	R	-	**Valid bytes in the FIFO for host read transfer**

SIRQCR0: SERIRQ Control Register 0 **Offset: 70h**

Bit	Name	Initial	Slave	Host	Description
7	**Q_C**	0	R	-	**Quiet/Continuous Mode Flag**
6		0	R	-	**Reserved**
5	**IEDIR**	0	RW	-	**Interrupt Enable Direct Mode**
4		0	R	-	**Reserved**
3	**SMIE3A**	0	RW	-	**Host SMI Interrupt Enable 3A**
2	**SMIE2**	0	RW	-	**Host SMI Interrupt Enable 2**
1	**IRQ12E1**	0	RW	-	**Host IRQ12 Interrupt Enable 1**
0	**IRQ1E1**	0	RW	-	**Host IRQ1 Interrupt Enable 1**

SIRQCR1: SERIRQ Control Register 1 **Offset: 74h**

Bit	Name	Initial	Slave	Host	Description
7	**IRQ11E3**	0	RW	-	**Host IRQ11 Interrupt Enable 3**
6	**IRQ10E**	0	RW	-	**Host IRQ10 Interrupt Enable 3**
5	**IRQ9E3**	0	RW	-	**Host IRQ9 Interrupt Enable 3**
4	**IRQ6E3**	0	RW	-	**Host IRQ6 Interrupt Enable 3**
3	**IRQ11E2**	0	RW	-	**Host IRQ11 Interrupt Enable 2**
2	**IRQ10E2**	0	RW	-	**Host IRQ10 Interrupt Enable 2**
1	**IRQ9E2**	0	RW	-	**Host IRQ9 Interrupt Enable 2**
0	**IRQ6E2**	0	RW	-	**Host IRQ6 Interrupt Enable 2**

SIRQCR2: SERIRQ Control Register 2 **Offset: 78h**

Bit	Name	Initial	Slave	Host	Description
7	**IEDIR3**	0	RW	-	**Interrupt Enable Direct Mode 3**
6:0		0	R	-	**Reserved**

SIRQCR3: SERIRQ Control Register 3					Offset: 7Ch
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
7		0	R	-	**Reserved**
6	**SELIRQ11**	0	RW	-	**Select SERIRQ11 Output**
5	**SELIRQ10**	0	RW	-	**Select SERIRQ10 Output**
4	**SELIRQ9**	0	RW	-	**Select SERIRQ9 Output**
3	**SELIRQ6**	0	RW	-	**Select SERIRQ6 Output**
2	**SELSMI**	0	RW	-	**Select SERSMI Output**
1	**SELIRQ12**	0	RW	-	**Select SERIRQ12 Output**
0	**SELIRQ1**	0	RW	-	**Select SERIRQ1 Output**

HICR5: Host Interface Control Register 5					Offset: 80h
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
31:24	**HWMBASE**	0	RW	-	**LPC to AHB bridge address decoding base bit [31:24]**
23:20	**ID3IRQX**	0	RW	-	**Select ID bit[3:0] of IRQX for channel #3**
19:16	**ID2IRQX**	0	RW	-	**Select ID bit[3:0] of IRQX for channel #2**
15	**SEL3IRQX**	0	RW	-	**Select SERIRQX output for channel #3**
14	**IRQXE3**	0	RW	-	**Host IRQX interrupt enable for channel #3**
13	**SEL2IRQX**	0	RW	-	**Select SERIRQX output for channel #2**
12	**IRQXE2**	0	RW	-	**Host IRQX interrupt enable for channel #2**
11		0	R	-	**Reserved**
10	**ENFWH**	0	RW	-	**Enable LPC FWH cycles**
9	**ENINT_PME**	0	RW	-	**Enable PME# interrupt** 0: Disable 1: Enable
8	**ENL2H**	0	RW	-	**Enable LPC to AHB bridge**
7:6		0	R	-	**Reserved**
5	**ENSET_SF**	0	RW	-	**Enable the capability to issue SIRQ start frame cycles** 0: No operation 1: Enable the capability This register is designed to enable LCP Slave Controller to be able to trig chipset to issue SIRQ start frame cycles.
4	**ENLCLK_REQ**	0	RW	-	**Enable LCLK Request** 0: Disable 1: Enable This bit will enable CLOCKRUN# signaling protocol to request chipset to start or speed up LPC bus clock.
3	**ENINT_SNP1W**	0	RW	-	**Enable snooping address #1 interrupt** 0: Disable 1: Enable This bit will enable LPC Slave Controller to issue interrupt for LPC bus-write cycles (snooping address #1).

to next page

from previous page

2	**EN_SNP1W**	0	RW	-	**Enable snooping address #1** 0: Disable 1: Enable This bit will enable LPC Slave Controller to snoop LPC bus cycle regarding to the snooping address #1.
1	**ENINT_SNP0W**	0	RW	-	**Enable snooping address #0 interrupt** 0: Disable 1: Enable This bit will enable LPC Slave Controller to issue interrupt for all LPC bus-write cycles matched with the snooping address #0.
0	**EN_SNP0W**	0	RW	-	**Enable snooping address #0** 0: Disable 1: Enable This bit will enable LPC Slave Controller to snoop LPC bus cycles regarding to the snooping address #0.

HICR6: Host Interface Control Register 6 **Offset: 84h**

Bit	Name	Initial	Slave	Host	Description
31:28		0	R	-	**Reserved**
27:24	**HWNCARE**	0	RW	-	**Address decoding range control bit [27:24] of LPC to AHB bridge**
23:8		0	R	-	**Reserved**
7	**SIRQSTOP**	0	R	-	**Reserved**
6	**ST_ENPME**	0	R	-	**Reserved**
5:3		0	R	-	**Reserved**
2	**STR_PME**	0	RW1C	-	**PME# interrupt status** 0: No interrupt 1: Interrupt is pending Writing '1' to this bit will clear the status.
1	**STR_SNP1W**	0	RW1C	-	**Snooping address #1 interrupt status** 0: No interrupt 1: Interrupt is pending Writing '1' to this bit will clear the status.
0	**STR_SNP0W**	0	RW1C	-	**Snooping address #0 interrupt status** 0: No interrupt 1: Interrupt is pending Writing '1' to this bit will clear the status.

HICR7: Host Interface Control Register 7 **Offset: 88h**

Bit	Name	Initial	Slave	Host	Description
31:16	**ADRBASE**	0	RW	-	**Remapping address base bit [31:16] of LPC to AHB bridge**
15:0		0	R	-	**Reserved**

HICR8: Host Interface Control Register 8					Offset: 8Ch
Bit	Name	Initial	Slave	Host	Description
31:16	ADRMASK	0	RW	-	**Remapping address mask bit [31:16] of LPC to AHB bridge**
15:0		0	R	-	**Reserved**

SNPWADR: LPC Snoop Address Register					Offset: 90h
Bit	Name	Initial	Slave	Host	Description
31:16	Bit [15:0]	U	RW	-	**Snooping address #1** This register is designed to set the snooping address #1 (Bit [15:0]) for monitoring LPC bus I/O-write cycles. Setting snooping address like 80h or 81h is the typical applications to monitor system BIOS activities.
15:0	Bit [15:0]	U	RW	-	**Snooping address #0** This register is designed to set the snooping address #0 Bit [15:0] for monitoring LPC bus I/O-write cycles. Setting snooping address like 80h or 81h is the typical applications to monitor system BIOS activities.

SNPWDR: LPC Snoop Data Register					Offset: 94h
Bit	Name	Initial	Slave	Host	Description
15:8	Bit [7:0]	U	R	-	**Snooping address #1 data Bit[7:0]** This register will always record the last data of LPC bus write cycles with address matched with SNPWADR [31:16]. This register will be automatically updated whenever a new write cycle with matched address. Firmware code needs to take care of the potential issue of reading transition data.
7:0	Bit [7:0]	U	R	-	**Snooping address #0 data Bit[7:0]** This register will always record the last data of the LPC bus write cycles with address matched with SNPWADR [15:0]. This register will be automatically updated whenever a new write cycle with matched address. Firmware code needs to take care of the potential issue of reading transition data.

LHCR0: LPC Host Control Register 0					Offset: A0h
Bit	Name	Initial	Slave	Host	Description
31:24	XIN	0	R	-	**Reserved**
23	LRSTNO	0	RW	-	**LPC reset pin output** 0: output low 1: output high
23:16		0	RW	-	**Reserved**
15	LRSTNOEN	1	RW	-	**LPC reset pin output control** 0: output mode 1: input mode

to next page

from previous page

14:13		-	RW	-	**Reserved**
12		1	RW	-	**Disable vector interrupt output connected to host serial IRQ** It provide an option for host to use GPIO when ARM is disabled. 0: enable 1: disable
11:10		-	RW	-	**Reserved**
9		1	RW	-	**Enable PUART serial IRQ low stretcher** 0: disable 1: enable
8		1	RW	-	**Enable VUART serial IRQ low stretcher** 0: disable 1: enable
7	**SIRQLONG**	0	RW	-	**Reserved**
6:5		0	R	-	**Reserved**
4	**ENP2L**	0	RW	-	**Enable APB to LPC bridge**
3		0	R	-	**Reserved**
2	**ENLHSIRQ**	0	RW	-	**Enable LPC Host SIRQ** 0: Disable LPC host SIRQ 1: Enable LPC host SIRQ When this bit is enabled, LPC Host Controller will be able to receive SIRQ from LPC slave devices. This bit can be enabled only when LPC Host Controller is enabled.
1	**ENLHTM-OUT**	0	RW	-	**Enable LPC host time-out function** 0: Disable LPC host time-out function 1: Enable LPC host time-out function When this bit has been enabled, LPC Host Controller will count the period of waiting cycles. If the period of waiting cycles is over LPC host time-out limit (LHCR1 Bit [31:16]), LPC Host Controller will automatically abort.
0	**ENLPC-HOST**	0	RW	-	**Enable LPC Host Controller** 0: Disable LPC Host Controller 1: Enable LPC Host Controller This function is designed to support BMC controller to update system flash BIOS through LPC bus. Since LPC bus protocol doesnt support multi-master mode, LPC Host Controller can only be enabled when host platform has been full shutdown; otherwise, it will cause serious LPC bus conflictions between chipset and AST2050 / AST1100 .

LHCR1: LPC Host Control Register 1					Offset: A4h
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
31:16	**LHTMOUTLMT**	U	RW	-	**LPC host time-out limit Bit[15:0]** This register sets the maximum number of cycles that LPC Host Controller can wait for Sync Ready from LPC slave devices. If the number of waiting cycles is over the limit, LPC Host Controller will automatically abort the cycle.
15:2		0	R	-	**Reserved**
1	**LHS-ABORT**	0	RW	-	**Force LPC Host Controller to abort** 0: No operation 1: Force LPC Host Controller to abort This bit will force LPC Host Controller to stop the current bus cycle and return to its initial state. S/W needs to reset this bit to resume LPC Host Controller.
0	**LHFIRE**	0	RW	-	**Fire LPC host bus cycle** 0: No operation 1: Fire a LPC bus cycle Writing '1' to this register will force LPC Host Controller to issue one LPC bus cycle based on the information provided by LHCR4 and LHCR5. The write data will be from LHCR6. The read back data will be latched in LHCR7. S/W needs to write '0' to this bit before firing the next bus cycle by writing '1' to this bit again.

LHCR2: LPC Host Control Register 2					Offset: A8h
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
31		0	R	-	**Reserved**
30	**LSIRQCLR**	0	RW	-	**Reserved**
29	**ENLSIRQW**	0	RW	-	**Reserved**
28:8	**ENLHSR-INT**	0	RW	-	**Enable LPC Host SIRQ Interrupt Bit [20:0]** Each bit of this register represents one of the interrupt enable bit for the IRQ sources listed below. (0: Disable, 1: Enable) Bit[0]: IRQ0 Bit[1]: IRQ1 .... Bit[15]: IQR15 Bit[16]: IOCK Bit[17]: INTA Bit[18]: INTB Bit[19]: INTC Bit[20]: INTD
7:4		0	RW	-	**Reserved**
3	**ENLHTO-INT**	U	RW	-	**Enable LPC host time-out interrupt** 0: Disable 1: Enable
2	**ENLHES-INT**	U	RW	-	**Enable LPC host sync error interrupt** 0: Disable 1: Enable

to next page

from previous page

1	**NLHNS-INT**	U	RW	-	**Enable LPC host no-sync error interrupt** 0: Disable 1: Enable
0	**ENLHDN-INT**	U	RW	-	**Enable LPC host cycle done interrupt** 0: Disable 1: Enable

LHCR3: LPC Host Control Register 3 — **Offset: ACh**

Bit	Name	Initial	Slave	Host	Description
31	**LHBUSY**	U	R	-	**LPC host busy cycle** 0: LPC host is in idle cycle 1: LPC host is in busy cycle This is read only register reflecting the status of LPC host controller.
30	**LHWAIT**	U	R	-	**LPC host waiting cycle** 0: LPC host is not in waiting cycles 1: LPC host is in waiting cycles This is read only register reflecting the status of LPC host controller.
29	**STLSIRQW**	0	R	-	**Reserved**
28:8	**STR_LHSRINT**	0	RW1C	-	**LPC host SIRQ interrupt status Bit [20:0]** Each bit of this register represents the interrupt status of the 20 IRQ sources listed below (0: no interrupt, 1: interrupt pending) Bit[0]: IRQ0 Bit[1]: IRQ1 .... Bit[15]: IQR15 Bit[16]: IOCK Bit[17]: INTA Bit[18]: INTB Bit[19]: INTC Bit[20]: INTD Writing '1' to this each bit will clear the status of the corresponding interrupt.
7:4		0	R	-	**Reserved**
3	**STR_LHTOINT**	0	RW1C	-	**LPC host time-out error interrupt status** 0: No interrupt 1: Time-out error interrupt is pending Writing '1' to this bit will clear the status. Time-out means the period of waiting sync is longer than time-out limit (LHCR1 [31:16]).
2	**STR_LHESINT**	0	RW1C	-	**LPC host sync error interrupt status** 0: No interrupt 1: Sync error interrupt is pending Writing '1' to this bit will clear the status. Sync error means the LPC target device needs to aid higher layers with more robust error recovery.

to next page

from previous page

1	**STR_LHNSINT**	0	RW1C	-	**LPC host no-sync interrupt status** 0: No interrupt 1: No-sync interrupt is pending Writing '1' to this bit will clear the status. No-sync means no LPC device acknowledge.
0	**STR_LHDNINT**	0	RW1C	-	**LPC host cycle done interrupt status** 0: No interrupt 1: Cycle done interrupt is pending Writing '1' to this bit will clear the status.

LHCR4: LPC Host Control Register 4 **Offset: B0h**

Bit	Name	Initial	Slave	Host	Description
31:28	**P2LBASE**	0	RW	-	**Remapping address base bit [31:28] of APB to LPC bridge** Note: The default address decoding base bit [31:0] of APB is 0x50000000 with the range 256MB.
27:8		0	R	-	**Reserved**
7:4	**LHCMD**	U	RW	-	**LPC host command Bit[3:0]** LPC host cycles will issue LPC bus cycles with commands regarding to the content of this register. S/W takes the responsibility to provide valid commands.
3:0	**LHHDR**	U	RW	-	**LPC host start header Bit[3:0]** LPC host controller will issue LPC bus cycles with start headers regarding to the content of this register. S/W takes the responsibility to provide valid headers.

LHCR5: LPC Host Control Register 5 **Offset: B4h**

Bit	Name	Initial	Slave	Host	Description
31:0	**LHADR**	U	RW	-	**LPC host address Bit[31:0]** LPC Host Controller will issue LPC bus cycles with address regarding the content of this register. The valid address bits, which can be either 16 bits or 32 bits, are automatically determined by LPC host command (LHCR4[7:4]) and LPC host start header (LHCR4[3:0]).

LHCR6: LPC Host Control Register 6 **Offset: B8h**

Bit	Name	Initial	Slave	Host	Description
31:0	**LHTXD**	U	RW	-	**LPC host write data Bit[31:0]** LPC host write cycles will write out data based on the content of this register. The valid bytes of the write data is determined by LHCR4 [7:4]. S/W needs to check the valid bytes for each write cycle.

LHCR7: LPC Host Control Register 7					**Offset: BCh**
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
31:0	**LHRXD**	U	R	-	**LPC host read data Bit[31:0]** This register will record the data latched from the last LPC host read cycle. It will be updated whenever a new LPC host read cycle has been issued. The valid bytes of the read data is automatically determined by LPC host command (LHCR4 [7:4]). S/W needs to check the valid bytes for each read cycle.

LHCR8: LPC Host Control Register 8					**Offset: C0h**
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
31:0		U	R	-	**Reserved**

LHCR9: LPC Host Control Register 9					**Offset: C4h**
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
31:0		U	R	-	**Reserved**

LHCRA: LPC Host Control Register A					**Offset: C8h**
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
31:21		U	R	-	**Reserved**
20:0	**LSIRQEG**	U	RW	-	**LPC host SIRQ Bit[31:0] edge trigger mode** 0: level trigger 1: edge trigger

LHCRB: LPC Host Control Register B					**Offset: CCh**
Bit	**Name**	**Initial**	**Slave**	**Host**	**Description**
31:21		U	R	-	**Reserved**
20:0	**LSIRQHV**	U	RW	-	**LPC host SIRQ Bit[31:0] high/rising trigger mode** 0: low level or falling edge trigger 1: high level or rising edge trigger

31 I2C/SMBus Controller

31.1 Overview

I2C/SMBus Controller implements one set of global registers and 7 sets of device registers to program the various functions supported by AST2050 / AST1100 . Each register has its own specific offset value to derive its physical address location.

Base address of Global Register = 0x1E78_A000
Physical register address = (Base address) + Offset

31.2 Features

31.2.1 I2C Master

- Compatible with Philips I2C-BUS Specification Version 2.1
- Multi Master Operation Supported
- Software programmable clock frequency
- Software programmable AC timing
- Support a wide range of transmission speed, 0.5Kbps - 8Mbps if core clock = 50MHz
- Clock Stretching and Wait state generation
- Interrupt or bit-polling driven byte-by-byte data-transfers
- Arbitration lost interrupt, with automatic transfer cancellation
- Start/Stop/Repeated Start/Acknowledge generation
- Bus busy detection
- Automatic ACK/NACK Generation for Receive and Detection for Transmit
- Programmable Size of Buffer Mode for improving performance
- Programmable Size of DMA Mode for large amount of data transfer
- Support bus lock recovery function

31.2.2 I2C Slave

- Start/Stop/Repeated Start detection
- Supports 7 bits addressing mode only
- **Controllable** support for General Call Address
- Operates from a wide range of input clock frequencies as Master mode.
- Clock Stretching and Wait state generation
- Automatic ACK/NACK response

31.2.3 SMBus

- Compatible with SBS SMBus Specification Version 2.0
- Involved all features of I2C
- Support transmission speed from 0.5Kbps - 8Mbps if core clock = 50MHz
- **Controllable** support for ARP Default Host Address 0001 000
- **Controllable** support for ARP Default Device Address 1100 001
- **Controllable** support for Alert Response Address 0001 100
- Support 2 alert pins for 2 sets of SMBus/I2C controller (Device #1 and #2)
  - Support Master Alert interrupt
  - Support Slave Alert function

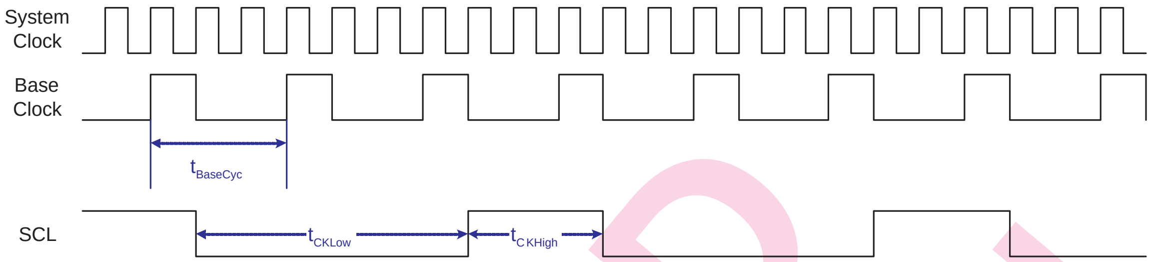
31.2.4 FML

- Device #1 and #2 can be programmed as FML controllers
- Involved all features of I2C and SMBus
- Support transmission speed up to 8Mbps

31.2.5 General

- Support totally 7 I2C/SMBus devices.
- Device #1 and #2 can be configured to FML or Alertable SMBus device.
- Support three buffer modes:
  - **Byte Buffer**: A dedicated register
  - **Pool Buffer**: 256 bytes of internal SRAM (can be dynamically allocated by the 7 devices with any size region)
  - **DMA Buffer**: 4K bytes shared form SDRAM memory (only for Device #1 and Device #2)
- Schmitt type of input data buffer and input clock buffers
- Optional anti-glitch data input
- Need external pull-up resistors

31.3 Timing Definition

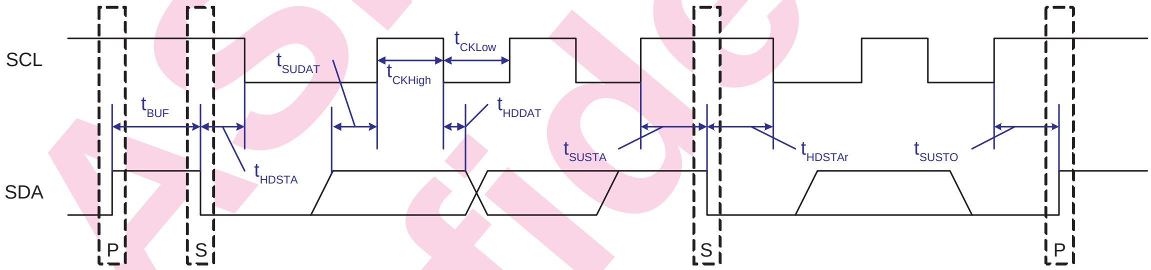


$$Freq_{SCL} = Freq_{CoreClock} / (t_{BaseCyc} * (t_{CKLow} + t_{CKHigh}))$$

$t_{BaseCyc}$ = 1, 2, 4, 8, ... , 32768
t_{CKLow} = 1 ~ 8
t_{CKHigh} = 1 ~ 8

Because all AC timing definition are based on the Base Clock, so the clock divider setting is prefer that the value of t_{CKLow} and t_{CKHigh} as larger as possible for increasing AC timing resolution.

Figure 69: Clock Prescalar



t_{SUSTA} : Repeated Start condition Setup time
t_{HDSTA} : Hold time after Start, After this period the first clock is generated
t_{HDSTAr} : Hold time after Repeated Start
t_{SUSTO} : Stop condition Setup time
t_{SUDAT} : Data Setup time
t_{HDDAT} : Data Hold time
t_{BUF} : Bus free time between Stop and Start condition

Relationship :
$t_{SUDAT} = t_{CKLow} - t_{HDDAT}$

$Width_{CKLow} = Max(t_{CKLow} , t_{HDDAT})$

t_{SUSTA} and t_{HDSTAr} and t_{SUSTO} are merged to a common timing setting t_{ACST}, so the setting for t_{ACST} is the max(t_{SUSTA}, t_{HDSTAr}, t_{SUSTO})

Figure 70: AC Timing

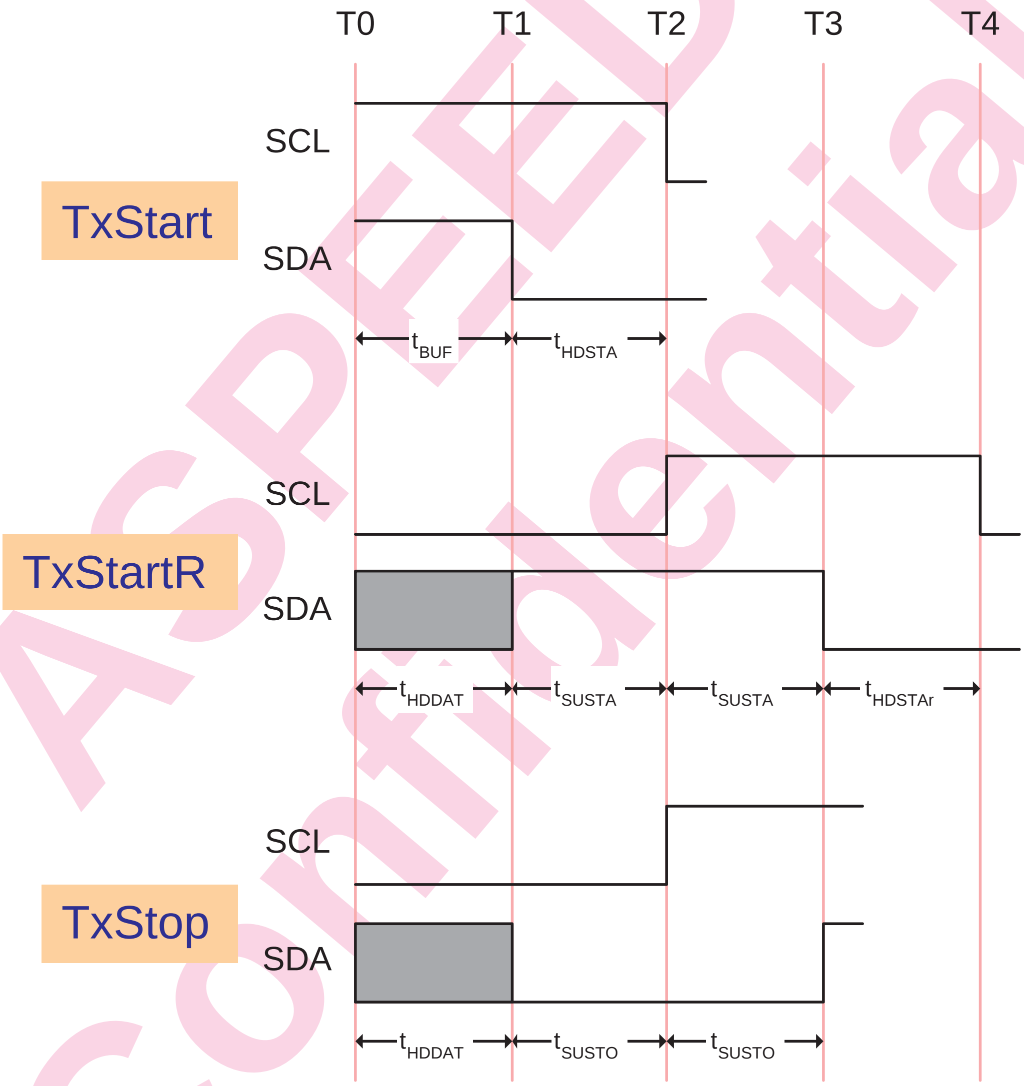


Figure 71: Master Start/Stop Timing

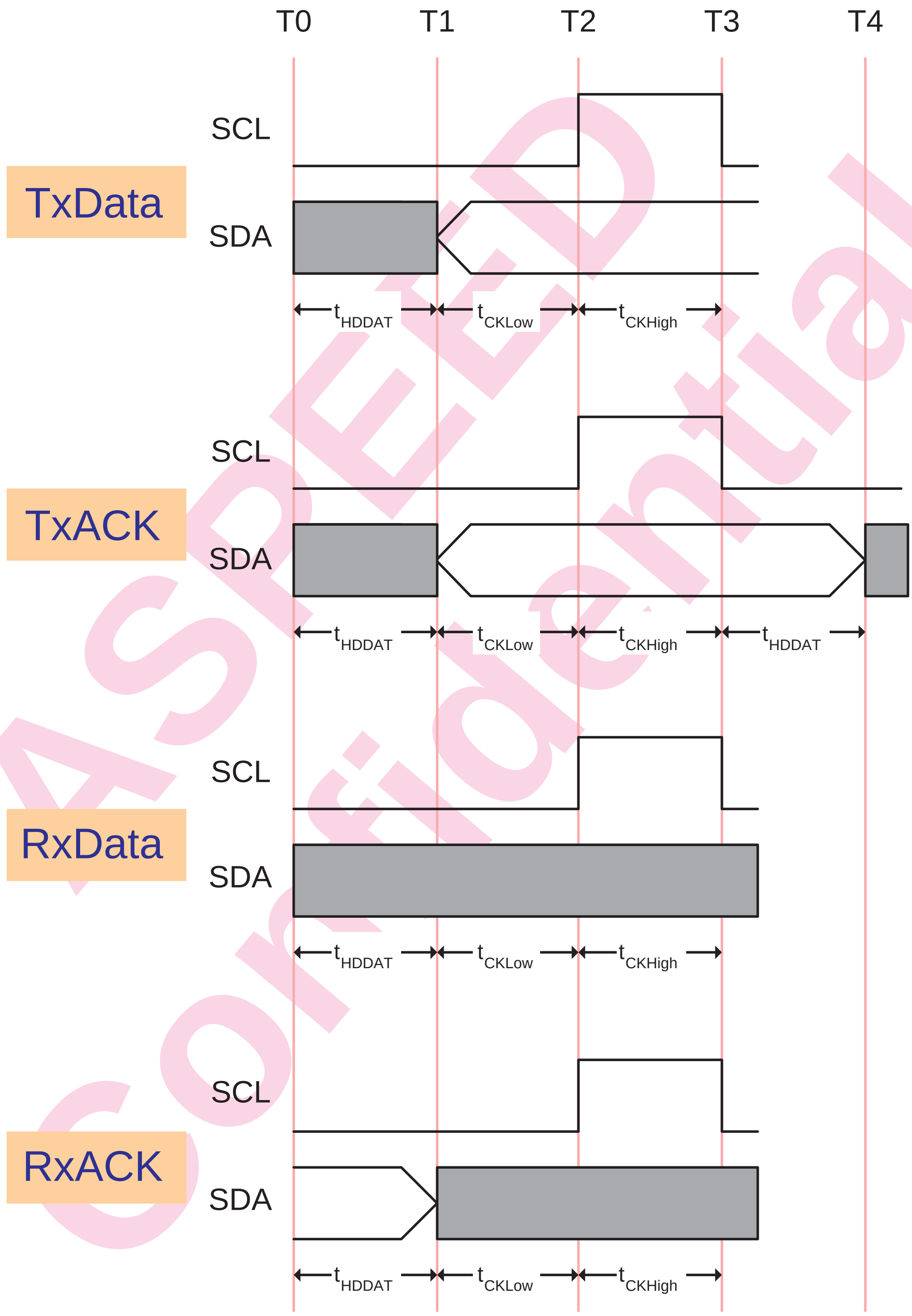


Figure 72: Master Tx/Rx Timing

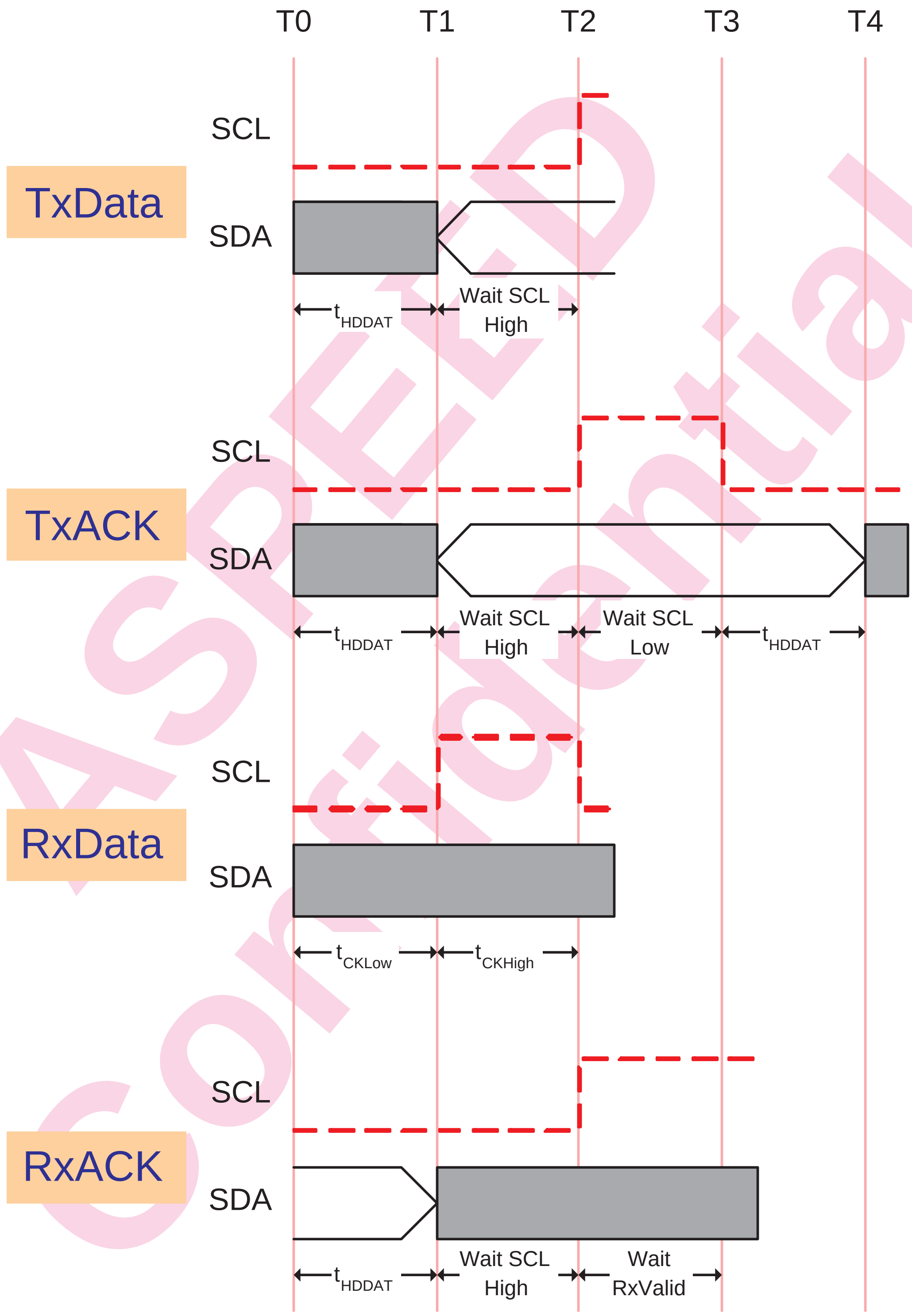


Figure 73: Slave Tx/Rx Timing

31.3.1 Clock Setting Table

- Data Bit Rate = Core Clock Frequency / Clock Divisor

Divisor	Base Clock	tCK High	tCK Low	Divisor	Base Clock	tCK High	tCK Low	Divisor	Base Clock	tCK High	tCK Low
				288	5	4	3	13312	10	6	5
6	0	2	2	320	5	4	4	14336	10	6	6
7	0	3	2	352	5	5	4	15360	10	7	6
8	0	3	3	384	5	5	5	16384	10	7	7
9	0	4	3	416	5	6	5	18432	11	4	3
10	0	4	4	448	5	6	6	20480	11	4	4
11	0	5	4	480	5	7	6	22528	11	5	4
12	0	5	5	512	5	7	7	24576	11	5	5
13	0	6	5	576	6	4	3	26624	11	6	5
14	0	6	6	640	6	4	4	28672	11	6	6
15	0	7	6	704	6	5	4	30720	11	7	6
16	0	7	7	768	6	5	5	32768	11	7	7
18	1	4	3	832	6	6	5	36864	12	4	3
20	1	4	4	896	6	6	6	40960	12	4	4
22	1	5	4	960	6	7	6	45056	12	5	4
24	1	5	5	1024	6	7	7	49152	12	5	5
26	1	6	5	1152	7	4	3	53248	12	6	5
28	1	6	6	1280	7	4	4	57344	12	6	6
30	1	7	6	1408	7	5	4	61440	12	7	6
32	1	7	7	1536	7	5	5	65536	12	7	7
36	2	4	3	1664	7	6	5	73728	13	4	3
40	2	4	4	1792	7	6	6	81920	13	4	4
44	2	5	4	1920	7	7	6	90112	13	5	4
48	2	5	5	2048	7	7	7	98304	13	5	5
52	2	6	5	2304	8	4	3	106496	13	6	5
56	2	6	6	2560	8	4	4	114688	13	6	6
60	2	7	6	2816	8	5	4	122880	13	7	6
64	2	7	7	3072	8	5	5	131072	13	7	7
72	3	4	3	3328	8	6	5	147456	14	4	3
80	3	4	4	3584	8	6	6	163840	14	4	4
88	3	5	4	3840	8	7	6	180224	14	5	4
96	3	5	5	4096	8	7	7	196608	14	5	5
104	3	6	5	4608	9	4	3	212992	14	6	5
112	3	6	6	5120	9	4	4	229376	14	6	6
120	3	7	6	5632	9	5	4	245760	14	7	6
128	3	7	7	6144	9	5	5	262144	14	7	7
144	4	4	3	6656	9	6	5	294912	15	4	3
160	4	4	4	7168	9	6	6	327680	15	4	4
176	4	5	4	7680	9	7	6	360448	15	5	4
192	4	5	5	8192	9	7	7	393216	15	5	5
208	4	6	5	9216	10	4	3	425984	15	6	5
224	4	6	6	10240	10	4	4	458752	15	6	6
240	4	7	6	11264	10	5	4	491520	15	7	6
256	4	7	7	12288	10	5	5	524288	15	7	7

31.4 Registers : Base Address = 0x1E78:A000

31.4.1 Address Definition

Offset	Size(Byte)	Description
03F-000	64	Global Register
07F-040	64	Device 1
0BF-080	64	Device 2
0FF-0C0	64	Device 3
13F-100	64	Device 4
17F-140	64	Device 5
1BF-180	64	Device 6
1FF-1C0	64	Device 7
2FF-200	256	Buffer Pool

31.4.2 Global Register Definition

Offset: 00h		I2CG00: Device Interrupt Status Register Init = 0
Bit	**R/W**	**Description**
31:7		**Reserved (0)**
6	R	**I2C/SMBus Device #7 Interrupt** 0 : No interrupt 1 : Interrupt occurs
5	R	**I2C/SMBus Device #6 Interrupt** 0 : No interrupt 1 : Interrupt occurs
4	R	**I2C/SMBus Device #5 Interrupt** 0 : No interrupt 1 : Interrupt occurs
3	R	**I2C/SMBus Device #4 Interrupt** 0 : No interrupt 1 : Interrupt occurs
2	R	**I2C/SMBus Device #3 Interrupt** 0 : No interrupt 1 : Interrupt occurs
1	R	**I2C/SMBus/FML Device #2 Interrupt** 0 : No interrupt 1 : Interrupt occurs
0	R	**I2C/SMBus/FML Device #1 Interrupt** 0 : No interrupt 1 : Interrupt occurs

Note :
This global register shows the summary report of the interrupt events from all the 7 devices. There is no need to clear the interrupt status for this register.

Offset: 04	I2CG04: I2C6/I2C7 Pin Multiplexing Setting Register	Init = 0
Bit	**R/W**	**Description**
31:2		**Reserved (0)**
1:0	RW	**Pin multiplexing setting** 00 : 7 set I2C 01 : 6 set I2C + 2 Alert for I2C1 and I2C2 10 : 6 set I2C + 1 FML for I2C1 11 : 5 set I2C + 2 FML for I2C1 and I2C2

Pin Name	00	01	10	11
SCL6/FLBINTCKEX2	SCL6	SCL6	SCL6	FLBINTCKEX2
SDA6/FLBSD2	SDA6	SDA6	SDA6	FLBSD2
SCL7/ALT1/FLBINTCKEX1	SCL7	ALT1	FLBINTCKEX1	FLBINTCKEX1
SDA7/ALT2/FLBSD1	SDA7	ALT2	FLBSD1	FLBSD1

The pin mux defines the functionality of I2C1, I2C2, I2C6 and I2C7.

31.4.3 Device Register Definition

Offset: 00h	I2CD00: Function Control Register	Init = 0
Bit	**R/W**	**Description**
31:16		**Reserved (0)**
15	RW	**Disable multi-master capability (for master function only)** 0: Enable 1: Disable (for single master application only) When disabled, device controller will assume there will be no arbitration lost possibility and ignore the related check.
14	RW	**Enable SCL direct drive mode (for master function only)** 0: Disable (SCL output buffer is configured as an open-drain buffer with an external pull-up resistor) 1: Enable (SCL always drives output buffer, no tri-state) This bit is an extension of bit[7].
13:12	RW	**Clock cycle selection for slowing down FML master clock** 00: Slow down FML master clock by 2 APB clock cycles 01: Slow down FML master clock by 3 APB clock cycles 10: Slow down FML master clock by 4 APB clock cycles 11: Slow down FML master clock by 5 APB clock cycles These two bits can only be applied to Device #1 and Device #2. And these two bits are reserved bits for other devices.
11	RW	**Enable slow down FML master clock** 0: Disable (no slow down) 1: Enable slow down FML master clock This bit can only be applied to Device #1 and Device #2. And this bit is a reserved bit for other devices.
10	RW	**Receiving data mode selection** 0: Receive SCL/SDA input data by filtering input signal (remove glitches of pulse width less than 1 APB clock cycles) 1: Receive SCL/SDA input data by sampling input signal (synchronization)
9	RW	**Data sequence MSB-First/LSB-First selection** 0 : MSB First 1 : LSB First

to next page

from previous page

8	RW	**Enable SDA signal to actively drive high for 1T** 0: Disable (SDA signal passively drives high by an external pull-up resistor) 1: Enable (SDA signal actively drives high for 1 APB clock cycle before entering tri-state) The function is designed to support a higher transfer rate.
7	RW	**Enable SCL signal to actively drive high for 1T (Master Only)** 0: Disable (SCL signal passively drives high by an external pull-up resistor) 1: Enable (SCL signal actively drives high for 1 APB clock cycle before entering tri-state) The function is designed to support a higher transfer rate.
6	RW	**Enable FML function mode** 0 : Disable 1 : Enable FML function mode. This register can only be applied to Device #1 and Device #2. This register is only a reserved bit for the other devices.
5	RW	**Enable I2C/SMBus Device Default Address (1100_001) Response** 0 : Disable 1 : Enable
4	RW	**Enable I2C/SMBus Device Alert Address (0001_100) Response** 0 : Disable 1 : Enable
3	RW	**Enable I2C/SMBus ARP Host Address (0001_000) Response** 0 : Disable 1 : Enable
2	RW	**Enable I2C/SMBus General Call Address (0000_0000) Response** 0 : Disable 1 : Enable
1	RW	**Enable slave function** 0 : Disable 1 : Enable Each device can be enabled to support either slave or master function or both functions.
0	RW	**Enable master function** 0 : Disable 1 : Enable Each device can be enabled to support either slave or master function or both functions.

Note :
The respective controller and the following registers of the respective controller will be reset whenever its master function and slave function are both disabled simultaneously.
1. I2CD0C : Interrupt Control Register
2. I2CD10 : Interrupt Status Register
3. I2CD14 : Command Control Register

<table>
<tr><th>Offset: 04h</th><th colspan="2">I2CD04: Clock and AC Timing Control Register #1   Init = X</th></tr>
<tr><th>Bit</th><th>R/W</th><th>Description</th></tr>
<tr><td>31:28</td><td>RW</td><td>Bus free time between Stop and Start timing pattern (tBUF)
0000: 1 Base Clock #1
0001: 2 Base Clock #1
....
0111: 8 Base Clock #1
1000: 1 Base Clock #2
1001: 2 Base Clock #2
....
1111: 8 Base Clock #2</td></tr>
<tr><td>27:24</td><td>RW</td><td>Hold time of master Start timing pattern (tHDSTA)
0000: 1 Base Clock #1
0001: 2 Base Clock #1
....
0111: 8 Base Clock #1
1000: 1 Base Clock #2
1001: 2 Base Clock #2
....
1111: 8 Base Clock #2</td></tr>
<tr><td>23:20</td><td>RW</td><td>Setup/Hold time of master Start/Stop timing pattern (tACST)
0000: 1 Base Clock #1
0001: 2 Base Clock #1
....
0111: 8 Base Clock #1
1000: 1 Base Clock #2
1001: 2 Base Clock #2
....
1111: 8 Base Clock #2
This register defines the setup time of Start (tSUSTA), the hold time of Repeated Start (tHDSTAr), and the setup time of Stop (tSUSTO); therefore the setting of this register must be the maximum value of all the three timing requirements.</td></tr>
<tr><td>18:16</td><td>RW</td><td>Cycles of master SCL clock-high pulse width (tCKHigh)
<table>
<tr><th></th><th>FML Enabled</th><th>FML Disabled
Divisor = 0</th><th>FML Disabled
Divisor = 1</th><th>FML Disabled
Divisor > 1</th></tr>
<tr><td>000</td><td>1</td><td>3</td><td>1.5</td><td>1</td></tr>
<tr><td>001</td><td>2</td><td>4</td><td>2</td><td>2</td></tr>
<tr><td>....</td><td></td><td></td><td></td><td></td></tr>
<tr><td>111</td><td>8</td><td>10</td><td>8</td><td>8</td></tr>
</table>
The Divisor in the table is the Base Clock #1 divisor (I2CD04: Bit [3:0]).
The unit for the table is one cycle of Base Clock #1.</td></tr>
<tr><td>14:12</td><td>RW</td><td>Cycles of master SCL clock-low pulse width (tCKLow)
000: 1 cycle of Base Clock #1
001: 2 cycles of Base Clock #1
....
111: 8 cycles of Base Clock #1</td></tr>
</table>

to next page

from previous page

11:10	RW	**Hold time of master/slave data (tHDDAT)** Master Slave 00 1 0 01 2 1 10 3 2 11 4 3 The unit for the table is numbers of Base Clock #1
9:8	RW	**Timeout base clock divisor** This divisor defines the Base Clock for Timeout Counter; the base clock is divided from APB Bus clock. 00: Divided by 16384 (16K) 01: Divided by 65536 (64K) 10: Divided by 262144 (256K) 11: Divided by 1048576 (1024K)
7:4	RW	**Base Clock #2 divisor** The divisor defines the frequency of Base Clock #1 which is divided from APB bus clock. 0000: Divided by 2 0001: Divided by 4 0010: Divided by 8 .... 1111: Divided by 65536 This register defines the frequency of a free running counter which generates Base Clock #2 for controlling the related AC timings.
3:0	RW	**Base Clock #1 divisor** The divisor defines the frequency of Base Clock #1 which is divided from APB bus clock. 0000: Divided by 1 0001: Divided by 2 0010: Divided by 4 .... 1111: Divided by 32768 This register defines the frequency of a free running counter which generates Base Clock #1 for controlling the related AC timings.

Offset: 08h — I2CD08: Clock and AC Timing Control Register #2 — Init = X

Bit	R/W	Description
31:12		**Reserved**
2:0	RW	**Cycles of clock-low timeout (tTimeOut)** 000: No Timeout Control 001: 1-2 cycles of Timeout Base Clock .... 111: 7-8 cycles of Timeout Base Clock Since Timeout Counter is a free run counter, there will be one cycle of uncertainty to generate timeout event.

Offset: 0Ch — I2CD0C: Interrupt Control Register — Init = 0

Bit	R/W	Description
31:14		**Reserved (0)**

to next page

from previous page

13	RW	**Enable Bus Recover Done interrupt**
12	RW	**Enable SMBus Device Alert interrupt**
11	RW	**Enable SMBus ARP Host Address Detection interrupt**
10	RW	**Enable SMBus Device Alert Response Address Detection interrupt**
9	RW	**Enable SMBus Device Default Address Detection interrupt**
8	RW	**Enable General Call Address Detection interrupt**
7	RW	**Enable Slave Address Received Match interrupt**
6	RW	**Enable SCL clock-low timeout interrupt**
5	RW	**Enable abnormal Start/Stop condition detection interrupt** This interrupt is used to indicate a bus Start/Stop condition has been detected at an illegal transfer state.
4	RW	**Enable normal Stop condition detection interrupt** For master mode, this interrupt is used to report that a Stop pattern has been issued. For salve mode, this interrupt is used to report a Stop pattern has been detected
3	RW	**Enable master arbitration loss interrupt Enable**
2	RW	**Enable Receive Done interrupt** Receive Done means : 1. Master : all the expected bytes have been received or the buffer is full. 2. Slave : the buffer is full, or salve controller receives a terminated signaling, and (applied to both master & slave modes) a. Last ACK/NACK returned b. Data has been received The buffer can be Byte Buffer, Poll Buffer or DMA buffer.
1	RW	**Enable Transmit with NACK Returned interrupt**
0	RW	**Enable Transmit Done with ACK Returned interrupt** Transmit Done means all the data buffer been transferred.

Note :
The definition of this register is :
0 : Disable
1 : Enable

Offset: 10h — I2CD10: Interrupt Status Register — Init = 0

Bit	R/W	Description
31:13		**Reserved**
13	RW	**(WC) Bus Recover Done interrupt status**
12	RW	**(WC) SMBus Device Alert interrupt status**
11	RW	**(WC) SMBus ARP Host Address Detection interrupt status**
10	RW	**(WC) SMBus Device Alert Response Address Detection interrupt status**
9	RW	**(WC) SMBus Device Default Address Detection interrupt status**
8	RW	**(WC) General Call Address Detection interrupt status**
7	RW	**(WC) Slave Address Received Match interrupt status**
6	RW	**(WC) SCL clock-low timeout interrupt status**
5	RW	**(WC) Abnormal Start/Stop Condition Detection interrupt status**
4	RW	**(WC) Normal Stop Condition Detection interrupt status**
3	RW	**(WC) Master Arbitration Loss interrupt status**

to next page

from previous page

2	RW	**(WC) Receive Done interrupt status** S/W needs to clear this status bit to allow next data receiving. And at byte buffer mode, this interrupt status may be set concurrently along with bit[11:7].
1	RW	**(WC) Transmit with NACK Returned interrupt status**
0	RW	**(WC) Transmit Done with ACK Returned interrupt status**
Note : 'WC' means this bit is cleared by writing '1'.		

Offset: 14h — I2CD14: Command/Status Register — Init = 0

Bit	R/W	Description
31:29		**Reserved (0)**
28	R	**SDA_OE** (for debugging purpose only)
27	R	**SDA_O** (for debugging purpose only)
26	R	**SCL_OE** (for debugging purpose only)
25	R	**SCL_O** (for debugging purpose only)
24:23	R	**Transfer Mode Timing Stage** (for debugging purpose only) 00: T0 01: T1 10: T2 11: T3
22:19	R	**Transfer Mode State Machine** (for debugging purpose only) 0000: IDLE 1000: MACTIVE 1001: MSTART 1010: MSTARTR 1011: MSTOP 1100: MTXD 1101: MRXACK 1110: MRXD 1111: MTXACK 0001: SWAIT 0100: SRXD 0101: STXACK 0110: STXD 0111: SRXACK 0011: RECOVER
18	R	**Sampled SCL Line State**
17	R	**Sampled SDA Line State**
16	R	**Bus Busy Status** 0: Bus is Idle 1: Bus is Busy or not meets Idle Timing Requirement
15	RW	**SDA_OE output direct control** Bit[15:12] is a GPIO function and only work when both Master and Slave function are disabled. When bus lock occurs, this GPIO function can help to recover the bus life. 0: output disable, tri-stated 1: output enable
14	RW	**SDA_O output direct control**
13	RW	**SCL_OE output direct control** 0: output disable, tri-stated 1: output enable

to next page

from previous page

12	RW	**SCL_O output direct control**
11	RW	**Enable Bus Recover Command** 0: NOP 1: Issue Bus Recover Command This command can be applied only when SCL line status is High. When Bus Recovery Command is fired, 1 ∼ 8 SCL clock cycles will be issued to recover the bus. (Automatically stop issuing clock cycle whenever SDA has been released to high state) This command is used to recover the bus when a slave device locks SDA. When using this command, the Transfer Mode State Machine must be at the IDLE state; otherwise, it must be reset to the IDLE state firstly by disabling the device function (I2CD00). When recover operation has been done, the current bus state can be read back from Bit [17] of this register.
10	RW	**Enable issuing I2C/SMBus Slave Alert signal** 0: NOP 1: Issuing Alert signal to a bus Master This command is valid only when slave function is enabled. Only Device #1 and Device #2 support this command. This register is a reserved bit for other devices. This bit will be cleared by hardware automatically after packet received with slave address matched.
9	RW	**Enable Master/Slave Receive from DMA Buffer** 0: Disable (Receiving data and writing into either Byte Buffer or Pool Buffer) 1: Enable (Receiving data from DMA Buffer) HW will clear this register automatically when data transmitting has been done. Only Device #1 and Device #2 support this command.
8	RW	**Enable Master/Slave Transmit from DMA Buffer** 0: Disable (Transmitting data from in either Byte Buffer or Pool Buffer) 1: Enable (Transmitting data stored in DMA Buffer) HW will clear this register automatically when data transmitting has been done. Only Device #1 and Device #2 support this command.
7	RW	**Enable Master/Slave Receive Data Buffer** 0: Disable (Receiving data and writing into either byte buffer or DMA buffer) 1: Enable (Receiving data and writing into Pool Buffer) This register will be automatically cleared by H/W when data receiving has been done.
6	RW	**Enable Master/Slave Transmit Data Buffer** 0: Disable (Transmitting data from either Byte Buffer or Pool Buffer) 1: Enable (Transmitting data from Pool Buffer This register will be automatically cleared by H/W when data receiving has been done.
5	RW	**Master Stop Command** 0: NOP 1: Issue Master Stop Command **4th** priority. This register will be automatically cleared by H/W when Stop Command has been issued. This command is valid only when master mode is enabled
4	RW	**Master/Slave Receive Command Last** 0: Receive Command can be continued by responding ACK 1: Receive Command will be ended by responding NACK When in buffer mode, the last control will acts after the lastest byte is been received. When in Master mode and Stop Command activated, the last control must be set to ending transfer.

to next page

from previous page

3	RW	**Master Receive Command** 0: NOP 1: Fire Master Receive Command **3rd** priority. HW will clear this register when RX buffer is full or receiving is terminated (Stop/Repeated Start). This command is valid only when Master mode is enabled
2	RW	**Slave Transmit Command** 0: NOP 1: Fire Slave Transmit Command HW will clear this register when TX buffer is empty or bus contention error has been detected.
1	RW	**Master Transmit Command** 0: NOP 1: Fire Master Transmit Command **2nd** priority. HW will clear this register when TX buffer is empty or Bus Contention error has been detected.
0	RW	**Master Start Command** 0: NOP 1: Issue Master Start/Repeated Start Command **1st** priority. This register will be automatically cleared by HW when Master Start Command or Repeated Start Command has been issued. This command will be executed by HW only when master mode is enabled and the bus is in idle state.

Note :
When multiple commands in this register are fired simultaneously, Device controller will execute these commands according to the following sequence (priority):

(1) Master Start Command
(2) Master Transmit Command
(3) Slave Transmit Command or Master Receive Command
(4) Master Stop Command

HW will automatically clear each command when it has been finished. Beside, HW will clear all the commands whenever Master Arbitration Lost or invalid Start/Stop conditions have been detected.
Attention: Master and Slave Command cannot be activated at the same time.

Offset: 18h — I2CD18: Slave Device Address Register — Init = X

Bit	R/W	Description
31:7		**Reserved (0)**
6:0	RW	**Slave Device Address**

Offset: 1Ch		I2CD1C: Pool Buffer Control Register Init = X
Bit	**R/W**	**Description**
31:24	R	**Actual Received Pool Buffer Address Pointer** 0 = 256 bytes 1 = 1 byte 2 = 2 bytes .... 255 = 255 bytes Actual Received Data Byte Count = Actual Received Address - Base Address * 4 + 1 When actual received address = 0, it means the byte count is 256 bytes (0x100)
23:16	RW	**Receive Pool Buffer End Address** 0 = 1 byte space 1 = 2 bytes space .... 255 = 256 bytes space Receive Buffer Size = End Address - Base Address * 4 + 1 This address indicates the latest Pool address that this controller can use for receiving.
15:8	RW	**Transmit Data Buffer Ended Address** 0 = 1 byte 1 = 2 bytes 2 = 3 bytes 3 = 4 bytes .... 255 = 256 bytes Transmit Data Byte Count = End Address - Base Address * 4 + 1 This address indicates the latest Pool address that this controller can use for transmitting.
7:6		**Reserved**
5:0	RW	**Buffer Base Address Pointer** Buffer Base Address Pointer is the start address of Transmit/Receive Buffer, its unit is one double word. Since Transmit Command is with higher priority than Receive Command, one base address is enough for both Transmit and Receive Command.

Offset: 20h		I2CD20: Transmit/Receive Byte Buffer Register Init = X
Bit	**R/W**	**Description**
31:16		**Reserved**
15:8	R	**Receive Byte Buffer** This register is applicable when Pool Buffer and DMA Buffer are not enabled.
7:0	RW	**Transmit Byte Buffer** This register is applicable when Pool Buffer and DMA Buffer are not enabled.

Offset: 24h		I2CD24: DMA Mode Control Register (Device #1/#2 Only) Init = X
Bit	**R/W**	**Description**
31:28		**Reserved**
27:12	RW	**DMA Buffer Base Address** DMA Buffer can be allocated from any location of SDRAM memory with 4K bytes boundary.

to next page

from previous page

11:0	RW	**DMA Buffer Size (Byte)** DMA Buffer is allocated from SDRAM memory. The maximum size of a DMA Buffer is 4K bytes. For transmit mode: (byte boundary is fine) 0: transmit 1 byte 1: transmit 2 bytes ... For receive mode: (8-bytes boundary is required) 0: receive 1 byte, must allocate 8 bytes buffer 1: receive 2 bytes, must allocate 8 bytes buffer ... 8: receive 9 bytes, must allocate 16 bytes buffer ...

Offset: 28	**I2CD28: DMA Mode Status Register (Device #1/#2 Only)**	**Init = X**
Bit	**R/W**	**Description**
31:12		**Reserved**
11:0	R	**The last accessed address for DMA Buffer** The register reports the last access address for DMA buffer (8-byte boundary), or the last byte count received from the bus (byte boundary) When transmit done, this address = DMA_Buffer_Size When receive done, the maximum address = DMA_Buffer_Size + 1

31.5 Software Programming Guide

31.5.1 Initialization

1. Write I2CD00 = enable function setting
2. Write I2CD04
3. Write I2CD08
4. Write I2CD10 = FFFFFFFF
5. Write I2CD0C = interrupt enable setting
6. Master and Slave function mode can be open individually or concurrently.

31.5.2 Byte Buffer and Buffer Pool Usage

- This IP supports Tx/Rx Byte buffer and buffer pool options.
- SW can choose 1 type or mix 2 types when using.
- The Byte buffer is dedicated to each device controller, but buffer pool is shared.
- The buffer pool can be assigned for Tx and Rx at the same time in a single command.

31.5.3 Buffer Pool Allocation

- The buffer pool is available for all devices.
- Follows the steps to use buffer pool:
  1. Allocate a region, SW management the buffer allocation.

2. Fill data in the buffer for transmitting coommand.
3. Set buffer control register I2CD1C for base pointer and Tx/Rx end pointer.
4. Set command/status register I2CD14, bit[6] : Tx_buffer_enable, bit[7] : Rx_buffer_enable. SW can fire Tx and Rx buffer command at the same time.

31.5.4 Master Mode Command

- In Master mode, there are 4 command types:
  1. 1st priority : Transmit Start pattern, when set, HW will generate Start pattern onto Bus when bus is idle.
  2. 2nd priority : Transmit Data, transmitting data until buffer is empty, arbitration lost or invalid bus condition occurs.
  3. 3rd priority : Receive Data, receiving data until buffer is full, transmission is stopped by NACK response or invalid bus condition occurs.
  4. 4th priority : Transmit Stop pattern, generate Stop pattern onto Bus.
- The 4 command types can be set at any combination follows the priority definition.
- The Receive Data Last command is part of Receive Data command for responding NACK state at the end of receiving cycle.

31.5.5 Slave Mode Command

- In Slave mode, controller is default at receiving mode and monitor the bus state to check if a matched address packet occurs, when such packet detected, HW will ACK this packet automatically and generate an interrupt to SW if interrupt is enabled.
- For the first byte when receiving, it is not recommend to use buffer pool mode, because HW do not clarify that the packet is for reading or writing, it will keep at receiving mode until the buffer full. Of course, SW can enable the buffer pool mode at start-up for the first packet receiving, if SW can confirm the next received packet is a write command.
- In Slave function mode, it does not support continuous command sequence like the Master function do.

31.5.6 Master/Slave Dual Mode Command

- The controller is designed with the capability to function as a Master and Slave concurrently.
- The usage is the same as Master/Slave individually.
- But if the Master function is accessing a device and another Master is accessing the Slave function at the same time, there will have a arbitration procedure, if the Slave function address is smaller than the device that the Master function access, then the Master function will loose the arbitration and the controller will terminate the Master command immediately and switch to Slave function mode.
- The Master command and Slave command can not be set at the same time.

31.5.7 Interrupt Handler

- When interrupt occurred, SW interrupt handler first read I2CG00 to decide which controller activating the interrupt.
- And then, read the controller interrupt status register (I2CD10) to see what happened.
- When processed the related interrupt operation, write '1' to clear the individual interrupt flag.

31.5.8 Resetting Device

- If SW wants to reset the device controller, it only needs to turn off both the Master and Slave function, then HW will reset all state machine and status (including I2CD0C, I2CD10 and I2CD14).
  - Write I2CD00 bit[1:0] = 0 , reset device.
  - The following status will be cleared when reset:
    - I2CD0C : The interrupt enable control will be cleared to 0.
    - I2CD10 : The interrupt status flag will be cleared.
    - I2CD14 : The command register will be cleared.
    - All internal control state machine and counter will be reset to initial state.
  - Write I2CD00 bit[1:0] = 3 , enable Master/Slave device (defined by user).
  - Write I2CD0C = interrupt mode needed.
- But if the I2C Bus was locked by another device, reset the controller can not recover the bus life. It needs to implement the recover operation.

31.5.9 DMA Buffer Mode Usage

- Device1 and Device2 support another DMA buffer mode. Which has a maximum 4Kbyte buffer allocated in the DRAM.
- The usage of DMA buffer mode is the same as Buffer Pool mode, except for the different data store area.

31.5.10 Command and Interrupt Processing Sequence

- Master : Clear interrupt status first and then active command.
- Slave : Active command first and then clear interrupt status. Because at Slave mode, there is no receive enable control, it is the default mode, so must set the command first to clarify the next command is transmit or receive cycle, then clear the interrupt status to fire the operation.

31.5.11 SDA Bus Lock Recover

When I2C bus lock occurred,

- Master function : Implement bus lock recover program.
- Slave function : Reset device to release bus lock. Don't implement bus lock recover program.

This controller supports 2 ways to solve SDA bus lock conditions. The first one is auto recover mode, which was implemented by hardware automatically. The second is GPIO mode, which was implemented by software and more flexibility.

Bus Lock Recognition When command or SCL low timeout occurred, the bus operation must have something wrong occurred. At this time, it must to make sure what is the last condition on the bus.

1. First release the bus by disabling the bus function (I2CD00[1:0] = 0).
2. Check the bus state on I2CD14[18:17]:
   - bit[18] = 1 and bit[17] = 0, it is a SDA bus lock condition and mostly can be recovered.
   - bit[18] = 1 and bit[17] = 1, there is no bus lock, it may SW programming error.
   - other cases, difficultly to recover.

Auto Recover Mode

1. Enable Master function
2. Set Recover command I2CD14[11] = 1
3. Wait recover done interrupt
4. Check bus state, I2CD14[18:17] = ”11”.
   - If meet, go to next.
   - If fail and retry $<$ 2, go to step 2 again.
   - If fail and retry $>=$ 2, bus cannot be recovered.
5. Set Start + Stop command to reset bus.

GPIO Mode At this mode, SW can implement your own recover mechanism.

31.6 Software Programming Example

31.6.1 Clock Rate Calculation (< 1MHz)

1. Get the H-PLL frequency, SCU70[11:9]
   000 : 266 MHz
   001 : 233 MHz
   010 : 200 MHz
   011 : 166 MHz
   100 : 133 MHz
   101 : 100 MHz
   110 : 300 MHz
   111 : 24 MHz

2. Get the PCLK frequency, SCU08[25:23]
   000 : PCLK = H-PLL/2
   001 : PCLK = H-PLL/4
   010 : PCLK = H-PLL/6
   011 : PCLK = H-PLL/8
   100 : PCLK = H-PLL/10
   101 : PCLK = H-PLL/12
   110 : PCLK = H-PLL/14
   111 : PCLK = H-PLL/16

3. Get the divider ratio = PCLK / I2C_desired_clock_rate

4. Get the Base Clock Divisor I2CD04[3:0]

```
inc = 0;
for(div = 0;divider_ratio >= 16;div++){
    inc | = divisor_ratio & 0x1;
    divider_ratio >>= 1;
}
divisor_ratio += inc;
I2CD04[3:0] = div;
I2CD04[7:4] = div;
```

5. Get the SCL High/Low pulse width setting
   SCL_Low I2CD04[14:12] = (divider_ratio >> 1) - 1;
   SCL_High I2CD04[18:16] = divider_ratio - SCL_Low - 2;

6. I2CD04 = 0x777XX3XX add above results

31.6.2 Master Transmit 1 Byte

1. Write I2CD00 = 0x00000001 , enable function mode

2. Set AC Timing : Assume PCLK = 50MHz, and I2C Clock rate = 100KHz, then
   - 50MHz / 100KHz = 500(32x16) ==> Div(5), CKLow(7), CKHigh(7) $\sim$ 97.7KHz
   - tBUF = 5120 ns ==> 8 x BaseClk
   - tHDSTA = 5120 ns ==> 8 x BaseClk
   - tSUSTA = tHDSTAr = tSUSTO = 5120 ns ==> 8 x BaseClk
   - Write I2CD04 = 0x77777355
   - Write I2CD08 = 0x00000000

3. Write I2CD10 = 0xFFFFFFFF , clearing interrupt status

4. Write I2CD0C = 0x000000BF , enable interrupt

5. Write I2CD20 = 0x000000DD , transmit byte data

6. Fire Command :
   - Write I2CD14 = 0x00000023 , Start –> Tx Byte –> Stop
   - Write I2CD14 = 0x00000003 , Start –> Tx Byte –> Waiting
   - Write I2CD14 = 0x00000002 , Waiting –> Tx Byte –> Waiting
   - Write I2CD14 = 0x00000022 , Waiting –> Tx Byte –> Stop
7. Waiting for Interrupt Event :
   - Read I2CD10 bit[0] == 1 , transmit byte finished OK.
   - Read I2CD10 bit[1] == 1 , transmit byte finished FAIL, response with NACK.
   - Read I2CD10 bit[3] == 1 , master arbitration lost, when arbitration lost, all command will be stopped and halted.
   - Read I2CD10 bit[4] == 1 , transmit byte finished, and STOP condition issued.
   - Read I2CD10 bit[5] == 1 , transmit FAIL, invalid STOP condition occurs, all command will be stopped and halted.
   - Read I2CD10 bit[6] == 1 , SCL low waiting timeout.
   - All other bits equals to ’1’ are invalid for this operation.
8. Write I2CD10 = 0xFFFFFFFF , clearing interrupt status

31.6.3 Master Transmit using Buffer Mode

1. Continues from byte mode prepare data stage (Write I2CD20).
2. Prepare buffer data.
3. Set buffer pointer :
   - Write I2CD1C bit[5:0] = buffer start pointer, 4 bytes boundary unit.
   - Write I2CD1C bit[15:8] = end address for transmitting, the transmitted byte count = bit[15:8] - (bit[5:0] * 4) + 1.
4. Fire Command :
   - Write I2CD14 = 0x00000063 , Start –> Tx Buffer –> Stop
   - Write I2CD14 = 0x00000043 , Start –> Tx Buffer –> Waiting
   - Write I2CD14 = 0x00000042 , Waiting –> Tx Buffer –> Waiting
   - Write I2CD14 = 0x00000062 , Waiting –> Tx Buffer –> Stop
5. Waiting for Interrupt Event :
   - Read I2CD10 bit[0] == 1 , transmit buffer finished OK.
   - Read I2CD10 bit[1] == 1 , transmit buffer ended FAIL, response with NACK.
   - Read I2CD10 bit[3] == 1 , master arbitration lost, when arbitration lost, all command will be stopped and halted.
   - Read I2CD10 bit[4] == 1 , transmit buffer finished, and STOP condition issued.
   - Read I2CD10 bit[5] == 1 , transmit FAIL, invalid STOP condition occurs, all command will be stopped and halted.
   - Read I2CD10 bit[6] == 1 , SCL low waiting timeout.
   - All other bits equals to ’1’ are invalid for this operation.
6. Write I2CD10 = 0xFFFFFFFF , clearing interrupt status

31.6.4 Master Transmit using DMA Mode

1. Continues from byte mode prepare data stage (Write I2CD20).
2. Prepare DMA buffer data.
3. Set DMA address :
   - Write I2CD24 bit[27:12] = buffer start address, 4K bytes boundary.
   - Write I2CD24 bit[11:0] = end address for transmitting, the transmitted byte count = bit[11:0] + 1.
4. Fire Command :
   - Write I2CD14 = 0x00000123 , Start –> Tx DMA –> Stop
   - Write I2CD14 = 0x00000103 , Start –> Tx DMA –> Waiting
   - Write I2CD14 = 0x00000102 , Waiting –> Tx DMA –> Waiting
   - Write I2CD14 = 0x00000122 , Waiting –> Tx DMA –> Stop
5. Waiting for Interrupt Event :
   - Read I2CD10 bit[0] == 1 , transmit DMA finished OK.
   - Read I2CD10 bit[1] == 1 , transmit DMA ended FAIL, response with NACK.
   - Read I2CD10 bit[3] == 1 , master arbitration lost, when arbitration lost, all command will be stopped and halted.
   - Read I2CD10 bit[4] == 1 , transmit DMA finished, and STOP condition issued.
   - Read I2CD10 bit[5] == 1 , transmit FAIL, invalid STOP condition occurs, all command will be stopped and halted.
   - Read I2CD10 bit[6] == 1 , SCL low waiting timeout.
   - All other bits equals to '1' are invalid for this operation.
6. The final transmitted bytes = I2CD28 bit[11:0] + 1.
7. Write I2CD10 = 0xFFFFFFFF , clearing interrupt status

31.6.5 Master Receive 1 Byte

1. Following the initial squence of Write operation
2. Fire Command :
   - Write I2CD14 = 0x00000008 , Waiting –> Rx Byte –> Tx ACK –> Waiting
   - Write I2CD14 = 0x00000018 , Waiting –> Rx Byte –> Tx NACK –> Waiting
   - Write I2CD14 = 0x00000038 , Waiting –> Rx Byte –> Tx NACK –> Stop
3. Waiting for Interrupt Event :
   - Read I2CD10 bit[2] == 1 , receive byte finished OK.
   - Read I2CD10 bit[4] == 1 , receive byte finished, and STOP condition issued.
   - Read I2CD10 bit[5] == 1 , receive FAIL, invalid STOP condition occurs, all command will be stopped and halted.
   - Read I2CD10 bit[6] == 1 , SCL low waiting timeout
   - All other bits equals to '1' are invalid for this operation.
4. Write I2CD10 = 0xffffffff , clearing interrupt status

31.6.6 Master Receive using Buffer Mode

1. Following the initial squence of Write operation
2. Allocate buffer for receiving :
   - Write I2CD1C bit[5:0] = buffer start pointer, 4 bytes boundary unit.
   - Write I2CD1C bit[23:16] = end address for receiving, the allocated space = bit[15:8] - (bit[5:0] * 4) + 1 bytes.
3. Fire Command :
   - Write I2CD14 = 0x00000088 , Waiting –> Rx Buffer –> Tx ACK –> Waiting
   - Write I2CD14 = 0x00000098 , Waiting –> Rx Buffer –> Tx NACK after received the last byte –> Waiting
   - Write I2CD14 = 0x000000B8 , Waiting –> Rx Buffer –> Tx NACK after received the last byte –> Stop
4. Waiting for Interrupt Event :
   - Read I2CD10 bit[2] == 1 , receive buffer finished OK.
   - Read I2CD10 bit[4] == 1 , receive buffer finished, and STOP condition issued.
   - Read I2CD10 bit[5] == 1 , receive FAIL, invalid STOP condition occurs, all command will be stopped and halted.
   - Read I2CD10 bit[6] == 1 , SCL low waiting timeout
   - All other bits equals to '1' are invalid for this operation.
5. The actual received bytes can be get from I2CD1C bit[31:24].
6. Write I2CD10 = 0xffffffff , clearing interrupt status

31.6.7 Master Receive using DMA Mode

1. Following the initial squence of Write operation
2. Allocate buffer for receiving :
   - Write I2CD24 bit[27:12] = buffer start address, 4K bytes boundary.
   - Write I2CD24 bit[11:0] = end address for receiving, the actual allocated space = (bit[11:3] + 1) * 8 bytes.
3. Fire Command :
   - Write I2CD14 = 0x00000208 , Waiting –> Rx DMA –> Tx ACK –> Waiting
   - Write I2CD14 = 0x00000218 , Waiting –> Rx DMA –> Tx NACK after received the last byte –> Waiting
   - Write I2CD14 = 0x00000238 , Waiting –> Rx DMA –> Tx NACK after received the last byte –> Stop
4. Waiting for Interrupt Event :
   - Read I2CD10 bit[2] == 1 , receive DMA finished OK.
   - Read I2CD10 bit[4] == 1 , receive DMA finished, and STOP condition issued.
   - Read I2CD10 bit[5] == 1 , receive FAIL, invalid STOP condition occurs, all command will be stopped and halted.
   - Read I2CD10 bit[6] == 1 , SCL low waiting timeout
   - All other bits equals to '1' are invalid for this operation.
5. The final received bytes = I2CD28 bit[11:0].
6. Write I2CD10 = 0xffffffff , clearing interrupt status

31.6.8 Master Transmit 1 Byte and then Receive 1 Byte

1. Following the initial sequence of Write operation
2. Write I2CD20 = 0x000000DD , transmit byte data
3. Fire Command :
   - Write I2CD14 = 0x0000003B , Start –> Tx Byte –> Rx Byte –> Stop
   - Write I2CD14 = 0x0000000B , Start –> Tx Byte –> Rx Byte –> Waiting
   - Write I2CD14 = 0x0000000A , Waiting –> Tx Byte –> Rx Byte –> Waiting
   - Write I2CD14 = 0x0000003A , Waiting –> Tx Byte –> Rx Byte –> Stop
4. Waiting for Interrupt Event :
   - Read I2CD10 bit[0] == 1 , transmit byte finished OK.
   - Read I2CD10 bit[1] == 1 , transmit byte finished FAIL, response with NACK.
   - Read I2CD10 bit[2] == 1 , receive byte finished OK.
   - Read I2CD10 bit[3] == 1 , master arbitration lost, when arbitration lost, all command will be stopped and halted.
   - Read I2CD10 bit[4] == 1 , transmit byte finished, and STOP condition issued.
   - Read I2CD10 bit[5] == 1 , transmit FAIL, invalid STOP condition occurs, all command will be stopped and halted.
   - Read I2CD10 bit[6] == 1 , SCL low waiting timeout
   - All other bits equals to '1' are invalid for this operation.
5. Write I2CD10 = 0xFFFFFFFF , clearing interrupt status

31.6.9 Slave Receive Address Match

1. Following the initial squence of Write operation.
2. Write I2CD18 = device address.
3. When the Slave controller received a packet that match the address assigned, then the following interrupt status flag will be raised:
   - I2CD10 bit[2] == 1 , indicates 1 byte data received.
   - I2CD10 bit[7] == 1 , indicates the received 1 byte data was the address byte.
4. At this time, SW must check the data bit[0] (I2CD20[8]) to determine the next transfer direction.
   - I2CD20 bit[8] == 0 , indicates write mode, the following bytes were sent from master until STOP or START conditions occurs.
   - I2CD20 bit[8] == 1 , indicates read mode, the following bytes were sent by slave until a TxNAK (I2CD10[1]) condition occurs.
5. Continues with following sections for Write or Read mode operations.

31.6.10 Slave Write Mode

1. Prepare the receive buffer.
   - Byte mode: NOP
   - Buffer mode:
     - Write I2CD1C bit[5:0] = buffer start pointer, 4 bytes boundary unit.
     - Write I2CD1C bit[23:16] = end address for receiving, the allocated space = bit[15:8] - (bit[5:0] * 4) + 1.
     - Write I2CD14 bit[7] = 1, enable receive buffer.
   - DMA mode:
     - Write I2CD24 bit[27:12] = buffer start address, 4K bytes boundary.
     - Write I2CD24 bit[11:0] = end address for receiving, the actual allocated space = (bit[11:3] + 1) * 8 bytes.
     - Write I2CD14 bit[9] = 1, enable receive DMA.
2. Write I2CD10 = 0xFFFFFFFF , clearing interrupt status.
3. Waiting I2CD10 bit[2] == 1, receive done interrupt.
4. Check data
   - Byte mode: Read I2CD20 bit[15:8], received data.
   - Buffer mode:
     - Check I2CD1C bit[31:24] for the final received byte count.
     - Read data from the buffer pool.
   - DMA mode:
     - The final received byte count = I2CD28 bit[11:0].
     - Read data from the DMA buffer.
5. Goto step 1 for next byte receiving.

31.6.11 Slave Read Mode

1. Prepare data.
   - Byte mode: Write I2CD20 bit[7:0] = data for transmitting.
   - Buffer mode:
     - Fill data into the buffer.
     - Write I2CD1C bit[5:0] = buffer start pointer, 4 bytes boundary unit.
     - Write I2CD1C bit[15:8] = end address for transmitting, the transmitted byte count = bit[15:8] - (bit[5:0] * 4) + 1.
     - Write I2CD14 bit[6] = 1, enable transmit buffer.
   - DMA mode:
     - Write I2CD24 bit[27:12] = buffer start address, 4K bytes boundary.
     - Write I2CD24 bit[11:0] = end address for transmitting, the transmitted byte count = bit[11:0] + 1 bytes.
     - Write I2CD14 bit[8] = 1, enable transmit DMA.
2. Write I2CD14 bit[2] = 1, enable slave transmit command.
3. Write I2CD10 = 0xFFFFFFFF , clearing interrupt status.
4. Waiting transmit done interrupt:
   - I2CD10 bit[0] == 1, transmit done with ACK, goto step 1 for next data transmitting.
   - I2CD10 bit[1] == 1, transmit done with NACK, ending transmit.
     - DMA mode: The final transmitted data byte count = I2CD28 bit[11:0] + 1.

31.6.12 Master Alert Handler

- When I2CD10 bit[12] == 1, indicates a slave device issuing a service request.
- Software start to polling all alertable slave devices until I2CD10 bit[12] == 0.
  1. Write I2CD20 bit[7:0] = slave device address.
  2. Write I2CD14 = 0x00000023
  3. Waiting I2CD10 bit[4] == 1
  4. Check I2CD10 bit[12]
     - bit[12] == 0 , polling device not the alert device, polling the next device.
     - bit[12] == 1 , polling device is the alert device, service it. Stop polling.

31.6.13 Slave Alert Handler

- Write I2CD14 bit[10] = 1, issuing alert.
- Wait Master's service.

31.6.14 High Speed Mode (> 1MHz) Programming Example

To run at high speed mode, a speed change sequence must be followed for each packet transfer. Below is some registers setting for high speed mode operation that differs from normal speed operation:

- I2CD00 bit[8:7] = "11", this enables an active current source to faster and sharpen the rising edge.
- Internal bus clock PCLK must be larger than 33MHz.
  - For AST2050/1100, H-PLL = 204MHz, set SCU08[25:23]="010" and get PCLK = 34MHz.
  - For AST2200/2100/2150, H-PLL = 264MHz, set SCU08[25:23]="011" and get PCLK = 33MHz.

There are also some AP circuit requirements to meet high speed mode timing:

- Pull high resistor = 1K ohm
- Maximum series resistor $\leq$ 33 ohm
- Maximum trace capacitive load must be smaller than 100pF (3.4MHz) or 400pF (1.7MHz)

Below is the speed change sequence for each packet transfer (between Start and Stop).

1. Set to normal speed
2. Send High speed master code command 0x08 $\sim$ 0x0F (for multi-master option)
3. Set to high speed
4. Start data transfer

Below is an example of high speed mode (400KHz/3MHz) initialization.

1. Write I2CD00 = 0x181
2. Write I2CD04 = 0x77744333
3. Write I2CD08 = 0x03
4. Write I2CD10 = 0xFFFFFFFF
5. Write I2CD14 = 0x0
6. Write I2CD0C = interrupt enable setting

Below is an example of high speed mode (400KHz/3MHz) transfer.

1. Write I2CD04 = 0x77744333
2. Write I2CD20 = 0x08
3. Write I2CD14 = 0x03
4. Wait transmit done interrupt
5. Write I2CD04 = 0x77725700
6. Write I2CD20 = slave address for write
7. Write I2CD14 = 0x03
8. Wait transmit done interrupt
9. Write I2CD20 = data
10. Write I2CD14 = 0x02
11. Wait transmit done interrupt
12. Repeat step 9 ~ 11 until all write data was sent
13. Write I2CD20 = slave address for read
14. Write I2CD14 = 0x03
15. Wait transmit done interrupt
16. Write I2CD14 = 0x08 or 0x18 for the last byte read
17. Wait receive done interrupt
18. Repeat step 16 ~ 17 until all data was read
19. Write I2CD14 = 0x20
20. Wait stop interrupt

Below are some speed setting for PCLK = 25MHz (H-PLL 204MHz/8):

96KHz	I2CD04 = 0x77777344
103KHz	I2CD04 = 0x77767344
384KHz	I2CD04 = 0x77777322
412KHz	I2CD04 = 0x77767322
772KHz	I2CD04 = 0x77767711
1546KHz	I2CD04 = 0x77766700
2475KHz	I2CD04 = 0x77724700

Below are some speed setting for PCLK = 33MHz (H-PLL 204MHz/6 or 264MHz/8):

96KHz	I2CD04 = 0x77745355
103KHz	I2CD04 = 0x77744355
384KHz	I2CD04 = 0x77745333
412KHz	I2CD04 = 0x77744333
750KHz	I2CD04 = 0x77744722
1650KHz	I2CD04 = 0x77734711
3000KHz	I2CD04 = 0x77725700 (SCL Low = 180 ns, SCL High = 150 ns)

Below are some speed setting for PCLK = 44MHz (H-PLL 264MHz/6):

98KHz	I2CD04 = 0x77766355
105KHz	I2CD04 = 0x77756355
393KHz	I2CD04 = 0x77766333
786KHz	I2CD04 = 0x77756722
1570KHz	I2CD04 = 0x77756711
3143KHz	I2CD04 = 0x77737700 (SCL Low = 184 ns, SCL High = 134 ns)

Below are some speed setting for PCLK = 50MHz (H-PLL 204MHz/4):

96KHz	I2CD04 = 0x77777355
103KHz	I2CD04 = 0x77767355
386KHz	I2CD04 = 0x77777333
412KHz	I2CD04 = 0x77767333
774KHz	I2CD04 = 0x77767722
1546KHz	I2CD04 = 0x77767711
2750KHz	I2CD04 = 0x77724711 (SCL Low = 200 ns, SCL High = 160 ns)

Below are some speed setting for PCLK = 66MHz (H-PLL 264MHz/4):

103KHz	I2CD04 = 0x77744366
412KHz	I2CD04 = 0x77744344
750KHz	I2CD04 = 0x77745733
825KHz	I2CD04 = 0x77744733
1650KHz	I2CD04 = 0x77744722
3000KHz	I2CD04 = 0x77735711 (SCL Low = 180 ns, SCL High = 150 ns)

32 PECI Controller

32.1 Overview

PECI Controller (PECI) supports PECI 1.1 and 2.0 protocols.

PECI totally implements 16 sets of 32-bit registers, which are listed below, to program the various supported functions. Each register has its own specific offset value, ranging from 0x00 to 0x3Ch, to derive its physical address location.

Base Address of PECI = 0x1E78_B000
Physical address of register = (Base address of PECI) + Offset

PECI00: Control Register
PECI04: Timing Negotiation Register
PECI08: Command Register
PECI0C: Read/Write Length Register
PECI10: Expected FCS Data Register
PECI14: Captured FCS Data Register
PECI18: Interrupt Register
PECI1C: Interrupt Status Register
PECI20: Write Data Register #0
PECI24: Write Data Register #1
PECI28: Write Data Register #2
PECI2C: Write Data Register #3
PECI30: Read Data Register #0
PECI34: Read Data Register #1
PECI38: Read Data Register #2
PECI3C: Read Data Register #3

32.2 Features

- Directly connected to APB bus
- Intel PECI 2.0/1.1 compliant
- Support up to 4 CPU and 2 domains per CPU
- Need an external analog comparator
- Shared with GPIO pins

32.3 Registers : Base Address = 0x1E78:B000

Offset: 00h		PECI00: Control Register Init = 0x000XXX00
Bit	**R/W**	**Description**
31:20		**Reserved (0)**

to next page

from previous page

19:16	RW	**Read sampling point selection** 0000: 0/16 0001: 1/16 0010: 2/16 0011: 3/16 ... 1111: 15/16 This register is only applied to Point Sampling mode. The whole period of a bit time will be divided into 16 time frames. This register will determine which time frame this controller will sample PECI signal for data read back. Usually in the middle of a bit time is the best sample point.
15:14	RW	**Reserved**
13:12	RW	**Read mode selection** 00: Point Sampling mode 01: Pulse Width Counting mode 10: Debugging mode 11: Invalid PECI supports two kinds of read mode selections. They are Point Sample mode and Pulse Width Counting Mode. Debugging mode is for debugging purpose. It can only be applied to ping command.
11:8	RW	**PECI clock divider** 0000: Divided by 1 0001: Divided by 2 0010: Divided by 4 0011: Divided by 8 ... 0111: Divided by 128 Others: Invalid This register will determine the operation frequency of PECI Controller. The input clock source is from 24MHz oscillator.
7	RW	**Inverse PECI output polarity** 0: Normal polarity 1: Inverse polarity
6	RW	**Inverse PECI input polarity** 0: Normal polarity 1: Inverse polarity
5	RW	**Enable bus contention** 0: Disable 1: Enable
4	RW	**Enable PECI** 0: Disable 1: Enable
3:2	RW	**Reserved**
1	RW	**Enable the automatic generation of AW FCS** 0: Disable 1: Enable Enabling this bit will have PECI Controller to automatically generate AW FCS (Assured Write Frame Check Sequence) based on the integrated hardware logic. When disabled, PECI Controller will generate AW FSC based on the checksum value calculated by S/W (pls. reference PECI10[31:24])

to next page

from previous page

0	RW	**Enable PECI clock** 0: Disable PECI clock 1: Enable PECI clock This register will stop or enable 24MHz clock source for power saving.

Offset: 04h	PECI04: Timing Negotiation Register	Init = X
Bit	**R/W**	**Description**
31:16		**Reserved (0)**
15:8	RW	**Message timing negotiation** This register will determine the period of message timing negotiation to be issued by PECI Controller. The unit of the programmed value is four times of PECI clock period.
7:0	RW	**Address timing negotiation** This register will determine the period of address timing negotiation to be issued by PECI Controller. The unit of the programmed value is four times of PECI clock period.

Offset: 08h	PECI08: Command Register	Init = 0
Bit	**R/W**	**Description**
31	R	**PECI pin monitoring** This bit can read back the signal status of PECI pin.
30:28		**Reserved (0)**
27:24	R	**PECI Controller state** 0000: PECI Controller is in idle state 0001: Fire state 0010: Initial address timing negotiation state 0011: Address timing negotiation state 0100: Address state 0101: Message timing negotiation state 0110: Write/read length state 0111: Write data state 1000: Reserved 1001: Write FCS state 1010: Read data state 1011: Read FCS state 1100: Stop state others: Reserved
23:1		**Reserved (0)**
0	RW	**Fire a PECI command** 0: No operation 1: Fire a PECI command

Offset: 0Ch	PECI0C: Read/Write Length Register	Init = X
Bit	**R/W**	**Description**
31	RW	**Enable AW FCS cycle** 0: Disable AW FCS cycle 1: Enable AW FCS cycle This register is only applied to PECI write commands. When enabled, PECI command will be with AW FCS cycle.
30:24		**Reserved (0)**

to next page

from previous page

23:16	RW	**Read data length (bytes)** This register determines the number of bytes to be read. The read back data will be stored in the following registers: PECI30, PECI34, PECI38, and PECI3C. Although this register is 8 bits, the maximum read data length is only 16 bytes.
15:8	RW	**Write data length (bytes)** This register determines the number of bytes to be written. The data to be written has to be pre-stored in the following registers: PECI20, PECI24, PECI28, and PECI2C.
7 :0	RW	**Target address** This register determines the 8-bit address of the PECI command to be fired.

Offset: 10h		**PECI10: Expected FCS Data Register** — **Init = X**
Bit	**R/W**	**Description**
31:24	RW	**Programmed AW FCS** This register is programmed by the checksum value calculated by S/W. When AW FCS is enabled and PECI00 [1] is "0", PECI Controller will generate AW FCS based on the value of this register.
23:16	R	**Expected read FCS** This register contains the FCS data generated by the internal hardware logic. This is for debugging purpose only.
15:8	R	**Expected auto AW FCS** This register contains the AW FCS data generated by the internal hardware logic. This is for debugging purpose only.
7 :0	R	**Expected write FCS** This register contains the FCS data generated by the internal hardware logic. This is for debugging purpose only.

Offset: 14h		**PECI14: Captured FCS Data Register** — **Init = X**
Bit	**R/W**	**Description**
31:24		**Reserved (0)**
23:16	R	**Captured FCS data from PECI data read command** This register contains the 8-bit FCS data captured by PECI Controller from a PECI data read command.
15:8		**Reserved (0)**
7 :0	R	**Captured FCS data from PECI write command** This register contains the 8-bit FCS data captured by PECI Controller from a PECI data write command.

Offset: 18h		**PECI18: Interrupt Register** — **Init = 0**
Bit	**R/W**	**Description**
31:30	RW	**Selection of timing negotiation result bit [1:0]** 00: 1st bit of address negotiation 01: 2nd bit of address negotiation 10: message negotiation 11: reserved
29:4		**Reserved (0)**
3	RW	**Enable PECI bus contention interrupt** 0: Disable 1: Enable

to next page

from previous page

2	RW	**Enable PECI write FCS bad interrupt** 0: Disable 1: Enable
1	RW	**Enable PECI write FCS abort interrupt** 0: Disable 1: Enable
0	RW	**Enable PECI command done interrupt** 0: Disable 1: Enable

Offset: 1Ch | **PECI1C: Interrupt Status Register** | **Init = 0xXXXX0000**

Bit	R/W	Description
31:16	R	**Timing negotiation result bit [15:0]**
15:4		**Reserved (0)**
3	RW	**PECI bus contention interrupt status** 0: No interrupt 1: Interrupt is pending Writing ”1” will clear this status.
2	RW	**PECI write FCS bad interrupt status** 0: No interrupt 1: Interrupt is pending Writing ”1” will clear this status.
1	RW	**PECI write FCS abort interrupt status** 0: No interrupt 1: Interrupt is pending Writing ”1” will clear this status.
0	RW	**PECI done interrupt status** 0: No interrupt 1: Interrupt is pending Writing ”1” will clear this status.

Offset: 20h | **PECI20: Write Data Register #0** | **Init = X**

Bit	R/W	Description
31:0	RW	**Write data bit [31:0]**

Offset: 24h | **PECI24: Write Data Register #1** | **Init = X**

Bit	R/W	Description
31:0	RW	**Write data bit [63:32]**

Offset: 28h | **PECI28: Write Data Register #2** | **Init = X**

Bit	R/W	Description
31:0	RW	**Write data bit [95:64]**

Offset: 2Ch | **PECI2C: Write Data Register #3** | **Init = X**

Bit	R/W	Description
31:0	RW	**Write data bit [127:96]**

Offset: 30h		PECI30: Read Data Register #0 Init = X
Bit	**R/W**	**Description**
31:0	R	**Read data bit [31:0]**

Offset: 34h		PECI34: Read Data Register #1 Init = X
Bit	**R/W**	**Description**
31:0	R	**Read data bit [63:32]**

Offset: 38h		PECI38: Read Data Register #2 Init = X
Bit	**R/W**	**Description**
31:0	R	**Read data bit [95:64]**

Offset: 3Ch		PECI3C: Read Data Register #3 Init = X
Bit	**R/W**	**Description**
31:0	R	**Read data bit [127:96]**

Part III

PCI Interface

33 PCI Slave Controller

33.1 Overview

PCI Slave Controller (PCIS) is a bus controller designed to bridge PCI bus and P-bus, which can directly communicate with VGA Controller, 2D Graphics Engine, SPI Host Controller, and P2A Bridge. PCIS total implements 13 PCI Configuration registers, which is listed below, to control the various functions supported by AST2050 / AST1100 .

PCIS00: Device and Vendor ID Register
PCIS04: Command and Status Register
PCIS08: Class and Revision ID Register
PCIS0C: Miscellaneous Register
PCIS10: Base Address 0 Register (for linear frame buffer)
PCIS14: Base Address 1 Register (for MMIO)
PCIS18: Base Address 2 Register (for relatable I/O)
PCIS2C: Subsystem ID Register
PCIS30: Expansion ROM Base Address Register
PCIS34: Capability Register
PCIS3C: Interrupt Register
PCIS40: PCI Power Management Capability Register
PCIS44: PCI Power Management Control and Status Register

33.2 Features

- Support 32-bit 33 MHz PCI bus interface with PCI 2.3 specification compliant
- Support big-endian & little-endian which can be enabled by register settings
- Support PME# & CLKRUN# control pins
- Support AD[31:0] bus reverse option for PCB layout optimization

Offset: 00h		**PCIS00: Device and Vendor ID Register** — **Init = 0x2000_1A03**
Bit	**R/W**	**Description**
31:16	R	**Device ID** The default setting of this register is **0x2000**, which is the device ID code being assigned for AST2050 / AST1100 . The device ID code of AST2050 / AST1100 is the same as AST2000. This arrangement is to make sure that AST2050 / AST1100 can directly run all the graphics display drivers developed for AST2000. The content of this register value can be changed by updating the corresponding register in SCU Controller, but its not recommended in normal cases.
15:0	R	**Vendor ID** The default setting of this register is **0x1A03** which is the vendor ID code being assigned for **ASPEED Technology Inc.** by **PCISIG**. The content of this register value can be changed by updating the corresponding register in SCU Controller, but its not recommended in normal cases.

Offset: 04h		PCIS04: Command and Status Register Init = 0x0210_0000
Bit	**R/W**	**Description**
31	R	**Detected parity error** AST2050 / AST1100 will not detect any parity errors; therefore, this bit will always return ”0”.
30	R	**Signaled system error** AST2050 / AST1100 will not signal any system errors; therefore, this bit will always return ”0”.
29	R	**Received master abort** AST2050 / AST1100 doesn’t play as a bus master; this register will always return ”0”.
28	R	**Received target abort** AST2050 / AST1100 doesn’t play as a bus master; this register will always return ”0”.
27	R	**Signaled target abort** AST2050 / AST1100 will not issue target abort; this register will always return ”0”.
26:25	R	**DEVSEL timing** AST2050 / AST1100 supports medium timing for DEVSEL signal; this register will always return ”01”.
24	R	**Master data parity error** AST2050 / AST1100 doesn’t play as a bus master; this register will always return ”0”.
23	R	**Fast back-to-back capable** AST2050 / AST1100 doesn’t support fast back-to-back; this register will always return ”0”.
22		**Reserved (0)**
21	R	**66 MHz capable** AST2050 / AST1100 supports 33MHz PCI bus running frequency; this register will always return ”0”.
20	R	**Capabilities list** AST2050 / AST1100 supports a linked list to implement PCI bus power management; this register will always return ”1”.
19	R	**Interrupt status** This read-only bit reflects the state of the only interrupt source generated by CRT Controller for detecting the end of vertical display enable. This is a legacy interrupt from VGA Controller. In most of the cases, this interrupt source will not be enabled.
18:11		**Reserved (0)**
10	RW	**Interrupt disable** 0: Enable interrupt 1: Disable interrupt
9	R	**Fast back-to-back enable** AST2050 / AST1100 doesn’t support fast back-to-back; this register will always return ”0”.
8	R	**SERR# enable** AST2050 / AST1100 wont signal any system errors; this bit will always return ”0”.
7		**Reserved (0)**
6	R	**Parity error response enable** AST2050 / AST1100 wont detect any parity errors; this bit will always return ”0”.
5	RW	**VGA palette snoop** AST2050 / AST1100 only provides a read/write bit for this register. But it wont impact any hardware behavior.

to next page

from previous page

4	RW	**Memory write and invalidate enable** Since AST2050 / AST1100 wont support this feature; this register will always return "0".
3	R	**Special cycles enable** Since AST2050 / AST1100 doesn't support PCI special cycles, this register will always return "0".
2	R	**Bus master enable** Since AST2050 / AST1100 doesn't support PCI bus master cycles, this register will always return "0".
1	RW	**Memory space access enable** 0: Disable memory space accesses 1: Enable memory space accesses This register will determine whether AST2050 / AST1100 will response to memory space accesses or not.
0	RW	**IO space access enable** 0: Disable I/O space accesses 1: Enable I/O space accesses This register will determine whether AST2050 / AST1100 will response to I/O space accesses or not.

Offset: 08h — PCIS08: Class and Revision ID Register — Init = 0x0X00_0010

Bit	R/W	Description
31:8	R	**Class code** When VGA Controller is enabled, AST2050 / AST1100 will always return "0x030000" as the class code for this register to claim that AST2050 / AST1100 is a VGA device. When VGA is disabled by an external trapping resistor, AST2050 / AST1100 will return "0x040000" as the class code of this register to claim that AST2050 / AST1100 is a video device. As a video device, AST2050 / AST1100 will not decode any VGA command cycles.
7 :0	R	**Revision ID** This register defines the revision ID of the current working silicon. It will change whenever a new revision is developed. The first revision ID of AST2050 / AST1100 is "00". The second revision ID of AST2050 / AST1100 is "10".

Offset: 0Ch — PCIS0C: Miscellaneous Register — Init = 0x0000_0000

Bit	R/W	Description
31	R	**BIST Capable** AST2050 / AST1100 doesn't support BIST; this register will always return "0".
30	R	**Start BIST** AST2050 / AST1100 doesn't support BIST; this register will always return "0".
29:28	R	**Start BIST** AST2050 / AST1100 doesn't support BIST; this register will always return "0".
27:24	R	**Completion code** AST2050 / AST1100 doesn't support BIST; this register will always return "0".
23:16	R	**Header type** This register will always return "0".
15:8	R	**Latency timer** AST2050 / AST1100 doesn't play as a bus master; this register will always return "0".
7 :0	RW	**Cache line size** This register will always return "0".

Offset: 10h		PCIS10: Base Address 0 Register Init = 0x0000_0000
Bit	**R/W**	**Description**
31:0	RW	**Base address 0 register** AST2050 / AST1100 will claim a re-locatable memory space (8MB /16MB /32MB /64MB) for linear frame buffer allocation by this base address register. The size of linear frame buffer will depend on the two corresponding trapping resistors.

Offset: 14h		PCIS14: Base Address 1 Register Init = 0x0000_0000
Bit	**R/W**	**Description**
31:0	RW	**Base address 1 register** AST2050 / AST1100 will claim a 128KB re-locatable I/O memory space allocation by this base address register. The first 64KB is for VGA I/O addressing space, the second 64KB is for P2A Bridge addressing space.

Offset: 18h		PCIS18: Base Address 2 Register Init = 0x0000_0001
Bit	**R/W**	**Description**
31:0	RW	**Base address 2 register** AST2050 / AST1100 will claim a 128B re-locatable I/O space allocation by this base address register. This addressing space is used for VGA legacy and extended I/O cycles.

Offset: 2Ch		PCIS2C: Subsystem ID Register Init = 0x2000_1A03
Bit	**R/W**	**Description**
31:16	RW	**Subsystem ID** This is a per-byte write once register. Once begin updated, this register cannot be modified again until next power-on. The default setting of this register is **0x2000**. Customer can modify this register if necessary.
15:0	RW	**Subsystem vendor ID** This is a per-byte write once register. Once begin updated, this register cannot be modified again until next power-on. The default setting of this register is **0x1A03**, which is following the vendor ID code of **ASPEED Technology Inc**. Customer can modify this register if necessary.

Offset: 30h		PCIS30: Expansion ROM Base Address Register Init = 0x0000_0000
Bit	**R/W**	**Description**
31:0	RW	**Expansion ROM base address** AST2050 / AST1100 will claim 64KB of memory space allocation for VGA BIOS. When VGA BIOS is merged with system BIOS, there will be no need to claim any ROM base address. Under such a condition, this base address claiming can be disabled by an external trapping resistor.

Offset: 34h		PCIS34: Capability Register Init = 0x0000_0040
Bit	**R/W**	**Description**
31:8		**Reserved (0)**
7 :0	R	**Capabilities pointer** This optional register is used to point to a linked list of new capabilities. AST2050 / AST1100 uses this register to point to 0x40 to implement PCI power management capability.

Offset: 3Ch		PCIS3C: Interrupt Register Init = 0x0000_0100
Bit	**R/W**	**Description**
31:24	RW	**Maximum latency** This register is used for specifying how often the device needs to gain access to the PCI bus.
23:16	RW	**Minimum grant** This register is used for specifying how long a burst period the device needs assuming a clock rate of 33 MHz.
15:8	RW	**Interrupt pin** AST2050 / AST1100 always returns 0x01 for this register to claim that the interrupt pin INTA# will be used by this device.
7 :0	RW	**Interrupt line** AST2050 / AST1100 provides an 8-bit read/write register to implement this function. The content of this register will not impact any hardware behavior.

Offset: 40h		PCIS: PCI Power Management Capability Register Init = 0xffc3_0001
Bit	**R/W**	**Description**
31:27	R	**PME support** AST2050 / AST1100 supports D0, D1, D2, D3hot and D3cold states. Therefore, this register will always return 0xF.
26	R	**D2 support** AST2050 / AST1100 supports D2 state; this register will always return ”1”.
25	R	**D1 support** AST2050 / AST1100 supports D1 state; this register will always return ”1”.
24:22	R	**Auxiliary current requirement** This register will always return ”111b”. It means that AST2050 / AST1100 requires 375mA from auxiliary current.
21	R	**Device specific initialization** AST2050 / AST1100 doesn’t need any special initializations. This register will always return ”0”.
20		**Reserved (0)**
19	R	**PME Clock** AST2050 / AST1100 doesn’t need to rely on PCI clock to generate PME#. This register will always return ”0”.
18:16	R	**Version** AST2050 / AST1100 complies with PCI Power Management Revision 1.2. Therefore, This register will always return ”011b”.
15:8	R	**Next item pointer** No next item pointer required. This register will always return ”0x00”.
7 :0	R	**ID** This register will always return ”0x01” to identify that the linked list item as being the PCI Power Management registers.

Offset: 44h		PCIS: PCI Power Management Control and Status Register Init = 0x0000_0000
Bit	**R/W**	**Description**
31:24	R	**Data register** This function is not implemented; this register always returns ”0x00”.
23	R	**Bus power and clock control enable** There is no secondary PCI bus; this register always returns ”0”.

to next page

from previous page

22	R	**B2/B3 support for D3hot** There is no secondary PCI bus; this register always returns ”0”.
21:16		**Reserved (0)**
15	RW	**PME Status** This bit is set when AST2050 / AST1100 would normally assert the PME signal independent of the state of the PME enable bit. Writing ”1” to this register will cause AST2050 / AST1100 to stop asserting the PME signal. Writing ”0” to this register has no effect.
14:13	R	**Data scale** AST2050 / AST1100 doesn’t implement Data register; this register always returns ”00b”.
12:9	RW	**Data select** AST2050 / AST1100 doesn’t implement Data register; this register always returns ”0000b”.
8	RW	**PME enable** 0: Disable PME assertion 1: Enable PME assertion
7 :4		**Reserved (0)**
3	R	**No soft reset** This register always returns ”0”.
2		**Reserved (0)**
1 :0	RW	**Power state** These two bits are used both to determine the current power state of AST2050 / AST1100 and to set AST2050 / AST1100 into a new power state. It will also impact CRT synchronization signals (HSYNC & VSYNC) and video DAC output. The definition of theses two bits is given below:

Bit[1:0]	State	Mode	HSYNC	VSYNC	DAC
00	D0	Active Mode	On	On	On
01	D1	Standby Mode	Off	On	Off
10	D2	Suspend Mode	On	Off	Off
11	D3	OFF Mode	Off	Off	Off

34 VGA Display Controller

34.1 Overview

VGA Display Controller (VGA) is one of the key modules integrated by AST2050 / AST1100 . The system bus interface adopted by VGA is 32-Bit PCI bus interface, which can be operating at 33MHz. VGA can be disabled by an external trapping resistor.

When VGA is enabled, the class code of "VGA Device" will be claimed by PCI configuration registers. When VGA is disabled, the class code of "Video Device" will be claimed for instead.

VGA is an in-band device which should be independent of ARM SOC system. Therefore, it can be reset only when either PCI bus reset or system power-on reset is asserted. VGA shares a portion of SDRAM memory for video frame buffer. The size of the shared frame buffer is determined by external trapping resistors. It will always occupy the highest portion of SDRAM memory. The initialization of SDRAM Controller is done by ARM SOC system. It should be finished well before host platform starting access video frame buffer.

VGA implements several groups of registers, which are listed below, to program the various supported functions. Each register has its own specific legacy address, and an offset value if available. AST2050 / AST1100 also provides memory-mapped I/O addressing mode for the need of advanced operating systems.

34.2 Features

- Fully IBM VGA compliant
- Maximum Display resolution: 1920x1200@60Hz with 200MHz video clock
- Integrate one deducted PLL for video clock generation which can be directly turned off by ARM CPU for power saving
- Support VESA DDC
- Support 64x64 hardware overlay cursor with mono and color formats
- RGB analog output
  - Integrate 200MHz triple DACs compliant with VESA monitor specification
  - Integrate 1.2V reference voltage generator
  - Need an external analog comparator for monitor sense
  - Support DAC power down function directly controlled by ARM CPU or host CPU
- Digital video outputs: 200MHz 24-bit single-edge DVO (3.3V digital signals)

34.3 Registers

VGAER: VGA Enable Register

R/W:3C3 — **Init = 00h**

Bit	Attr.	Description
7:1	RW	**Reserved (0)**
0	RW	**VGA enable** 0: Disable VGA 1: Enable VGA

VGAMR: Miscellaneous Output Register		
W:3C2 R:3CC		**Init = 00h**
Bit	**Attr.**	**Description**
7	RW	**Vertical sync polarity selection** 0: Select positive polarity 1: Select negative polarity
6	RW	**Horizontal sync polarity selection** 0: Select positive polarity 1: Select negative polarity
5	RW	**Page bit for odd/even modes** 0: Select lower page address 1: Select higher page address
4	RW	**Reserved**
3:2	RW	**Clock selection bit[1:0]** 00: Video clock frequency is 25.175MHz 01: Video clock frequency is 28.322MHz 1x: Video clock frequency is determined by the register programming for D-PLL
1	RW	**Enable video memory at VGA aperture** 0: Disable video memory address decoding 1: Enable video memory address decoding
0	RW	**I/O address selection** 0: Select 3Bx address decoding 1: Select 3Dx address decoding

VGAFCR: Feature Control Register		
W:3BA/3DA R:3CA		**Init = 00h**
Bit	**Attr.**	**Description**
7:4	RW	**Reserved (0)**
3	RW	**Feature control bit[2]**
2	RW	**Reserved**
1:0	RW	**Feature control bit[1:0]**

VGAIR0: Input Status Register #0		
R:3C2		**Init = 00h**
Bit	**Attr.**	**Description**
7	RW	**Vertical retrace interrupt flag**
6:5	RW	**Reserved (0)**
4	RW	**Video DAC comparator read back**
3:0	RW	**Reserved (0)**

VGAIR1: Input Status Register #1		
R:3BA/3DA		**Init = X1h**
Bit	**Attr.**	**Description**
7:6	RW	**Reserved (0)**

to next page

from previous page

5:4	RW	**Diagnostic bit[1:0]** 00: P2, P0 01: P5, P4 10: P3, P1 11: P7, P6 P7 ~ P0 are digital video output signals before RAMDAC controller.
3	RW	**Vertical retrace signal**
2:1	RW	**Reserved (0)**
0	RW	**Inversion of display enable signal** 0: During display enable period 1: Out of display enable period

VGAFBR0: Frame Buffer Segment Address Register #0		
R/W: 3CD		**Init = 00h**
Bit	**Attr.**	**Description**
7:4	RW	**Segment read address bit [3:0]**
3:0	RW	**Segment write address bit [3:0]**

VGAFBR1: Frame Buffer Segment Address Register #1		
R/W: 3CB		**Init = 00h**
Bit	**Attr.**	**Description**
7:4	RW	**Segment read address bit [7:4]**
3:0	RW	**Segment write address bit [7:4]**

34.4 Sequential Controller Registers

VGASRI: Sequential Controller Index Register		
R/W:3C4		**Init = 00h**
Bit	**Attr.**	**Description**
7:6	RW	**Reserved (0)**
5:0	RW	**Index register bit[5:0]**

VGASR0: Reset Register		
R/W:3C5 Index 00		**Init = 00h**
Bit	**Attr.**	**Description**
7:2	RW	**Reserved (0)**
1	RW	**Asynchronous reset (active low)** 0: Reset 1: No operation
0	RW	**Synchronous reset (active low)** 0: Reset 1: No operation

VGASR1: Clocking Mode Register		
R/W:3C5 Index:01		Init = 00h
Bit	**Attr.**	**Description**
7:6	RW	**Reserved**
5	RW	**Screen off** 0: Screen on 1: Screen off
4	RW	**Shift load by 4**
3	RW	**Divide video clock by 2**
2	RW	**Shift load by 2**
1	RW	**Reserved**
0	RW	**Select 8-dot period of character clock** 0: Select 9-dot character 1: Select 8-dot character

VGASR2: Map Mask Register		
R/W:3C5 Index:02		Init = 00h
Bit	**Attr.**	**Description**
7 :4	RW	**Reserved (0)**
3 :0	RW	**Enable memory write map [3:0]**

VGASR3: Character Map Selection Register		
R/W:3C5 Index:03		Init = 00h
Bit	**Attr.**	**Description**
7:6	RW	**Reserved (0)**
5	RW	**CG map A selection bit [2]**
4	RW	**CG map B selection bit [2]**
3:2	RW	**CG map A selection bit [1:0]**
1:0	RW	**CG map B selection bit [1:0]**

VGASR4: Character Map Selection Register		
R/W:3C5 Index:04		Init = 00h
Bit	**Attr.**	**Description**
7:4	RW	**Reserved (0)**
3	RW	**Enable Chain-4 mode**
2	RW	**Odd/even mode** 0: Odd/even mode 1: Sequential mode
1	RW	**Extended memory** 0: 64KB memory addressing mode 1: 256KB memory addressing mode
0	RW	**Reserved (0)**

34.5 CRT Controller Registers

VGACRI: CRT Controller Index Register		
R/W:3B4/3D4		Init = 00h
Bit	**Attr.**	**Description**
7:0	RW	Index register bit [7:0]

VGACR0: Horizontal Total Register		
R/W:3B5/3D5 Index:00		Init = XXh
Bit	**Attr.**	**Description**
7:0	RW	Horizontal total bit[7:0] (-5)

VGACR1: Horizontal Display Enable End Register		
R/W:3B5/3D5 Index:01		Init = XXh
Bit	**Attr.**	**Description**
7 :0	RW	Horizontal display enable bit[7:0] (-1)

VGACR2: Horizontal Blank Start Register		
R/W:3B5/3D5 Index:02		Init = XXh
Bit	**Attr.**	**Description**
7 :0	RW	Horizontal blank start bit[7:0]

VGACR3: Horizontal Blank End Register		
R/W:3B5/3D5 Index:03		Init = XXh
Bit	**Attr.**	**Description**
7	RW	Enable register read back for registers indexed from 10-11
6 :5	RW	Horizontal display enable skew bit[1:0]
4 :0	RW	Horizontal blank end bit[4:0]

VGACR4: Horizontal Retrace Start Register		
R/W:3B5/3D5 Index:04		Init = XXh
Bit	**Attr.**	**Description**
7 :0	RW	Horizontal retrace start bit [7:0]

VGACR5: Horizontal Retrace End Register		
R/W:3B5/3D5 Index:05		Init = XXh
Bit	**Attr.**	**Description**
7	RW	Horizontal blank end bit [5]
6 :5	RW	Horizontal retrace delay bit [1:0]
4 :0	RW	Horizontal retrace end bit [4:0]

VGACR6: Vertical Total Register		
R/W:3B5/3D5 Index:06		Init = XXh
Bit	**Attr.**	**Description**
7:0	RW	Vertical total bit [7:0]

VGACR7: Overflow Register		
R/W:3B5/3D5 Index:07		Init = XXh
Bit	**Attr.**	**Description**
7	RW	**Vertical retrace start bit [9]**
6	RW	**Vertical display enable end bit [9]**
5	RW	**Vertical total bit [9]**
4	RW	**Line compare bit [8]** This bit is out of the control by CRT register protection bit (Index 11, bit[7])
3	RW	**Vertical blank start bit [8]**
2	RW	**Vertical retrace start bit [8]**
1	RW	**Vertical display enable end bit [8]**
0	RW	**Vertical total bit [8]**

VGACR8: Preset Row Scan Register		
R/W:3B5/3D5 Index:08		Init = XXh
Bit	**Attr.**	**Description**
7	RW	**Reserved**
6:5	RW	**Byte panning bit[1:0]**
4:0	RW	**Preset row scan bit[4:0]**

VGACR9: Maximum Scan Line Register		
R/W:3B5/3D5 Index:09		Init = XXh
Bit	**Attr.**	**Description**
7	RW	**Enable double scan** Convert 200 scan lines to 400 scan lines
6	RW	**Line compare bit [9]**
5	RW	**Vertical blank bit [9]**
4:0	RW	**Maximum row scan bit [4:0]**

VGACRA: Cursor Start Register		
R/W:3B5/3D5 Index:0A		Init = XXh
Bit	**Attr.**	**Description**
7:6	RW	**Reserved (0)**
5	RW	**Cursor off**
4:0	RW	**Cursor start bit [4:0]**

VGACRB: Cursor End Register		
R/W:3B5/3D5 Index:0B		Init = XXh
Bit	**Attr.**	**Description**
7	RW	**Reserved (0)**
6:5	RW	**Cursor skew bit [1:0]**
4:0	RW	**Cursor end bit [4:0]**

VGACRC: Starting Address High Register		
R/W:3B5/3D5 Index:0C		**Init = XXh**
Bit	**Attr.**	**Description**
7:0	RW	**Starting address bit[15:8]**

VGACRD: Starting Address Low Register		
R/W:3B5/3D5 Index:0D		**Init = XXh**
Bit	**Attr.**	**Description**
7:0	RW	**Starting address bit[7:0]**

VGACRE: Cursor Location High Register		
R/W:3B5/3D5 Index:0E		**Init = XXh**
Bit	**Attr.**	**Description**
7:0	RW	**Cursor location bit[15:8]**

VGACRF: Cursor Location Low Register		
R/W:3B5/3D5 Index:0F		**Init = XXh**
Bit	**Attr.**	**Description**
7:0	RW	**Cursor location bit[7:0]**

VGACR10: Vertical Retrace Start Register		
R/W:3B5/3D5 Index:10		**Init = XXh**
Bit	**Attr.**	**Description**
7:0	RW	**Vertical retrace start bit[7:0]**

VGACR11: Vertical Retrace End Register		
R/W:3B5/3D5 Index:11		**Init = 00h**
Bit	**Attr.**	**Description**
7	RW	**Protect CRT registers from index 00 to index 07** Index 07[4] is the only exception
6	RW	**Reserved** This bit is for register read/write only
5	RW	**Disable vertical interrupt**
4	RW	**Clear vertical interrupt flag** 0: Clear vertical interrupt flag 1: No operation
3:0	RW	**Vertical retrace end bit[3:0]**

VGACR12: Vertical Display Enable End Register		
R/W:3B5/3D5 Index:12		**Init = XXh**
Bit	**Attr.**	**Description**
7:0	RW	**Vertical display enable end bit [7:0]**

VGACR13: Offset Register		
R/W:3B5/3D5 Index:13		Init = 00h
Bit	Attr.	Description
7:0	RW	Offset bit [7:0]

VGACR14: Underline Location Register		
R/W:3B5/3D5 Index:14		Init = XXh
Bit	Attr.	Description
7	RW	Reserved (0)
6	RW	Select double word mode
5	RW	Select count-by-4 mode This function is not implemented
4:0	RW	Underline location bit [4:0]

VGACR15: Vertical Blank Start Register		
R/W:3B5/3D5 Index:15		Init = XXh
Bit	Attr.	Description
7:0	RW	Vertical blank start [7:0]

VGACR16: Vertical Blank End Register		
R/W:3B5/3D5 Index:16		Init = XXh
Bit	Attr.	Description
7:0	RW	Vertical blank end [7:0]

VGACR17: Mode Control Register		
R/W:3B5/3D5 Index:17		Init = 00h
Bit	Attr.	Description
7	RW	Hardware reset (active low)
6	RW	Select byte mode
5	RW	Address wrap enable This function is not implemented
4	RW	Reserved (0)
3	RW	Select count-by-2 mode This function is not implemented
2	RW	Horizontal retrace selection This function is not implemented
1	RW	Replace MA14 by RA1 (active low)
0	RW	Replace MA13 by RA0 (active low)

VGACR18: Line Compare Register		
R/W:3B5/3D5 Index:18		Init = XXh
Bit	Attr.	Description
7:0	RW	Line compare bit [7:0]

VGACR1E: Graphics Latched Data 0 Register		
R:3B5/3D5 Index:1E		Init = XXh
Bit	**Attr.**	**Description**
7:2	R	Reserved
1	R	Attribute controller register index toggle bit
0	R	Reserved

VGACR1F: Graphics Latched Data 1 Register		
R:3B5/3D5 Index:1F		Init = 00h
Bit	**Attr.**	**Description**
7:6	R	Reserved
5:0	R	Attribute controller register index bit [5:0]

VGACR22: Graphics Latched Data 2 Register		
R:3B5/3D5 Index:22		Init = XXh
Bit	**Attr.**	**Description**
7:0	R	Graphics latched data bit[7:0]

34.6 Graphics Controller Registers

VGAGRI: Graphics Controller Index Register		
R/W:3CE		Init = 00h
Bit	**Attr.**	**Description**
7:4	RW	Reserved (0)
3:0	RW	Index register bit [3:0]

VGAGR0: Set/Reset Map Register		
R/W:3CF Index:00		Init = 00h
Bit	**Attr.**	**Description**
7:4	RW	Reserved (0)
3:0	RW	Set/reset map bit [3:0]

VGAGR1: Enable Set/Reset Map Register		
R/W:3CF Index:01		Init = 00h
Bit	**Attr.**	**Description**
7:4	RW	Reserved (0)
3:0	RW	Enable set/reset map bit [3:0]

VGAGR2: Color Compare Register		
R/W:3CF Index:02		Init = 00h
Bit	**Attr.**	**Description**
7:4	RW	Reserved (0)
3:0	RW	Color compare map bit [3:0]

VGAGR3: Data Rotate Register		
R/W:3CF Index:03		Init = 00h
Bit	**Attr.**	**Description**
7:5	RW	**Reserved (0)**
4:3	RW	**Function selection bit [1:0]**
2:0	RW	**Data rotate bit [3:0]**

VGAGR4: Read Map Selection Register		
R/W:3CF Index:04		Init = 00h
Bit	**Attr.**	**Description**
7:2	RW	**Reserved (0)**
1:0	RW	**Read map selection bit [1:0]**

VGAGR5: Mode Register		
R/W:3CF Index:05		Init = 00h
Bit	**Attr.**	**Description**
7	RW	**Reserved (0)**
6	RW	**Enable shift mode for graphics display mode 13**
5	RW	**Enable shift mode for graphics display mode 4 and mode 5**
4	RW	**Enable odd/even mode**
3	RW	**Read mode selection** 0: Select normal read mode 1: Select color compare read mode
2	RW	**Reserved (0)**
1:0	RW	**Write mode selection bit [1:0]**

VGAGR6: Miscellaneous Register		
R/W:3CF Index:06		Init = 00h
Bit	**Attr.**	**Description**
7:4	RW	**Reserved (0)**
3:2	RW	**Memory addressing space selection bit [1:0]** 00: A000H/128KB 01: A000H/64KB 10: B000H/32KB 11: B800H/32KB
1	RW	**Chain odd/even plan enable**
0	RW	**Select graphics mode** 0: text mode 1: graphics mode

VGAGR7: Color Don't Care Register		
R/W:3CF Index:07		Init = 00h
Bit	**Attr.**	**Description**
7:4	RW	**Reserved (0)**
3:0	RW	**Color don't care bit[3:0]**

VGAGR8: Bit Mask Register		
R/W:3CF Index:08		**Init = 00h**
Bit	**Attr.**	**Description**
7:0	RW	**Bit mask bit [7:0]**

34.7 Attribute Controller Registers

VGAARI: Attribute Controller Index Register		
R:3C1 W:3C0		**Init = 00h**
Bit	**Attr.**	**Description**
7:6	RW	**Reserved (0)**
5	RW	**Pallet address source selection** 0: Address source is from register read/write address issued by CPU 1: Address source is from graphics streaming data
4:0	RW	**Index register bit [4:0]**

VGAAR0–VGAARF: Pallet Register 00 ~ 0F		
R:3C1 W:3C0 Index:00–0F		**Init = XXh**
Bit	**Attr.**	**Description**
7:6	RW	**Reserved (0)**
5:0	RW	**Pallet data bit [5:0]** There are total 16 sets of 5-bit palette registers. Their index number is from 00h to 0Fh. The address source of the pallet is determined by Attribute Controller Index Register bit [5].

VGAAR10: Mode Control Register		
R:3C1 W:3C0 Index:10		**Init = 00h**
Bit	**Attr.**	**Description**
7	RW	**Internal palette size selection** 0: Select 6 bits per pixel 1: Select 4 bits per pixel (cascaded with Index 14[1:0])
6	RW	**Pixel width selection** For Mode 13 only
5	RW	**Pixel panning compatibility** 0: Pixel panning will be applied to both screens (before and after line compare) 1: Pixel panning will only be applied to screen before line compare
4	RW	**Reserved (0)**
3	RW	**Enable blink mode**
2	RW	**Enable line graphics extension for ASCII codes from 0xC0 to 0xDF**
1	RW	**Select monochrome display mode**
0	RW	**Select graphics mode** 0: Select text mode 1: Select graphics mode

VGAAR11: Boarder Color Register		
R:3C1 W:3C0 Index:11		Init = XXh
Bit	**Attr.**	**Description**
7:0	RW	**Boarder color bit [7:0]**

VGAAR12: Color Plan Enable Register		
R:3C1 W:3C0 Index:12		Init = XXh
Bit	**Attr.**	**Description**
7:6	RW	**Reserved (0)**
5:4	RW	**Video status multiplexing bit [1:0]**
3:0	RW	**Color plan enable bit [3:0]**

VGAAR13: Horizontal Pixel Panning Register		
R:3C1 W:3C0 Index:13		Init = 0Xh
Bit	**Attr.**	**Description**
7:4	RW	**Reserved (0)**
3:0	RW	**Horizontal pixel panning bit [3:0]**

VGAAR14: Color Select Register		
R:3C1 W:3C0 Index:14		Init = 0Xh
Bit	**Attr.**	**Description**
7:4	RW	**Reserved (0)**
3:0	RW	**Color selection bit [3:0]**

34.8 RAMDAC Registers

VGAPMR: RAMDAC Pixel Mask Register		
R/W:3C6		Init = FFh
Bit	**Attr.**	**Description**
7:0	W	**Pixel mask bit [7:0]**

VGADSR: RAMDAC Status Register		
R:3C7		Init = 00h
Bit	**Attr.**	**Description**
7:2	R	**Reserved (0)**
1:0	R	**Status bit [1:0]**

VGADRR: RAMDAC Read Mode Address Register		
W:3C7		Init = 00h
Bit	**Attr.**	**Description**
7:0	W	**Read mode address bit [7:0]**

VGADWR: RAMDAC Write Mode Address Register		
R/W:3C8		Init = 00h
Bit	**Attr.**	**Description**
7:0	RW	**Write mode address bit [7:0]**

VGAPDR: RAMDAC Pallet Data Register		
R/W:3C9		Init = 00h
Bit	**Attr.**	**Description**
7:0	RW	**Pallet data bit [7:0]**

34.9 Extended CRT Registers

Index Range	3	2	1	0
Index 83 - 80	VGA Scratch Register			Password
Index 87 - 84	VGA Scratch Register			
Index 8B - 88	VGA Scratch Register			
Index 8F - 8C	VGA Scratch Register			
Index 93 - 90	VGA Scratch Register			
Index 97 - 94	VGA Scratch Register			
Index 9B - 98	VGA Scratch Register			
Index 9F - 9C	VGA Scratch Register			
Index A3 - A0	Color Mode	PCI Bus Control		
Index A7 - A4	CRT Threshold		Segment Adr	Misc Control
Index AB - A8	Power-On Trapping		RAMDAC Control	
Index AF - AC	Starting Overflow	Vertical Overflow	Horizontal Overflow	
Index B3 - B0	CRT Counter Read Back			Offset Overflow
Index B7 - B4	DDC Control	Power Control	Reserved (0)	
Index BB - B8	PLL Overflow	RGB CRC Signature Read Back		
Index BF - BC	28MHz PLL		25MHz PLL	
Index C3 - C0	Hardware Cursor Offset		Video PLL	
Index C7 - C4	Hardware Cursor Y Position		Hardware Cursor X Position	
Index CB - C8	Cursor Mode	Hardware Cursor Pattern Address		
Index CF - CC	Reserved			
Index D3 - D0	SOC Scratch Register Read Back			
Index D7 - D4	SOC Scratch Register Read Back			

VGACR80: Password Register

R/W:3B5/3D5 Index:80 MMIO:Base+80 — **Init = 00h**

Bit	Attr.	Description
7:0	RW	**Password bit [7:0]** Password: A8h

VGACR81–9E: Scratch Register #1 ～ #30

R/W:3B5/3D5 Index:81–9E MMIO:Base+81–9E — **Init = XXh**

Bit	Attr.	Description
7:0	RW	**Scratch register bit [7:0]** Only for the usage of VGA BIOS and Display Drivers

VGACR9F: Scratch Register #31

R/W:3B5/3D5 Index:9F MMIO:Base+9F — **Init = XXh**

Bit	Attr.	Description
7:6	R	**Indicate the PCI power state D0 ～ D3** Map to PCIS44 bit[1:0]

to next page

from previous page

5	R	**Indicate the Pallet address source selection** Map to VGAARI bit[5]
4	R	**Indicate the Pixel Mask Status** Map to the Logical-OR result of VGAPMR bit[7:0]
3	R	**Indicate the Reset Status of VGA** Map to VGACR17 bit[7]
2	R	**Indicate the Status of Screen Display** Map to VGASR1 bit[5]
1	R	**Indicate the Reset Status of VGA Controller** Map to the Logical-AND result of VGASR0 bit[0] and bit[1]
0	R	**VGA Enable Status Register** Map to VGAER:3C3 bit[0]

VGACRA0: PCI Control Register #1

R/W:3B5/3D5 Index:A0 MMIO:Base+A0 — Init = 00h

Bit	Attr.	Description
7	RW	**Reserved**
6	RW	**Enable video memory access by 32-bit china-4 mode** This bit is for graphics mode only
5	RW	**Enable linear extended memory access (> 256KB)**
4	RW	**Enable extended segmented memory address (> 256KB)**
3	RW	**Enable burst memory read**
2	RW	**Enable burst memory write**
1	RW	**Enable read-ahead cache**
0	RW	**Enable post-write buffer**

VGACRA1: PCI Control Register #2

R/W:3B5/3D5 Index:A1 MMIO:Base+A1 — Init = 00h

Bit	Attr.	Description
7:4	RW	**Reserved**
3	RW	**Disable re-locatable I/O-mapped VGA I/O address decoding**
2	RW	**Enable re-locatable memory-mapped VGA I/O address decoding**
1	RW	**Disable standard VGA I/O address decoding**
0	RW	**Disable standard VGA memory address (0xA000~0xBFFF) decoding**

VGACRA2: PCI Control Register #3

R/W:3B5/3D5 Index:A2 MMIO:Base+A2 — Init = 00h

Bit	Attr.	Description
7	RW	**Enable big-endian mode**
6	RW	**Enable 16-bit big-endian mode** 0: 32-bit 1: 16-bit
5	RW	**Reserved**
4	RW	**Enable PCI retry for I/O cycles while memory post-write buffer is not empty**

to next page

from previous page

3	RW	**Enable PCI retry for memory write cycles**
2	RW	**Enable PCI retry for memory read cycles**
1:0	RW	**Reserved**

VGACRA3: Enhanced Color Mode Register

R/W:3B5/3D5 Index:A3 MMIO:Base+A3 **Init = 00h**

Bit	Attr.	Description
7	RW	**Enable DVO interface**
6	RW	**Enable dual-edge DVO interface**
5:4	RW	**Reserved**
3	RW	**Enable 32-BPP true color display mode (ARGB:8888)**
2	RW	**Enable 16-BPP high color display mode (RGB:565)**
1	RW	**Enable 15-BPP high color display mode (RGB:555)**
0	RW	**Enable enhanced 256 color display mode**

VGACRA4: Misc. Control Register

R/W:3B5/3D5 Index:A4 MMIO:Base+A4 **Init = 00h**

Bit	Attr.	Description
7	RW	**Software reset 2D engine**
6	RW	**Trigger bit of VGA interrupt to BMC**
5	RW	**Enable Sub-System ID and Sub-Vendor ID write cycles**
4	RW	**Enable VGA BIOS flash write**
3:2	RW	**2D Engine clock source selection** 00: MCLK 01: ~MCLK (inverted clock phase) 10: Reserved 11: Reserved
1	RW	**Enable clock throttling for 2D Engine** When 2D Engine is in idle state, its clock will be automatically slowed down to 1/16 for power saving. When receiving a new command, 2D Engine will speed up at full speed automatically.
0	RW	**Enable 2D Engine Clock** 0: Stop 1: Enable

VGACRA5: Segmented Memory Address Overflow Register

R/W:3B5/3D5 Index:A5 MMIO:Base+A5 **Init = 00h**

Bit	Attr.	Description
7:6	RW	**Reserved**
5:4	RW	**Segmented memory read address bit [9:8]**
3:2	RW	**Reserved**
1:0	RW	**Segmented memory write address bit [9:8]**

VGACRA6: CRT Request Threshold Low Register		
R/W:3B5/3D5 Index:A6 MMIO:Base+A6		Init = 00h
Bit	**Attr.**	**Description**
7:6	RW	**Reserved**
5:0	RW	**CRT request threshold low bit [5:0]**

VGACRA7: CRT Request Threshold High Register		
R/W:3B5/3D5 Index:A7 MMIO:Base+A7		Init = 00h
Bit	**Attr.**	**Description**
7:6	RW	**Reserved**
5:0	RW	**CRT memory request threshold high bit [5:0]**

VGACRA8: RAMDAC Control Register		
R/W:3B5/3D5 Index:A8 MMIO:Base+A8		Init = 00h
Bit	**Attr.**	**Description**
7	RW	**Reserved**
6	RW	**Enable RAMDAC test mode for monitor sense application**
5	RW	**Reserved**
4	RW	**Disable RAMDAC mask function**
3	RW	**Reserved**
2	RW	**Protect palette/gamma RAM from write cycles**
1	RW	**Enable 24-bit gamma correction RAM**
0	RW	**Reserved**

VGACRA9: RAMDAC Test Pattern Register		
R/W:3B5/3D5 Index:A9 MMIO:Base+A9		Init = 00h
Bit	**Attr.**	**Description**
7:0	RW	**RAMDAC test pattern bit[7:0]**

VGACRAA: Power-On Trapping Status Register #1		
R/W:3B5/3D5 Index:AA MMIO:Base+AA		Init = X
Bit	**Attr.**	**Description**
7	R	**CPU clock frequency selection bit[0]**
6	R	**Bypass Clock Mode** 0: Normal clock operation 1: Bypass all clock with test clock input (Test Mode)
5:4	R	**ARM CPU boot code selection** 00: Boot from ROMCS0#, NOR flash memory 01: Boot from ROMCS1#, NAND flash memory 10: Boot from ROMCS2#, SPI flash memory 11: Disable ARM CPU operation
3	R	**Enable VGA BIOS ROM** 0: Disable VGA BIOS ROM 1: Enable VGA BIOS ROM

to next page

from previous page

2	R	**PCI interface selection** 0: Enable PCI salve controller 1: Enable PCI host controller
1:0	R	**Total VGA memory size setting** 00: 8MB 01: 16MB 10: 32MB 11: 64MB

VGACRAB: Power-On Trapping Status Register #2

R/W:3B5/3D5 Index:AB MMIO:Base+AB **Init = 00h**

Bit	Attr.	Description
7:6	R	**Chip ID** 11: AST2100 10: AST2050/AST1100
5	R	**Bypass all PLL modules**
4	R	**PCI class code selection** 0: A video device is claimed in PCI class code register 1: A VGA device is claimed in PCI class code register
3	R	**PCI VGA Config Prefetch status** 0: Prefetch bit = 0 1: prefetch bit = 1
2	R	**PCI AD Bus Order Swap** 0: Disable swap 1: Enable swap
1:0	R	**CPU clock frequency selection bit[2:1]** 000: Select 266 MHz 001: Select 233 MHz 010: Select 200 MHz 011: Select 166 MHz 100: Select 133 MHz 101: Select 100 MHz 110: Select 300 MHz 111: Select 24 MHz (by enabling H-PLL bypass mode)

VGACRAC: Extended Horizontal Overflow Register #1

R/W:3B5/3D5 Index:AC MMIO:Base+AC **Init = 00h**

Bit	Attr.	Description
7	RW	**Reserved**
6	RW	**Horizontal retrace start bit [8]**
5	RW	**Reserved**
4	RW	**Horizontal blank start bit [8]**
3	RW	**Reserved**
2	RW	**Horizontal display enable end bit [8]**
1	RW	**Reserved**
0	RW	**Horizontal total bit [8]**

VGACRAD: Extended Horizontal Overflow Register #2		
R/W:3B5/3D5 Index:AD MMIO:Base+AD		**Init = 00h**
Bit	**Attr.**	**Description**
7	RW	**Reserved**
6:4	RW	**Horizontal retrace skew bit [2:0]**
3	RW	**Reserved**
2	RW	**Horizontal retrace end bit [5]**
1	RW	**Reserved**
0	RW	**Horizontal blank end bit [6]**

VGACRAE: Extended Vertical Overflow Register		
R/W:3B5/3D5 Index:AE MMIO:Base+AE		**Init = 00h**
Bit	**Attr.**	**Description**
7	RW	**Disable line compare**
6:5	RW	**Vertical retrace end bit [5:4]**
4	RW	**Vertical blank end bit [8]**
3	RW	**Vertical retrace start bit [10]**
2	RW	**Vertical blank start bit [10]**
1	RW	**Vertical display enable end bit [10]**
0	RW	**Vertical total bit [10]**

VGACRAF: Extended CRT Starting Address Register		
R/W:3B5/3D5 Index:AF MMIO:Base+AF		**Init = 00h**
Bit	**Attr.**	**Description**
7:0	RW	**CRT starting address bit [23:16]**

VGACRB0: Extended CRT Offset Register		
R/W:3B5/3D5 Index:B0 MMIO:Base+B0		**Init = 00h**
Bit	**Attr.**	**Description**
7:6	RW	**Reserved**
5:0	RW	**Offset bit[13:8]**

VGACRB1: Horizontal Counter read Back Register		
R:3B5/3D5 Index:B1 MMIO:Base+B1		**Init = 00h**
Bit	**Attr.**	**Description**
7:0	RW	**Horizontal counter read back bit[7:0]**

VGACRB2: Vertical Counter read Back Register		
R:3B5/3D5 Index:B2 MMIO:Base+B2		**Init = 00h**
Bit	**Attr.**	**Description**
7:0	RW	**Vertical counter read back bit[7:0]**

VGACRB3: CRT Counter read Back Overflow Register		
R/W:3B5/3D5 Index:B3 MMIO:Base+B3		Init = 00h
Bit	**Attr.**	**Description**
7:5	RW	**Reserved (0)**
4	RW	**Horizontal counter read back bit[8]**
3	RW	**Reserved (0)**
2:0	RW	**Vertical counter read back bit[10:8]**

VGACRB6: Power Management Register		
R/W:3B5/3D5 Index:B6 MMIO:Base+B6		Init = 00h
Bit	**Attr.**	**Description**
7:5	RW	**Reserved**
4	RW	**Enable bypass mode for video PLL**
3	RW	**Power down video PLL**
2	RW	**Power on RAMDAC** 0: RAMDAC is power down 1: RAMDAC is power on
1	RW	**Enable VSYNC off**
0	RW	**Enable HSYNC off**

VGACRB7: DDC Control Register		
R/W:3B5/3D5 Index:B7 MMIO:Base+B7		Init = 00h
Bit	**Attr.**	**Description**
7	RW	**Status of CRC signature generation** 0: Invalid (still in progress or never triggered) 1: Valid (finished)
6	RW	**Trig CRC signature generation** 0: No operation 1: Trig CRC signature generation CRC signature generation will take at least one frame of cycle time to finish the task. S/W needs to poll the status of CRC signature generation before reading back RGB signature data.
5	RW	**DDC data input**
4	RW	**DDC clock input**
3	RW	**DDC data output**
2	RW	**Enable DDC data output buffer**
1	RW	**DDC clock output**
0	RW	**Enable DDC clock output buffer**

VGACRB8: Blue CRC Signature Read Back Register		
R/W:3B5/3D5 Index:B8 MMIO:Base+B8		Init = FCh
Bit	**Attr.**	**Description**
7	RW	**Blue CRC signature read back bit [7:0]**

VGACRB9: Green CRC Signature Read Back Register		
R/W:3B5/3D5 Index:B9 MMIO:Base+B9		Init = FCh
Bit	**Attr.**	**Description**
7	RW	**Green CRC signature read back bit [7:0]**

VGACRBA: Red CRC Signature Read Back Register		
R/W:3B5/3D5 Index:BA MMIO:Base+BA		Init = FCh
Bit	**Attr.**	**Description**
7	RW	**Red CRC signature read back bit [7:0]**

VGACRBB: PLL Overflow Register		
R/W:3B5/3D5 Index:BB MMIO:Base+BB		Init = 1Fh
Bit	**Attr.**	**Description**
7:6	RW	**Reserved**
5:4	RW	**Video PLL extended post divider** 00: 1/1 01: 1/2 10: 1/2 11: 1/4
3:2	RW	**28.322MHz PLL extended post divider bit [1:0]** 00: 1/1 01: 1/2 10: 1/2 11: 1/4 The content of this register will only determine the clock frequency of D-PLL when 28.322MHz is selected by legacy register.
1:0	RW	**25.175MHz PLL extended post divider bit [1:0]** 00: 1/1 01: 1/2 10: 1/2 11: 1/4 The content of this register will only determine the clock frequency of D-PLL when 25.175MHz is selected by legacy register.

VGACRBC: 25.175MHz PLL Setting Register		
R/W:3B5/3D5 Index:BC MMIO:Base+BC		Init = E9h
Bit	**Attr.**	**Description**
7:0	RW	**Video PLL numerator bit[7:0]**

VGACRBD: 25.175MHz PLL Setting Register		
R/W:3B5/3D5 Index:BD MMIO:Base+BD		Init = 65h
Bit	**Attr.**	**Description**
7	RW	**Reserved**

to next page

from previous page

6:5	RW	**Video PLL post divider bit [1:0]** 00: 1/1 01: 1/2 10: 1/2 11: 1/4 The content of this register will only determine the clock frequency of D-PLL when 25.175MHz is selected by legacy register.
4:0	RW	**Video PLL de-numerator bit [4:0]**

VGACRBE: 28.322MHz PLL Setting Register

R/W:3B5/3D5 Index:BE MMIO:Base+BE **Init = 95h**

Bit	Attr.	Description
7:0	RW	**Video PLL numerator bit[7:0]**

VGACRBF: 28.322MHz PLL Setting Register

R/W:3B5/3D5 Index:BF MMIO:Base+BF **Init = 62h**

Bit	Attr.	Description
7	RW	**Reserved**
6:5	RW	**Video PLL post divider bit [1:0]** 00: 1/1 01: 1/2 10: 1/2 11: 1/4 The content of this register will only determine the clock frequency of D-PLL when 28.322MHz is selected by legacy register.
4:0	RW	**Video PLL de-numerator bit[4:0]**

VGACRC0: Video PLL Setting Register

R/W:3B5/3D5 Index:C0 MMIO:Base+C0 **Init = 4Eh**

Bit	Attr.	Description
7:0	RW	**Video PLL numerator bit [7:0]**

VGACRC1: Video PLL Setting Register

R/W:3B5/3D5 Index:C1 MMIO:Base+C1 **Init = 61h**

Bit	Attr.	Description
7	RW	**Reserved**
6:5	RW	**Video PLL post divider bit [1:0]** 00: 1/1 01: 1/2 10: 1/2 11: 1/4
4:0	RW	**Video PLL de-numerator bit [4:0]**

VGACRC2: H/W Cursor X Position Offset Register

R/W:3B5/3D5 Index:C2 MMIO:Base+C2 — Init = 00h

Bit	Attr.	Description
7:6	RW	Reserved
5:0	RW	H/W cursor X position offset bit [5:0]

VGACRC3: H/W Cursor Y Position Offset Register

R/W:3B5/3D5 Index:C3 MMIO:Base+C3 — Init = 00h

Bit	Attr.	Description
7:6	RW	Reserved
5:0	RW	H/W cursor Y position offset bit [5:0]

VGACRC4: H/W Cursor X Position Register #1

R/W:3B5/3D5 Index:C4 MMIO:Base+C4 — Init = 00h

Bit	Attr.	Description
7:0	RW	H/W cursor X position bit [7:0]

VGACRC5: H/W Cursor X Position Register #2

R/W:3B5/3D5 Index:C5 MMIO:Base+C5 — Init = 00h

Bit	Attr.	Description
7:4	RW	Reserved
3:0	RW	H/W cursor X position bit [11:8]

VGACRC6: H/W Cursor Y Position Register #1

R/W:3B5/3D5 Index:C6 MMIO:Base+C6 — Init = 00h

Bit	Attr.	Description
7:0	RW	H/W cursor Y position bit [7:0]

VGACRC7: H/W Cursor Y Position Register #2

R/W:3B5/3D5 Index:C7 MMIO:Base+C7 — Init = 00h

Bit	Attr.	Description
7:3	RW	Reserved
2:0	RW	H/W cursor Y position bit [10:8]

VGACRC8: H/W Cursor Pattern Address Register #1

R/W:3B5/3D5 Index:C8 MMIO:Base+C8 — Init = 00h

Bit	Attr.	Description
7:0	RW	**Cursor pattern memory address bit [11:4]** The address must be 16-byte aligned. Therefore address bit [3:0] is always ”0”.

VGACRC9: H/W Cursor Pattern Address Register #2		
R/W:3B5/3D5 Index:C9 MMIO:Base+C9		Init = 00h
Bit	**Attr.**	**Description**
7:0	RW	**Cursor pattern memory address bit [19:12]**

VGACRCA: H/W Cursor Pattern Address Register #3		
R/W:3B5/3D5 Index:CA MMIO:Base+CA		Init = 00h
Bit	**Attr.**	**Description**
7:0	RW	**Cursor pattern memory address bit [27:20]**

VGACRCB: H/W Cursor Control Register		
R/W:3B5/3D5 Index:CB MMIO:Base+CB		Init = 00h
Bit	**Attr.**	**Description**
7:2	RW	**Reserved**
1	RW	**Enable H/W cursor** 0: Disable H/W cursor display 1: Enable H/W cursor display
0	RW	**H/W cursor type selection** 0: Select 2-BPP 1: Select 16-BPP (ARGB:1555)

VGACRCC–D7: Scratch Register #32 ~ #43		
R/W:3B5/3D5 Index:CC–D7 MMIO:Base+CC–D7		Init = XXh
Bit	**Attr.**	**Description**
7:0	RW	**Scratch register bit [7:0]**

35 2D Graphics Engine

35.1 Overview

2D Graphics Engine supports a variety of 2D graphics commands to accelerate rendering performance. The maximum running frequency is 266MHz. The highest throughput this engine can achieve is 64 bits of data output per clock. This throughput number can be converted to 8 pixels per clock for 256 color modes, 4 pixels per clock for high color modes, and 2 pixels per clock for true color modes. AST2050 / AST1100 supports the following commands:

- **BitBLT operations**: logic operations among source, destination, pattern, and mask
- **Font expansion**: expanding monochrome bitmaps to color bitmaps
- **Line drawing**: rendering lines with style option

2D Graphics Engine implements 83 sets of 32-bit registers, which are listed below, to program the various supported functions. Some of the registers have different definitions for different 2D graphics commands, especially for BitBLT command and line drawing command. All these register can be access through PCI memory-mapped I/O cycles regarding to the following formula.

Base address of 2D Graphics Engine = (PCIS14: Base Address 1 Register) + 0x8000

Register address of 2D Graphics Engine = (Base address of 2D Graphics Engine) + Offset

35.2 Features

- Directly access data through M-Bus
- High performance pipelined one-cycle 64-bit 2D graphics engine
- 2D engine commands
  - BitBlt Rectangle Fill
  - BitBlt Pattern Fill
  - BitBlt Rectangle Copy from Source to Destination
  - Support 256 Raster Operations
  - Integrate 8x8 Pattern Registers
  - Integrate 8x8 Mask Registers
  - Support Rectangle Clip
  - Support Color Expansion
  - Support Enhanced Color Expansion
  - Support Line Drawing with Style Pattern
- Programmable 256K/512K/1M/2M off-screen command buffer
- Integrate 16 stages of hardware command queue for 2D command pre-fetch from off-screen memory space of frame buffer
- Integrate 64x16 source buffer and 64x16 destination buffer to improve 2D engine performance

35.3 2D Engine Registers

Offset: 00h		GER00: Base Address of Source Buffer Register — Init = X
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:3	RW	**Base address of source buffer bit [27:3]** The address should be always 8-bytes aligned. Therefore, bit [2:0] is always ”0”.
2 :0		**Reserved (0)**
		GER00: Base Address of Font Buffer Register (Enhanced Font Expansion)
31:28		**Reserved (0)**
27:3	RW	**Base address of font buffer bit [27:3]** The address should be always 8-bytes aligned. Therefore, bit [2:0] is always ”0”.
2 :0		**Reserved (0)**

<table>
<tr><th>Offset: 04h</th><th></th><th>GER04: Row Pitch of Source Buffer Register — Init = X</th></tr>
<tr><td>Bit</td><td>R/W</td><td>Description</td></tr>
<tr><td>31:29</td><td></td><td>Reserved (0)</td></tr>
<tr><td>28:16</td><td>RW</td><td>Row pitch of source buffer bit[12:0]
Row pitch of source buffer is equal to the width of source bitmap multiplied by bytes per pixel.

The range of row pitch of source buffer has to meet the following limitations (number of bytes):
<table>
<tr><th>MODE</th><th>Value</th></tr>
<tr><td>256 color</td><td>0000h∼07FFh</td></tr>
<tr><td>High color</td><td>0000h∼0FFFh</td></tr>
<tr><td>True color</td><td>0000h∼1FFFh</td></tr>
</table></td></tr>
<tr><td>15:0</td><td></td><td>Reserved (0)</td></tr>
<tr><td></td><td></td><td>GER04: Row Pitch of Font Buffer Register (Enhanced Font Expansion)</td></tr>
<tr><td>31:29</td><td></td><td>Reserved (0)</td></tr>
<tr><td>28:16</td><td>RW</td><td>Row pitch of font buffer bit[12:0]
GER04[28:16] = (GER18[26:16] + 7) >> 3

0 < GER04[28:16] * GER18[10:0] <= fffh</td></tr>
<tr><td>15:0</td><td></td><td>Reserved (0)</td></tr>
</table>

Offset: 08h		GER08: Base Address of Destination Buffer Register — Init = X
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:3	RW	**Base address of destination buffer (bit [27:3])** The address should be always 8-bytes aligned. Therefore, bit [2:0] is always ”0”.
2 :0		**Reserved (0)**

Offset: 0Ch		GER0C: Row Pitch and Height of Destination Buffer Register Init = X
Bit	**R/W**	**Description**
31:29		**Reserved (0)**
28:19	RW	**Row pitch of destination buffer bit [12:3]** Row pitch of destination buffer is equal to the width of destination buffer multiplied by bytes per pixel. The range of row pitch of destination buffer has to meet the following limitations (number of bytes): MODE: Value 256 color: 0000h~07F8h High color: 0000h~0FF8h True color: 0000h~1FF8h
18:11		**Reserved (0)**
10:0	RW	**Height of destination buffer bit [10:0]** Height of destination buffer has to be in the range of 0000h~07FFh.

Offset: 10h		GER10: Coordinate of Destination Bitmap Register Init = X
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:16	RW	**X coordinate of top-left corner of destination bitmap bit [11:0]** The data format of this register is S11.0
15:12		**Reserved (0)**
11:0	RW	**Y coordinate of top-left corner of destination bitmap bit [11:0]** The data format of this register is S11.0
		GER10: Coordinate of Start Point of Line Drawing Register (Line)
31:28		**Reserved (0)**
27:16	RW	**X coordinate of start point of line drawing bit [11:0]** The data format of this register is S11.0
15:12		**Reserved (0)**
11:0	RW	**Y coordinate of start point of line drawing bit [11:0]** The data format of this register is S11.0

Offset: 14h		GER14: Coordinate of Source Bitmap Register Init = X
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:16	RW	**X coordinate of top-left corner of source bitmap bit [11:0]** The data format of this register is S11.0 When GER3C[2:0] = 2 or 3, 0 MUST be the value.
15:12		**Reserved (0)**
11:0	RW	**Y coordinate of top-left corner of source bitmap bit [11:0]** The data format of this register is S11.0 When GER3C[2:0] = 2 or 3, 0 MUST be the value.
		GER14: Major and Error Term of Line Drawing Register (Line)
31:25		**Reserved (0)**

to next page

from previous page

24	RW	**Major axis selection** 0: Select Y-Axis as the major axis for line drawing 1: Select X-Axis as the major axis for line drawing
23:22		**Reserved (0)**
21:0	RW	**Error term of line drawing algorithm bit [21:0]** This register defines the Error Term of a line drawing algorithm.

Offset: 18h	**GER18: Drawing Width and Drawing Height Register**	**Init = X**
Bit	**R/W**	**Description**
31:27		**Reserved (0)**
26:16	RW	**Width of destination bitmap bit [10:0]** Width of destination bitmap should be in the range below. MODE: Value 256 color: 1~2040 High color: 1~2044 True color: 1~2046
15:11		**Reserved (0)**
10:0	RW	**Height of destination bitmap bit [10:0]**
	GER18: Width of Major Axis of Line Drawing Register (Line)	
31:27		**Reserved (0)**
26:16	RW	**Width of major axis of line drawing bit [10:0]**
15:0		**Reserved (0)**

Offset: 1Ch	**GER1C: Foreground Color of Pattern Register**	**Init = X**
Bit	**R/W**	**Description**
31:0	RW	**Foreground color of pattern bit[31:0]**

Offset: 20h	**GER20: Background Color of Pattern Register**	**Init = X**
Bit	**R/W**	**Description**
31:0	RW	**Background color of pattern bit [31:0]**

Offset: 24h	**GER24: Foreground Color of Source Register**	**Init = X**
Bit	**R/W**	**Description**
31:0	RW	**Foreground color of source bit [31:0]**
	GER24: K1 Term of Line Drawing Register (Line)	
31:22		**Reserved (0)**
21:0	RW	**K1 term of line drawing algorithm bit [21:0]**

Offset: 28h	**GER28: Background Color of Source Register**	**Init = X**
Bit	**R/W**	**Description**
31:0	RW	**Background color of source bit [31:0]**
	GER28: K2 Term of Line Drawing Register (Line)	
31:22		**Reserved (0)**

to next page

from previous page

21:0	RW	**K2 term of line drawing algorithm bit [21:0]**

Offset: 2Ch		**GER2C: Monochrome Mask of Pattern Register # 0** **Init = X**
Bit	**R/W**	**Description**
31:0	RW	**Monochrome mask of pattern bit [31:0]**
		GER2C: Pattern of Line Style Register # 0 (Line)
31:0	RW	**Pattern of line style bit [31:0]**

Offset: 30h		**GER30: Monochrome Mask of Pattern Register # 1** **Init = X**
Bit	**R/W**	**Description**
31:0	RW	**Monochrome mask of pattern bit [63:32]**
		GER30: Pattern of Line Style Register # 1 (Line)
31:0	RW	**Pattern of line style bit [63:32]**

Offset: 34h		**GER34: Top-Left Clipping Corner of Rectangular Register** **Init = X**
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:16	RW	**X coordinate of top-left corner of clipping rectangular bit [11:0]**
15:12		**Reserved (0)**
11:0	RW	**Y coordinate of top-left corner of clipping rectangular bit [11:0]**

Offset: 38h		**GER38: Bottom-Right Corner of Clipping Rectangular Register** **Init = X**
Bit	**R/W**	**Description**
31:28		**Reserved (0)**
27:16	RW	**X coordinate of bottom-right corner of clipping rectangular bit [11:0]**
15:12		**Reserved (0)**
11:0	RW	**Y coordinate of bottom-right corner of clipping rectangular bit [11:0]**

Offset: 3Ch		**GER3C: 2D Engine Command Register** **Init = 0**
Bit	**R/W**	**Description**
31	RW	**Reset line style counter** 0: No operation 1: Reset line style counter This register will determine the line style counter to be reset or not when executing a new line drawing command.
30	RW	**Enable line drawing command with style pattern** 0: Disable (line drawing command without a style pattern) 1: Enable (line drawing command with a style pattern)
29:24	RW	**Line style period[5:0]** Line style period can be up to 64 points at most.
23	RW	**End-point rendering control for line drawing commands** 0: Disable end-point rendering 1: Enable end-point rendering

to next page

from previous page

22		**Reserved (0)**
21	RW	**X-axis rendering direction control** 0: Rendering in positive-X direction 1: Rendering in negative-X direction
20	RW	**Y-axis rendering direction control** 0: Rendering in positive-Y direction 1: Rendering in negative-Y direction
19		**Reserved (0)**
18	RW	**Enable transparent font expansion** 0: Enable opaque font expansion 1: Enable transparent font expansion
17:16	RW	**Pattern selection** 00: Pattern is from foreground color of pattern register 01: Pattern is from monochrome mask register 10: Pattern is from pattern register 11: **Invalid**
15:8	RW	**Command code of 256 raster operations bit[7:0]**
7	RW	**Enable transparent of monochrome mask** 0: Enable opaque mode 1: Enable transparent mode
6	RW	**Source bitmap selection** 0: Source bitmap is from video frame buffer 1: Source bitmap is from command queue (Line drawing command does NOT support)
5 :4	RW	**Color mode selection** 00: 256 color mode (8-bpp) 01: High color mode (16-bpp) 10: True color mode (24-bpp) 11: **Invalid**
3	RW	**Enable rectangular clipping** 0: Disable rectangular clipping 1: Enable rectangular clipping
2 :0	RW	**Command type selection** 000: BitBLT command 001: Line drawing command 010: Font expansion command (patterns are from registers) 011: Enhanced font expansion command (patterns are from frame buffer) 1xx: **Invalid**

Offset: 44h		GER44: Command Queue Setting Register Init = 0
Bit	**R/W**	**Description**
31:28	RW	**Available size of hardware command queue bit[3:0]** 0000: 0 Bytes 0001: 8 Bytes 0010: 16 Bytes 0011: 24 Bytes ... 1111: 120 Bytes

to next page

from previous page

27:26	RW	**Command queue buffer size** 00: 256KB 01: 512KB 10: 1MB 11: 2MB
25	RW	**Mode of command queue operation** 0: Command data is from video frame buffer 1: Command data is from memory-mapped I/O command
24:0	RW	**Base address of command queue buffer bit[27:3]**

Offset: 48h		GER48: Write-Pointer of Command Queue Register Init = 0
Bit	**R/W**	**Description**
31:18		**Reserved (0)**
17:0	RW	**Write-pointer of command queue bit [20:3]**

Offset: 4Ch		GER4C: 2D Engine Status Register Init = 0
Bit	**R/W**	**Description**
31	R	**Status of 2D Graphic Engine** 0: Engine is idle 1: Engine is busy
30:18	R	**Debug Port** **(for debugging purpose only)**
17:0	R	**Read-pointer of command queue bit [20:3]**

Offset: 100~ 1FCh		PTR00 ~ PTRFC: Pattern Register # 1 ~ #64 Init = X
Bit	**R/W**	**Description**
31:0	RW	**Pattern register for ROP bit [31:0]**
		PTR00 ~ PTRFC: Monochrome Bitmap Register # 1 ~ #64 (Font Expansion)
31:0	RW	**Monochrome bitmap register bit [31:0]**

36 P-Bus to AHB Bridge

36.1 Overview

P-to-AHB Bus Bridge (P2A) is an interface controller bridging two internal buses:

P-Bus: The internal expansion bus supporting bus commands from PCI slave controller
AHB : The internal system bus supporting ARM SOC subsystem

P2A is a one-way bus bridge providing a back door for host CPU to access all the internal IP modules in ARM SOC sub-system. Since P2A is a one-way bridge, ARM CPU cannot issue any PCI bus commands through the help of this bridge. In a normal condition, this back door should be well locked. The two potential usages of this bus bridge are:

1. Updating flash memory through host CPU
2. H/W or S/W debugging through host CPU

P2A only implements two sets of 32-bit registers to provide a protection mechanism and specify the base address of the 64KB address re-mapping window.

36.2 Registers : Base Address = MMIOBASE

Offset: F000h		P2A00: Protection Key Register Init = 0
Bit	**R/W**	**Description**
31:1		**Reserved (0)**
0	RW	**Protection key** 0: Disable P2A bridge 1: Enable P2A bridge When P2A is disabled, it will ignore all the P-Bus commands. Therefore, there will be no command conversion from P-Bus to AHB. Always keep this protect key in disabled state when there is no need.

Offset: F004h		P2A04: Re-mapping Base Address Register Init = X
Bit	**R/W**	**Description**
31:16	R	**Re-mapping base address** This register defines the address re-mapping scheme from P-Bus to AHB. Bit [31:16] of AHB address is from the Bit [31:16] of this register, Bit [15:0] is directly from P-Bus command address. AHB Address = (Re-mapping base address[31:16]) + (P-bus address[15:0]) P2A will convert all the commands from P-bus with 64KB address range from **(MMIOBASE + 0x10000)** to **(MMIOBASE + 0x1FFFF)**. Where MMIOBASE is the re-locatable memory-mapped I/O base address defined in PCI configuration space. P2A supports byte, word or double word type of access commands.
15:0		**Reserved (0)**

37 Graphics Hardware Cursor

37.1 Features

- Supports 64x64 monochrome cursor with AND-XOR-RGB444 pixel format
- Supports 64x64 color cursor with ARGB4444 pixel format
- Supports X-Offset & Y-Offset options
- Cursor information can be read from VGA Scratch Registers
- Cursor bit-map can be read from the designated area within VGA frame buffer
- Automatically generates Cursor Interrupt when cursor information or cursor bit-map address is changed

37.2 Register Definition

Offset: 1E70:0008h — VR008: Video Engine Control Register — Init = 0

Bit	R/W	Description
8	RW	**Disable hardware cursor overlay for internal VGA** 0: With VGA hardware cursor overlay image 1: Without VGA hardware cursor overlay image This register can be set by ARM CPU to inform internal VGA controller to generate video data without hardware cursor overlay image. When this register is enabled, the hardware cursor overlay has to be done in clients by Quick Cursor algorithm. The DAC output of internal VGA controller is, if necessary, with hardware cursor overlay image even this register is set to 1.

Offset: 1E6E:2018h — SCU18: Interrupt Control and Status Register — Init = 0

Bit	R/W	Description
31:18		**Reserved (0)**
17	RW	**VGA scratch register change Interrupt and status** 0 : No interrupt occurs 1 : Interrupt occurs The status flag can be cleared by writing '1' to this bit.
16	RW	**VGA cursor change interrupt and status** 0 : No interrupt occurs 1 : Interrupt occurs The status flag can be cleared by writing '1' to this bit.
15:2		**Reserved (0)**
1	RW	**Enable VGA scratch register change interrupt** 0 : Disable Interrupt 1 : Enable Interrupt generation
0	RW	**Enable VGA cursor change interrupt** 0 : Disable Interrupt 1 : Enable Interrupt generation

Offset: 1E6E:2050 — VGA Scratch Register #1 — Init = 0

Bit	R/W	Description
31:30	R	**Reserved**

to next page

from previous page

29:24	R	**Hardware cursor X position offset bit [5:0]**
23:22	R	**Reserved**
21:16	R	**Hardware cursor Y position offset bit[5:0]**
15:10	R	**Reserved**
9	R	**Hardware cursor type selection** 0 : Select monochrome cursor type 1 : Select color cursor type
8	R	**Hardware cursor is enabled** 0 : Disabled hardware cursor 1 : Enable hardware cursor
7:0	R	**Reserved**

Offset: 1E6E:2054h — VGA Scratch Register #2 — Init = 0

Bit	R/W	Description
31:27	R	**Reserved**
26:16	R	**Hardware cursor Y position bit[10:0]**
15:12	R	**Reserved**
11:0	R	**Hardware cursor X position offset bit[11:0]**

Offset: 1E6E:2058h — VGA Scratch Register #3 — Init = 0

Bit	R/W	Description
31:28	R	**Reserved**
27:0	R	**Hardware cursor pattern memory address bit [27:0]**

37.3 Cursor Shape Structure Definition

37.3.1 Monochrome Cursor Format (AND-XOR-RGB444 pixel format)

Bit[15] : AND Mask bit
Bit[14] : XOR Mask bit
Bit[13:12] : Reserved
Bit[11:8] : Cursor R bit[3:0]
Bit[7:4] : Cursor G bit[3:0]
Bit[3:0] : Cursor B bit[3:0]

Description	AND Mask bit	XOR Mask bit	Output Color
Background Color	0	0	Cursor R/G/B
Foreground Color	0	1	Cursor R/G/B
Transparent	1	0	Graphics R/G/B
Inversed	1	1	NOT Graphics R/G/B

37.3.2 Color Cursor Format (ARGB4444 pixel format)

Bit[15:12] : Alpha bit[3:0]
Bit[11:8] : Cursor R bit[3:0]
Bit[7:4] : Cursor G bit[3:0]
Bit[3:0] : Cursor B bit[3:0]

- Output Color = Alpha x Graphics R/G/B + (1-Alpha) * Cursor R/G/B
- Note:
  1. Graphcis R/G/B is the color bit-map decompressed from video stream
  2. The output color should be normalized to the target display format
  3. When X-Offset or Y-Offset is enabled, only a partial bit-map is displayed