VT6509

9-PORT 10/100BASE-T/TX ETHERNET SWITCH CONTROLLER

REVISION 'F' DATASHEET
(Preliminary)

ISSUE 2: Aug 28, 2000

VIA Technologies, Inc.

PRELIMINARY RELEASE

Please contact VIA Technologies for the latest documentation.

Copyright Notice:

Disclaimer Notice:

No license is granted, implied or otherwise, under any patent or patent rights of VIA Technologies. VIA Technologies makes no warranties, implied or otherwise, in regard to this document and to the products described in this document. The information provided by this document is believed to be accurate and reliable to the publication date of this document. However, VIA Technologies assumes no responsibility for any errors in this document. Furthermore, VIA Technologies assumes no responsibility for the use or misuse of the information in this document and for any patent infringements that may arise from the use of this document. The information and product specifications within this document are subject to change at any time, without notice and without obligation to notify any person of such change.

Offices:

1045 Mission Court
Fremont, CA 94539
USA

Tel: (510) 683-3300
Fax: (510) 683-3301

8th Floor, No. 533
Chung-Cheng Rd., Hsin-Tien
Taipei, Taiwan ROC

Tel: (886-2) 2218-5452
Fax: (886-2) 2218-5453

Online Services:

BBS : 886-2-2186408
FTP : FTP.VIA.COM.TW
HTTP:**WWW.VIA.COM.TW** –or- WWW.VIATECH.COM

TABLE OF CONTENTS

TABLE OF CONTENTS 3
FIGURES AND TABLES 4
REVERSION HISTORY 5
FEATURES 6
BLOCK DIAGRAM 7
PINOUT DIAGRAM 8
PIN DESCRIPTIONS 9
DEFINITION OF VT6509E STRAPPING PINS 12

SECTION I FUNCTIONAL DESCRIPTIONS 17

1 GENERAL DESCRIPTION 17
2 THE VIA ETHER SWITCH ARCHITECTURE 17
2.1 Switch initialization procedures 17
2.2 Packet Switching Flow 17
2.3 Packet Buffers and Forwarding Table 18
2.4 RMII Interface 21
2.5 MII or Reverse MII Interface 21
2.6 Management Interface and Auto Negotiation 21
2.7 Flow Control 23
2.8 Broadcast Storm Filtering 26
2.9 Serial EEPROM interface and Configuration commands 27
2.10 Trunking 28
3 THE VT6509 SRAM ADDRESS MAPPING TABLE 28

SECTION II REGISTER MAP 31

1. REGISTERS TABLE 31

SECTION III ELECTRICAL SPECIFICATIONS 43

ABSOLUTE MAXIMUM RATINGS 43
DC CHARACTERISTICS 43
AC CHARACTERISTICS 44
PACKAGE MECHANICAL SPECIFICATIONS 46

FIGURES AND TABLES

Figure 1: Function Block Diagram of VT6509. 7
Figure 2: Pinout Diagram of VT6509..... 8
Figure 3. SRAM memory layout. 19
Figure 4. Data structure of forwarding table slot..... 19
Figure 5. Data structure of embedded link node..... 19
Figure 6. XON/XOFF Window Concept..... 25

REVERSION HISTORY

Reversion	Date	Reason for change	By
V0.01	2/17/2000	First release for version E	Murphy Chen
V0.02	8/28/2000	First release for version F	Murphy Chen

FEATURES

- Single chip 9 ports 10/100Mbps Ethernet switch controller
  - Highly integrated single chip shared memory switch engine
  - Supports one MII (Media Independent Interface) port and 8 RMII (Reduced MII) ports
  - Non-blocking layer 2 switch, 148,810 packets/sec on each 100Mbps Ethernet port
- Media Access Control (MAC)
  - Dual 64-byte FIFO's per port for receiving and transmitting
- Auto-sensing 10/100Mbps media speed, full/half -duplex mode, and flow-control capability
- Two switching mechanisms
  - Supports 'store and forward' switching with filtering CRC-bad packets
  - Supports 'cut through' switching subject to long packets of length over 64 bytes
- Packet buffering
  - Glueless 32 or 64-bit interface to SSRAM as a packet buffer pool
  - 64 packet buffers for 32Kx32 SSRAM, 149 packet buffers for 64Kx32 SSRAM
  - 1536 bytes for each packet buffer
- Control data within external SSRAM
  - Packet link entry is stored at the packet buffer's tail word, buffer status is stored in the internal free-buffer bit map registers
  - Shared forwarding table of 2K entries (or 4K slots) to support multiple (up to 4K) Mac addresses per port
- Efficient address recognition, self learning, and auto-aging mechanism
  - Two-slot hashing algorithm to prevent hash collision
  - Two optional hashing algorithms, CRC-map and direct-map
- Advanced congestion control mechanism
  - IEEE 802.3X compliant flow control for full duplex ports
  - Backpressure for half duplex ports
  - Drop control for full duplex ports without flow control capability
  - Incorporating with the output private buffering scheme to prevent HOL (head of line) blocking
  - Broadcast storm filtering for Ethernet ports and Early Broadcast storm filtering for CPU port
- Supports Port mirroring (Sniffer feature)
- By-pass VLAN packets
- Supports Port-based trunking
  - Support two individual trunk groups, each of 2 member ports
  - Load balance according to DMAC address and source port number
- Supports 8-bit IDE bus for network management
- Reverse MII interface for SOHO router application with an embedded processor
- Supports STP (Spanning Tree Protocols)
- Supports 8-bit IDE bus for network management by an embedded CPU
- Chip initialization through CPU, EEPROM, or strapping only
  - Supports I^2C EEPROM interface for customized configuration

- Supports LED serial-out in the strapping-only initialization mode
- Supports LED serial-out to display port status
- 50MHz internal reference clock rate
- 50~100MHz SSRAM clock rate, typically 50MHz enough for wire-speed throughput
- Single +3.3V supply, 0.3µm TSMC CMOS technology
- 128-pin PQFP package

BLOCK DIAGRAM

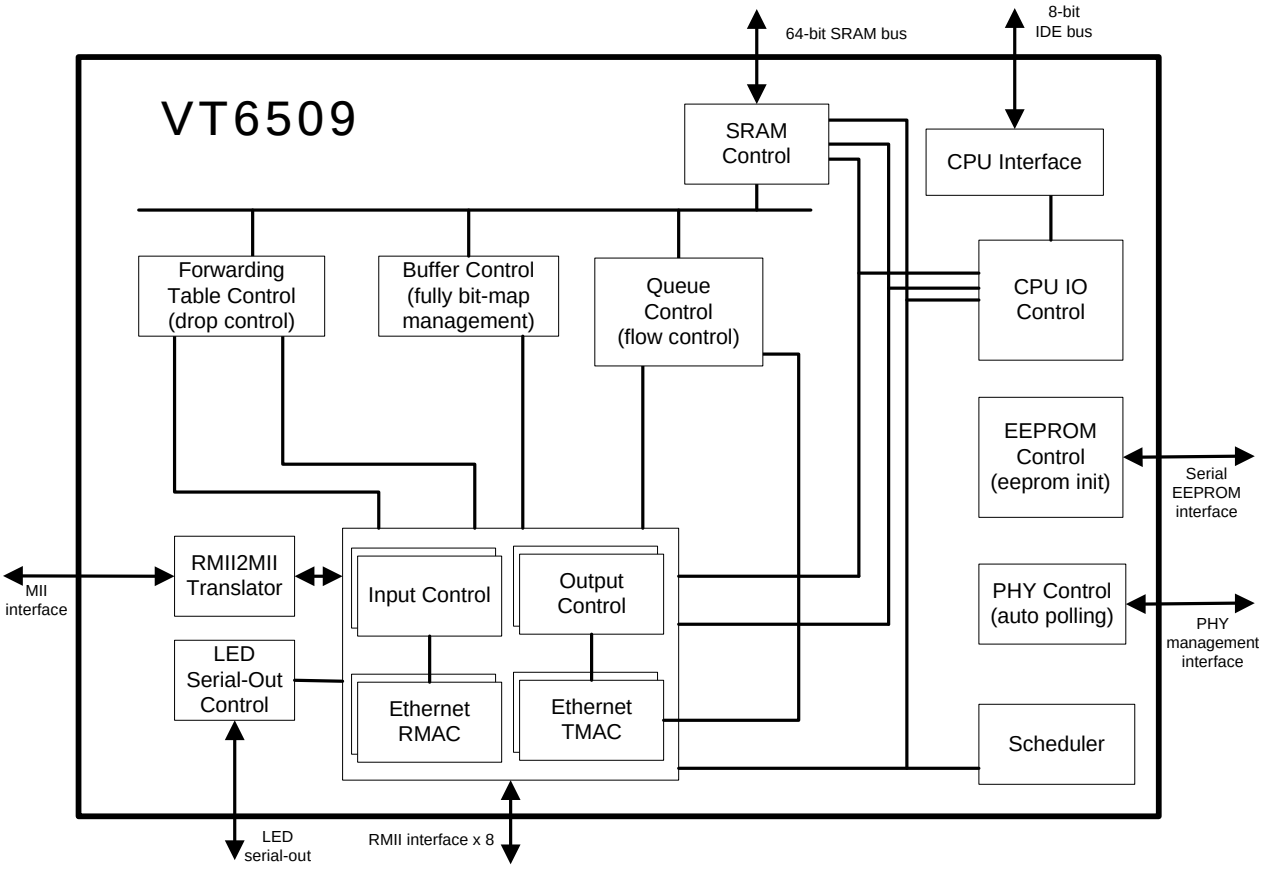

Figure 1: Function Block Diagram of VT6509.

PINOUT DIAGRAM

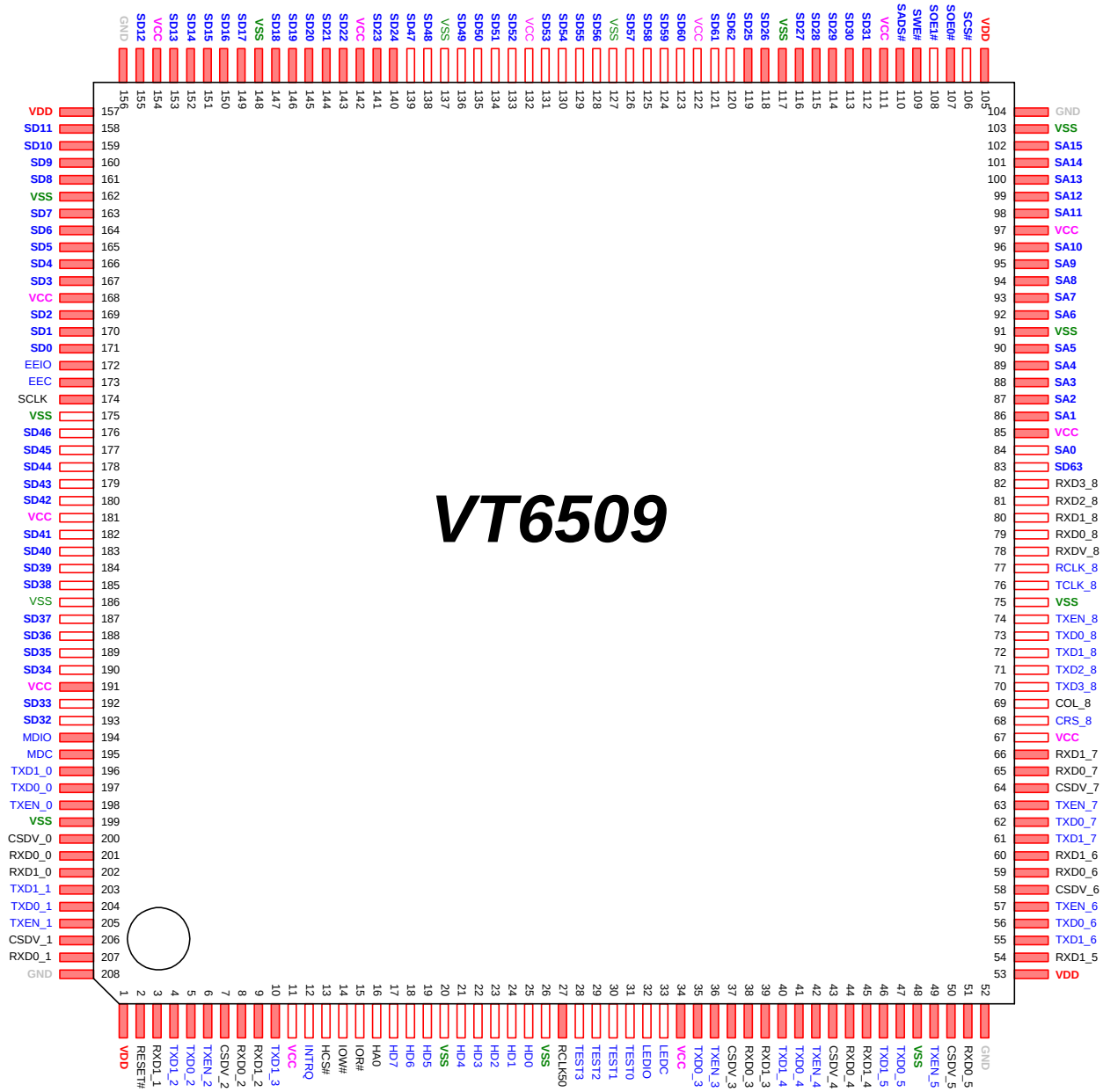

Figure 2: Pinout Diagram of VT6509.

PIN DESCRIPTIONS

No.	Name	Type	Description
			RMII interface
See Fig. 2	CSDV[7:0]	I	Carries sense and data valid from port 7 to port 0 :
See Fig. 2	RXD0[7:0]	I	Receive data zero from port 7 to port 0 :
See Fig. 2	RXD1[7:0]	I	Receive data one from port 7 to port 0 :
See Fig. 2	TXEN[7:0]	O	Transmit enable from port 7 to port 0 :
See Fig. 2	TXD0[7:0]	O	Transmit data zero from port 7 to port 0 :
See Fig. 2	TXD1[7:0]	O	Transmit data one from port 7 to port 0 :
			MII interface
See Fig. 2	TCLK_8	I O	**Transmit Clock for Port 8:** TCLK is sourced from the PHY. TCLK is a continuous clock that provides the timing reference for the transfer of the TXEN and TXD signals to the PHY. A PHY operating at 100Mbps must provide a TCLK frequency of 25MHz and a PHY operating at 10Mbps must provide a TCLK frequency of 2.5MHz. It becomes an output pin when reverse MII is enabled.
See Fig. 2	TXD<3:0>_8	O	**Transmit Data for Port 8:** TXD is a bundle of 4 data signals (TXD<3:0>) that shall transition to the TCLK. For each TCLK period in which TXEN is asserted, TXD<3:0> are accepted for transmission by the PHY. TXD<0> is the least significant bit. While TXEN is de-asserted, TXD<3:0> shall have no effect upon the PHY, and the value of TXD<3:0> is unspecified.
See Fig. 2	TXEN_8	O	**Transmit Enable for Port 8:** TXEN shall transition synchronous to the TCLK. TXEN indicates the nibbles presenting on the MII for transmission. It shall be asserted synchronously with the first nibble of the preamble and shall remain asserted while all nibbles to be transmitted are presented to the MII.
See Fig. 2	COL_8	I	**Collision Detected for Port 8:** COL shall be asserted by the PHY a synchronously upon detection of a collision on the medium, and shall remain asserted while the collision condition persists. It does not drive when reverse MII is enabled because it must be full duplex.
See Fig. 2	CRS_8	I O	**Carrier Sense for Port 8:** CRS shall be asserted by the PHY asynchronously upon detection of a non-idle medium or while TX_EN is asserted. CRS shall be de-asserted by the PHY asynchronously upon detection of idle conditions on both transmit and receive media. The PHY shall ensure that CRS remains asserted throughout the duration of a collision condition. It becomes an output pin when reverse MII is enabled.
See Fig. 2	RXD<3:0>_8	I	**Receive Data for Port 8:** RXD is a bundle of 4 data signals (RXD<3:0>) that shall transition to the RCLK. For each RCLK period in which RXDV is asserted, RXD<3:0> from the PHY are accepted by the switch's MAC. RXD<0> is the least significant bit. While RXDV is de-asserted, RXD<3:0> shall have no effect upon the switch's MAC, and the value of RXD<3:0> is unspecified.

See Fig. 2	RCLK_8	I O	**Receive Clock for Port 8:** RCLK is sourced from the PHY. RCLK is a continuous clock that provides the timing reference for the transfer of the RXDV and RXD signals from the PHY. A PHY operating at 100Mbps must provide a RCLK frequency of 25MHz and a PHY operating at 10Mbps must provide a RCLK frequency of 2.5MHz. It becomes an output pin when reverse MII is enabled.
See Fig. 2	RXDV_8	I O	**Receive Data Valid for Port 8:** RXDV is driven by the PHY to indicate the nibbles presenting on the MII for receiving. RXDV shall transition synchronous to the RCLK. It shall be asserted synchronously with the first nibble of the preamble and shall remain asserted while all nibbles to be received are presented to the MII. It becomes an output pin when reverse MII is enabled.
SRAM Interface			
See Fig. 2	SA[15:0]	O	**SRAM Address Bus:** 15-bit SRAM address bus. These signals connect directly to the address input of the SRAM devices.
See Fig. 2	$\overline{SCS}$	O	**Chip Select**
See Fig. 2	$\overline{SOE}$[1:0]	O	**Output Enable**
See Fig. 2	$\overline{SWE}$	O	**SRAM Write Enable**
See Fig. 2	$\overline{SADS}$	O	**Synchronous Processor Address Status**
See Fig. 2	SD[63:0]	I/O	**SRAM Data:** 64-bit SRAM data bus. These signals connect directly to the data input/output pins of the SRAM devices.
HOST Interface			
See Fig. 2	HD[7:0]	I/O	**HOST IDE-Interface Data Bus:** The 8-bit data bus is used for internal registers read/write.
See Fig. 2	HA	I	**HOST IDE-Interface Address Bus:** 1'b0: reference to 16-bit address register For the first write of HA=0, the low byte of the 16-bit address will be updated. For the subsequent write of HA=0, the high byte of the 16-bit address will be updated. The read/write to any internal registers should follows the steps: (1) write low-byte address (HA=0) (2) write high-byte address (HA=0) (3) read/write 8-bit data (HA=1) or (1) write low-byte address (HA=0) (2) read/write 8-bit data (HA=1) 1'b1: reference to 8-bit data register

See Fig. 2	$\overline{HCS}$	I	**HOST Chip Select:** Active LOW. $\overline{HCS}$ must be asserted during the access of HOST IDE interface.
See Fig. 2	$\overline{IOR}$	I	**IO READ:** High-to-Low Edge Trigger. $\overline{IOR}$ must be asserted from high to low to begin the read cycle of HOST IDE interface.
See Fig. 2	$\overline{IOW}$	I	**IO READ:** High-to-Low Edge Trigger. $\overline{IOW}$ must be asserted from high to low to begin the write cycle of HOST IDE interface.
Miscellaneous Interface			
See Fig. 2	EEC	O	**Serial EEPROM Interface Clock** The on-board EEPROM device address must be 1010 001 XXXXXXXX.
See Fig. 2	EEIO	I/O	**Serial EEPROM Interface Data**
See Fig. 2	LEDC	O	**Serial LED Serial-Out Clock** The LEDC is 1MHz and has a burst per 50ms.
See Fig. 2	LEDIO	I/O	**Serial LED Serial-Out Data Output**
See Fig. 2	MDIO	I/O	**Management Interface (MI) Data I/O**
See Fig. 2	RCLK50	I	**50MHz Main Reference Clock**
See Fig. 2	SCLK	I	**SRAM Reference Clock** **The suggested clock rate is 83MHz or more high for non-blocking requirement.**
See Fig. 2	$\overline{RESET}$	I	**SYSTEM RESET**

Power Supply & Ground			
See Fig. 2	VDD	P	Positive 3.3V supply to the core digital logic.
See Fig. 2	VCC	P	Positive 3.3V supply to all I/O pads.
See Fig. 2	GND	G	Ground supply to the core digital logic.
See Fig. 2	VSS	G	Ground supply to all I/O pads.

DEFINITION OF VT6509E STRAPPING PINS

SD[63:0] and TEST0, TEST1: **Note that the default strapping bit value is "1".*

Pin	Description
SD[0]	**Enable BPDU packets broadcast as CPU_FWD_CFG[1] == 0:** BCAST_BPDU == 1' b1 => broadcast BPDU packet if CPU_FWD_CFG[1] == 0 (default) BCAST_BPDU == 1' b0 => drop BPDU packet if CPU_FWD_CFG[1] == 0 Note that BPDU packets will be forwarded to CPU port if CPU_FWD_CFG[1] == 1 without regard to the value of BCAST_BPDU.
SD[1]	**LED Display Combination of Link Activity Status with RX/TX Event:** LED_COMB [0] == 1' b1 => combined (default) LED_COMB [0] == 1' b0 => not combined Note that the LED bit sequence [10:1] is used originally to display link activity status. By setting combination with RX/TX event, it will indicate both the link activity status and RX/TX events so that required LED number can be reduced.
SD[2]	**Enable Drop Control for Private Buffer Reservation:** DROP_CONTROL_EN [0] == 1' b1 => enabled (default) DROP_CONTROL_EN [1] == 1' b0 => disabled Note that drop control has a lower priority than flow control and backpressure. If it is disabled, all TMACs will not make the drop window signals to Forwarding Control. If it is enabled and the flow control/backpressure mechanism is not enabled, all TMACs will make the drop window signals to Forwarding Control. If it is enabled and the flow control mechanism is also enabled, but the full-duplex party has no flow control capability, TMAC will make the drop window signal to Forwarding Control.
SD[3,47]	**SRAM TYPE [1:0]:** SRAM_CONFIG[1:0] == 2' b11 => 64Kx32 (default) SRAM_CONFIG[1:0] == 2' b10 => 64Kx64 SRAM_CONFIG[1:0] == 2' b01 => 32Kx32 SRAM_CONFIG[1:0] == 2' b00 => 128Kx32 Note that SRAM_CONFIG[1] is at SD[3] and SRAM_CONFIG[0] is at SD[47]. size=128KB: FREEMCNT=64, XON_THRED=43, PRIVATE_BUF_SIZE=1 V_FREE=10, MIN_XOFF_CONST_THRED=**15** size=256KB: FREEMCNT=149, XON_THRED=98, PRIVATE_BUF_SIZE=4 V_FREE=10, MIN_XOFF_CONST_THRED=28 size=512KB: FREEMCNT=320, XON_THRED=198, PRIVATE_BUF_SIZE=6 V_FREE=20, MIN_XOFF_CONST_THRED=37 size=1MB : FREEMCNT=661, XON_THRED=381, PRIVATE_BUF_SIZE=8 V_FREE=30, MIN_XOFF_CONST_THRED=37
SD[5:4]	**CHIP INIT MODE [1:0]:** CHIP_INIT[1:0] == 2' b11 => Chip Initialization via strapping only (default) CHIP_INIT[1:0] == 2' b10 => Chip Initialization via EEPROM CHIP_INIT[1:0] == 2' b01 => Chip Initialization via CPU CHIP_INIT[1:0] == 2' b00 => Chip Initialization via EEPROM in speedup mode In none speedup mode, EEC = 78.125K Hz, LED period = 50ms. In speedup mode for testing, EEC = 2.778MHz, LED period = 1ms without one-second flash.
SD[6]	**Enable Jabber Control:** EN_JABBER_CTL == 1' b1 => enable jabber control (default)

	EN_JABBER_CTL == 1' b0 => disable jabber control By default, when jabber (defined as packet length > 6192 bytes) is detected in some port, the following operations are performed: (1) issue a RE-AUTO-NEGO command to the corresponding external PHY device, (2) disable that port' s I/O Control, (3) mask carrier sense signal RX_CRS to let TMAC unblocked. If EN_JABBER_CTL == 1' b0, the above operations will not executed even when jabber is detected.
SD[7]	**Number of Tries of Excessive Collisions Before Dropping:** RETRY_EXCE_COLL == 1' b1 => 3 tries (default) RETRY_EXCE_COLL == 1' b0 => retry forever By default, after 3 tries (i.e. 3 turns of (excessive) 16 collisions), the outgoing packet will be dropped.
SD[44,42,15:8]	**RMII SPEED REG CFG [9:0]:** SPEED_REG_CFG [4:0] : accessed PHY_REG number SPEED_REG_CFG [8:5] : accessed PHY_REG bit number bit[9]==1 => when the accessed bit equal "1" means Speed = 10Mbps when the accessed bit equal "0" r bit[9]==0 => when the accessed bit equal "1" means Speed = 100Mbps when the accessed bit equal "0" r Note: this register is valid only when RMII_SPEED_ARB == 0. Note: In test mode, this 10-bit strapping register is used as follows: * RMII SPEED REG CFG [0] / TEST MODE SELECT [0] * RMII SPEED REG CFG [1] / TEST MODE SELECT [1] * RMII SPEED REG CFG [2] / TEST MODE SELECT [2] * RMII SPEED REG CFG [3] / TEST MODE SELECT [3] * RMII SPEED REG CFG [4] / TEST MODE SELECT [4] * RMII SPEED REG CFG [5] / DEFAULT SPEED * RMII SPEED REG CFG [6] / DEFAULT DUPLEX MODE * RMII SPEED REG CFG [7] / DEFAULT FLOW CONTROL
SD[45,17, 43,16]	**Trunking Mode:** TRUNK_MODE[0] == 1, port (0,1) no trunking (default) 0, port (0,1) in trunking mode, TRUNK_MODE[1] == 1, port (0,1) off-board trunking (default) 0, port (0,1) on-board trunking TRUNK_MODE[2] == 1, port (6,7) no trunking (default) 0, port (6,7) in trunking mode. TRUNK_MODE[3] == 1, port (6,7) off-board trunking (default) 0, port (6,7) on-board trunking To simplify layout, trunk group (6,7) can connect with neighboring (right-side) VT6509' s trunk group (0,1) in on-board manner. Note that on-board trunk ports do not need auto-polling and they must be full-duplex & 100Mbps ports & forced flow control enable mode.
SD[18]	**Enable Backpressure:** BACKPRESSURE_EN [0] == 1' b1 => enabled (default) BACKPRESSURE_EN [1] == 1' b0 => disabled Note that if it is disabled, all TMAC of half-duplex ports will ignore the flow control XON/XOFF signals from Queue Control.
SD[20:19]	**Enable RMII Port Flow Control:** RMII_PORT_FC_EN[1:0] == 2' b11 => for RMII ports, set self flow control ability register bit and detect party' s flow control ability (default) RMII_PORT_FC_EN[1:0] == 2' b10 => for RMII ports, disable self flow control ability register bit and assume (ignore auto-polling) party has no flow control capability

	RMII_PORT_FC_EN[1:0] == 2' b01 => for RMII ports, enable self flow control ability register bit and assume (ignore auto-polling) party has flow control capability RMII_PORT_FC_EN[1:0] == 2' b00 => same as 2' b11. While enabled (RMII_PORT_FC_EN[1:0] == 2' b11 or 2' b00), set self each PHY device' s flow control ability register bit PHY_ANAR_4.10 as 1, and auto polling PHY_ANLPAR_5.10 to check whether the party has the flow control capability. While forced disabling (RMII_PORT_FC_EN[1:0] == 2' b10), set self PHY devices' flow control ability register bit PHY_ANAR_4.10 as 0 (no such capability), and does not need auto polling PHY_ANLPAR_5.10. While forced disabling (RMII_PORT_FC_EN[1:0] == 2' b01), set self PHY devices' flow control ability register bit PHY_ANAR_4.10 as 1 (with flow control capability), and does not need auto polling PHY_ANLPAR_5.10. For forced disabling, turn off flow control enable bits of all RMII ports. For (forced) enabling, turn on flow control enable bits of all RMII ports.
SD[23:21]	**LED Serial-Out Mask for Groups 0,1,2,3 [3:0]:** LED_MASK[0] == 1' b1 => enable group 0 data out (default) LED_MASK[1] == 1' b1 => enable group 1 data out (default) LED_MASK[2] == 1' b1 => enable group 2 data out (default) LED Groups 3,4,5 serial out is always enabled. The time period of LED sequence is 50ms.
SD[24]	**Hash Algorithm Selection:** HASH_ALG_SEL == 1 => CRC-map with scramble (default) HASH_ALG_SEL == 0 => Direct-map without scramble HASH_ALG_SEL == 1 will make the register HASH_ALG[1:0]=0 that is helpful to the X-Stream test. HASH_ALG_SEL == 0 will make the register HASH_ALG[1:0]=3 that can get a good result of 4094 MAC addresses in Address Handling test.
SD[25]	**No Swap EEPROM and LED output pins:** NO_SWAP_EEPROM_LED == 1 => The pins 192/193 are used as EEIO/EEC. (default) NO_SWAP_EEPROM_LED == 0 => The pins 192/193 are used as LEDIO/LEDC. In 128-pin package, we can bond this pin to VSS to select LED output from the original EEPROM output pins.
SD[26]	**Enable Aging:** EN_AGING == 1 => enable aging function (default) EN_AGING == 0 => disable aging function
SD[27]	**Disable Cut Through Enable:** DIS_CUT_THRU == 1 => disable cut-through feature (default, i.e. store & forward) DIS_CUT_THRU == 0 => enable cut-through feature
SD[28]	**Disable Aggressive Backoff Algorithm:** DIS_AGGRES_BACKOFF == 1' b1 => disabled (default) DIS_AGGRES_BACKOFF == 1' b0 => enabled Note that if this feature is enabled, the backoff algorithm for all Ethernet ports is in MBA mode. By default, the backoff algorithm bits MBA=0 and OFSET=1.
SD[29]	**Backpreasure Collision Point Configuration:** COLL_POINT_CFG == 1 => 32th byte (default) COLL_POINT_CFG == 0 => 2nd byte
SD[30]	**Disable TEST mode:** DIS_TEST_MODE == 1 => disabled (default) DIS_TEST_MODE == 1 => enabled

SD[31]	**Disable Latchup Mode:** DIS_LATCH_UP_MODE == 1' b1 => disable Latchup mode (default) DIS_LATCH_UP_MODE == 1' b0 => enable Latchup mode, all output only IO PADs are in tri-state after reset.
SD[39:32]	**Monitored Port Bit Mask [7:0]:** For sniffer port ID = 8, monitored are ports 0~7, corresponding to bit 0~7. For sniffer port ID = 5, monitored are ports 0~4,6~8, corresponding to bit 0~7. For sniffer port ID = 4, monitored are ports 0~3,5~8, corresponding to bit 0~7. bit value 1 => not to be monitored (default) bit value 0 => to be monitored
SD[41:40]	**Sniffer Port ID Selection [1:0]:** SEL_SNIFFER_PORT[1:0] == 2' b11 => disable sniffer feature (default) SEL_SNIFFER_PORT[1:0] == 2' b10 => sniffer port ID = 8 SEL_SNIFFER_PORT[1:0] == 2' b01 => sniffer port ID = 4 SEL_SNIFFER_PORT[1:0] == 2' b00 => sniffer port ID = 5
SD[46]	**SRAM NUM:** SRAM_NUM == 1 => 1 sram chip (default) SRAM_NUM == 0 => 2 sram chip
SD[48]	**Disable Drop Control for Buffer Starvation:** DIS_DROP_STARV == 1' b1 => disabled (default) DIS_DROP_STARV == 1' b0 => enabled Note that if this feature is enabled, the drop control will drop incoming packets when free memory count <= 9.
SD[51:49]	**MII Port Configuration:** MII_PORT_CFG == 3' b111 => enable port 8 as a normal MII port without Auto-polling (forced 100Mbps, full-duplex, Enable tx/rx flow control) (default for 208-pin) MII_PORT_CFG == 3' b110 => enable port 8 as a normal MII port without Auto-polling (forced 100Mbps, full-duplex, Disable tx/rx flow control) MII_PORT_CFG == 3' b101 => enable port 8 as a normal MII port with Auto-polling speed, duplex, & party' s flow Control ability, enable self flow control ability MII_PORT_CFG == 3' b100 => enable port 8 as a normal MII port with Auto-polling speed & duplex, but force disable Self flow control ability (ignore party' s ability) MII_PORT_CFG == 3' b001 => enable port 8 as a normal MII port with Auto-polling speed & duplex, but force enable Self flow control ability (ignore party' s ability) MII_PORT_CFG == 3' b011 => enable port 8 as a reverse MII port without Auto-polling (forced 100Mbps, full-duplex, Disable tx/rx flow control) MII_PORT_CFG == 3' b010 => enable port 8 as a reverse MII port without Auto-polling (forced 100Mbps, full-duplex, Enable tx/rx flow control) MII_PORT_CFG == 3' b000 => disable port 8 (default for 128-pin package) Then, auto-polling will not work for port 8 and Port 8' s I/O control is disabled (not hold). Note: in 128-pin PQFP package, the internal strapping control module will force MII_PORT_CFG == 3' b000 without respect to latched strapping value.
SD[52]	**SRAM 32BIT MODE:** SRAM_32BIT_MODE == 1 => SRAM 32bit access (default) SRAM_32BIT_MODE == 0 => SRAM 64bit access
SD[53]	**Disable short preamble for 100Mbps ports:** DIS_SHORT_PREAM == 1' b1 => 7-byte preamble for 100M ports (default)

	DIS_SHORT_PREAM == 1' b0 => 6-byte preamble for 100M ports Note that IO Control sends {PREAM_CFG[2], PREAM_CFG[1], PREAM_CFG[0] & (SPD_10M | DIS_SHORT_PREAM)} to MAC as the preamble length.
SD[54]	**PHY ADDRESS SEQ:** PHY_ADDRESS_SEQ == 1 (default) => port0 ~ port7 PHYAD = 4~11 (i.e. 5' b00100 ~ 5' b01011) port8 PHYAD = 12 (i.e. 5' b01100) PHY_ADDRESS_SEQ == 0 => port0 ~ port7 PHYAD = 11~4 (i.e. 5' b01011 ~ 5' b00100) port8 PHYAD = 12 (i.e. 5' b01100)
SD[55]	**RMII SPEED ARBITRATE:** RMII_SPEED_ARB == 1 => arbitrate RMII speed from PHY Reg5 (ANLPAR) (default) RMII_SPEED_ARB == 0 => arbitrate RMII speed from a dedicated PHY Reg The RMII speed information is feasible even though N-way fails. Note that MII speed is derived from TX_CLK by MII2RMII module.
SD[56]	**Disable forced LED 9-port display mode:** DIS_9_PORT_LED_MODE == 1 => 8/9-port LED display mode depends on If or not Port 8 is enabled (default) DIS_9_PORT_LED_MODE == 0 => forced 9-port LED display mode, if Port 8 is disabled, its status is also displayed although maybe incorrect. This is used to facilitate validation
SD[63]	**Disable Initial Debug Mode:** DIS_INIT_DBG_MODE == 1' b1 => disable initial debug mode (default) (none test pins are used) DIS_INIT_DBG_MODE == 1' b0 => enable initial debug mode (test pins in MII port 8 are enabled for Group-0 signals about IDE)
TEST0	**Package Type:** TEST0 == 1' b1 => 128-pin package mode (default) TEST0 == 1' b0 => 208-pin package mode
TEST1	**Disable IDE in 128-pin mode:** TEST1 == 1' b1 => disable IDE in 128-pin package mode (default) TEST1 == 1' b0 => enable IDE in 128-pin package mode (only for test) Note that IDE is always enabled in 208-pin package mode.

SECTION I FUNCTIONAL DESCRIPTIONS

1 GENERAL DESCRIPTION

The VT6509 is a low-cost switch engine chip implementation of an 9 ports 10/100Mbps Ethernet switch system for IEEE 802.3 and IEEE 802.3u networks. Each port can be either auto-sensing or manually selected via EEPROM configuration to run at 10Mbps or 100Mbps speed rate, full or half duplex mode.
The VT6509 supports an 8-bit IDE bus for access by the embedded CPU to provide network management functions, e.g. SNMP, Web-management, Spanning Tree Protocol (STP), Telnet. The network management software with the above features had been constructed in both ARM-7 and x86 platforms. The VT6509 supports one port in MII interface and 8 ports in RMII (reduce MII) interface. There are 9 independent MACs within the VT6509 chip. The MAC controller controls the receiving, transmitting, and deferring of each individual port, and the MAC controller also provides framing, FCS checking, error handling, status indication and flow control function. It has wire-speed performance with forwarding rate of 148,810 packets/sec on each 100Mbps Ethernet port. The VT6509 can be configured through CPU, EEPROM, or strapping only.

2 THE VIA ETHER SWITCH ARCHITECTURE

The VT6509 switch engine uses the shared memory architecture. In order to improve the packet latency, VT6509 provides two methods for packet switching, cut-through and store-and-forwarding.

2.1 Switch initialization procedures

Step_1 Read strapping signal.
Step_2 If it is CPU initialization mode, go to step 7, else trigger SRAM Control module to start SRAM auto-test.
Step_3 If it is Strapping Only initialization mode, go to Step_4.
If initialization by EEPROM, EEPROM Control module will download register data from EEPROM.
Step_4 After SRAM auto-test completed, trigger Forward Control module to start auto aging.
Step_5 trigger PHY Control module to start phy_auto-polling.
Step_6 trigger LED Control module to display LED.
Step_7 Chip initialization done. Then, the embedded CPU can have access to VT6509.

2.2 Packet Switching Flow

1. After initialization, the input control of each enabled port will pre-allocate one packet buffer from the free buffer pool.

2. When the Receive MAC (RMAC) receives a packet data from the PHY device via the RMII network interface, it packs the data into 16-bit words then passes it to input control. If RMAC detects any error, it also notifies input control to stop forwarding process.
3. Input control extracts the destination MAC address from incoming data, passes it to forwarding table control for forwarding decision. In the mean while, it packs 16-bit words into 64-bit quad-words, and saves it to an input FIFO before storing the packet data to packet buffers in SRAM.
4. If the switch is configured to "store and forward" mode, input control queues the packet to the output queue of the destination port after input control is informed by RMAC that this is a good packet and it stores all packet data to SRAM. If the switch is configured to "cut-through" mode, the input control queues the packet to the output queue of the destination port after the first 64-byte packet data is stored in SRAM to prevent output FIFO underrun.
5. After the whole packet is received and FCS is correct, input control pass the source MAC address of the packet to forwarding table control for address learning.
6. Output control of the outbound port dequeues the packet from itself output queue, and fetches packet data from SRAM and saves it into output FIFO. Then it notifies the Transmit MAC (TMAC) to transmit the read-to-outgoing packet.
7. TMAC grabs 16-bit word at a time from output control, adds preamble and SFD to the beginning of the packet, then send them out. Proper deferring is done for collision to conform to 802.3 standard.
8. After the packet is successfully transmitted, TMAC notifies output control of the successful transmission. Output control then returns the packet to the free buffer pool.

2.3 Packet Buffers and Forwarding Table

VT6509 provides a 64-bit SRAM interface for packet buffering and maintaining address table and per-port output link lists. Each packet buffer is a 1536-byte contiguous memory block in SRAM, and it also contains to a 8-byte link node data structure in every buffer's tail. A link node corresponds uniquely to a packet buffer that could belong an output queue, the free buffer pool, incoming packet buffers of an input control, or outgoing packet buffers of an output control.

Initially, each input port control will request one packet buffer from its private buffer pool. Each time when a packet buffer is consumed by an incoming packet, the input port control will request another packet buffer to prepare for next packet.

There are two data structures in SRAM memory layout, shown in Figure 3. One is the forwarding table that locates at the lowest 32K-byte address space and contains 2048 entries (buckets), each of 2 slots. The structure of an 64-bit forwarding table slot is shown in Figure 4, composed of:

(1) bits 39..0: MAC Tag

The MAC Tag is used to record the partial content of MAC address for lookup identification. The mapping of Tag and MAC address is as follows:

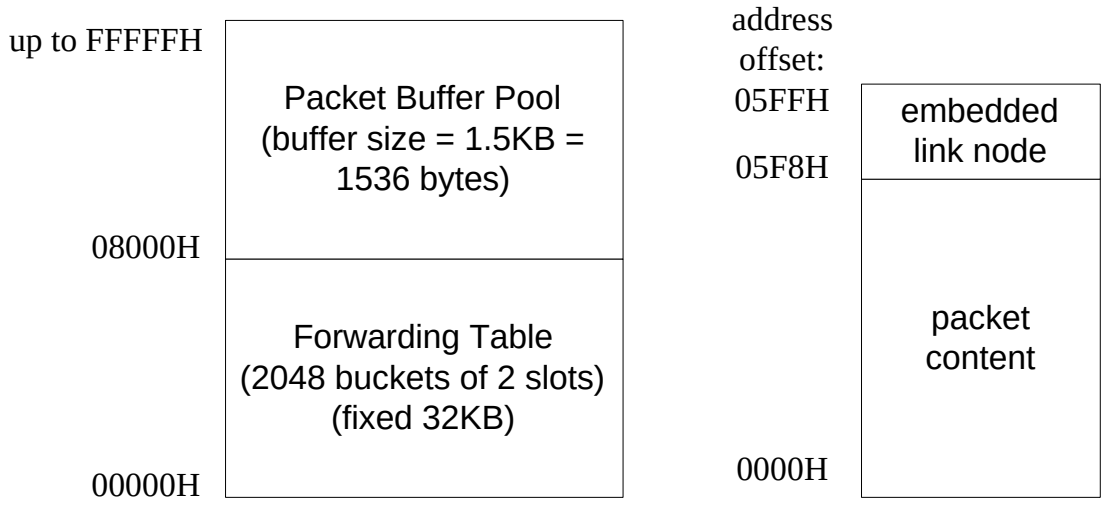

Figure 3. SRAM memory layout.

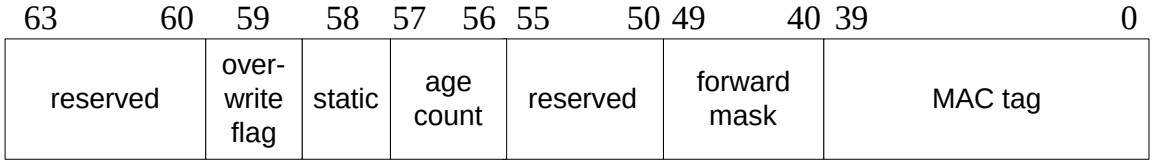

Figure 4. Data structure of forwarding table slot.

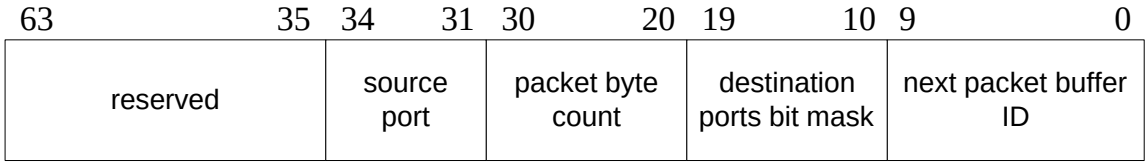

Figure 5. Data structure of embedded link node.

Tag[39:32] = MAC[15:8]
Tag[31:24] = MAC[23:16]
Tag[23:16] = MAC[31:24]
Tag[15:8] = MAC[39:32]
Tag[7:0] = MAC[47:40]

(2) bits 49..40: Forwarding Port Mask

(3) This is the bit mask of the destination ports corresponding to the forwarding table slot. The bits 49 and 48, corresponding to CPU port and MII port respectively, should be zero in VT6509.

(4) bits 57..56: Age Count

(5) If this field is equal to zero, it means this slot is an empty one; otherwise, it is valid. When the auto-aging is enabled (by default), every valid slot' age count will be decreased by one per 100 seconds.

(6) bits 58: Static Flag

(7) This flag is used to program a static MAC slot, that will not be overwritten by dynamic learning mechanism, with specified forwarding port mask for special purpose.

(8) bits 59: Over-Write Flag

During learning, if both slots are used, the 3rd address will replace one of the slots according to current value of the concatenated over-write order pointer and their static bits regardless of ages of the two slots.

Let the concatenated over-write pointer ow[1:0] = (over-write pointer of slot 1, over-write pointer of slot 0).

Let static[1:0] = {static bit of slot 1, static bit of slot 0}.

The possible mapping of ow[1:0] and static[1:0] are as follows:

	static[1:0]			
ow[1:0]	(0,0)	(0,1)	(1,1)	(1,0)
(0,0)	0	1	x	0
(0,1)	1	1	x	0
(1,1)	0	1	x	0
(1,0)	1	1	x	0

The internal operation rules for the over-write pointer are as follows:

(1) When a slot is over-written, its over-write pointer is toggled.

(2) When one of the slots is a static entry, the static entry must always be placed in slot 0, and slot 1 is always the entry over-written. Over-write pointer has no effect what so ever here.

(3) When both slots are static, no entry can be written to either slot by the learning process.

(4) When update an entry, value of the over-write pointer must not be changed unless indicated otherwise, i.e. over-writing an existing slot.

Another data structures in SRAM memory layout is the packet buffer pool, located at the upper address space. Every packet buffer is 1.5K bytes with a corresponding embedded link node in the tailing 8 bytes. The embedded link nodes are used to construct output queues, one queue for each output port. Its structure is shown in Figure 5, composed of:

(1) bits 9..0: Next Packet Buffer ID
(2) bits 19..10: Destination Ports Bit Mask
(3) bits 30..20: Packet Byte Count
(4) bits 34..31: Source Port ID

The number of packet buffers is determined by the configured SRAM size, listed as follows:

SRAM Size	128KB	256KB	512KB	1MB
maximum # of packet buffers	64 buffers	149 buffers	320 buffers	661 buffers

2.4 RMII Interface

The VT6509 can directly connect to an 8 port RMII PHY device through the reduced MII (RMII) interface to construct a small-sized system board. The signals of RMII interface are described as follows:

Name	Type	Description
CRSDV	I	Carrier sense and Data valid
RXD[0-1]	I	Receive data bit 0 to 1 , data rate with 50MHz
TXEN	O	Transmit Enable
TXD[0-1]	O	Transmit Data bit 0 to 1

2.5 MII or Reverse MII Interface

The VT6509 also provide one port of MII interface to connect a single-port PHY device as a 100BaseFX uplink. This MII port can be programmed in reverse MII mode so that it could be connected with an embedded processor with built-in 100Mbps MAC for SOHO router application or etc. The signals of MII interface are described as follows:

Name	Type	Description
RXDV	I	Data valid
CRS	I (O)	Carrier sense (for reverse MII mode)
COL	I	Collision (it should be tied to ground for reverse MII mode)
RXCLK	I (O)	Receive Clock (for reverse MII mode)
RXD[0-3]	I	Receive data bits 0 to 3 referring to RXCLK
TXCLK	I (O)	Transmit Clock (for reverse MII mode)
TXEN	O	Transmit Enable
TXD[0-3]	O	Transmit Data bits 0 to 3 referring to TXCLK

2.6 Management Interface and Auto Negotiation

The VT6509 communicates with the external 10/100M PHY and access the PHY register through MDC, MDIO**.** The signals of management interface are described as follows:

Name	Type	Description
MDC	O	management interface clock
MDIO	IO	management interface data signal

Auto-polling state machine:

step1. Auto-polling idle
- if auto-polling_command=”on” then go to step2

step2. Check the chip internal register, phyint_setup_reg, to setup the pause capability (Reg4.10) and phy auto-negotiation enable (Reg0.12)
- if (this port do not need phy setup) and (phyint_setup_reg == done), then go to step 10(done)
- else if (this port do not need phy setup) and (phyint_setup_reg != done), then config ioctl, set phyint_setup_reg == done, go to step10(done)
- else if (phyint_setup_reg != done), then go to step3(phy fc setup)
- else if (this port do not need auto-poll, then go to step 10(done)
- else go to step 5(read phy link status)

step3. Phy flow_control ability setup
- if the phy_pause_ability(reg4.10) != port_phy_fc_set, then write reg4.10 = port_phy_fc_set and go to step4 (phy auto-negotiation ability setup)
- else go to step4 (phy auto-negotiation ability setup)

step4. PHY auto-negotiation ability setup
- if (Reg0.12)=”1”(on), then modify chip internal register for phyint_setup_reg = done; go to step5
- else if (Reg0.12)=”0”(off) or phy_pause_ability have been modified (in step3), then write PHY auto-negotiation_enable bit (Reg0.12) =”1” (on), restart_auto-negotiation bit (Reg0.9) = “1” and modify chip internal register for phyint_setup_reg = done; go to step5

step5. Read PHY link_status bit (Reg1.2) 1st time
- if link is down (Reg1.2)=”0”, then go to step6
- else if link is up (Reg1.2)=”1” and chip_internal_reg for link status is “down”, then go to step7

- else if link is up (Reg1.2)=”1” and chip_internal_reg for link status is “up”, then go to step10.

step6. Read PHY link_status bit (Reg1.2) 2st time
- if link is up (Reg1.2)=”1”, then go to step7
- else if link is down (Reg1.2)=”0”, then go to step10
*if chip_internal_reg for link status is “down”=> generate link_change interrupt and disable io_port

step7. Read PHY Auto-negotiation Complete bit (Reg1.5)
- if auto-negotiation is completed (Reg1.5)=”1”, then go to step8.
- else if auto-negotiation is not completed (Reg1.5)=”0”, then go to step10.
*if chip_internal_reg for link status is “down”=> generate link_change interrupt and disable io_port

Step8. Read PHY ANAR(Reg4) and ANLPAR(Reg5)
- if speed_arbit_bit =”0”(speed from PHY proprietary reg), then go to step9
- if speed_arbit_bit =”1”(speed from ANLPAR), then go to step10
* generate link_change interrupt and enable io_port

Step9. Read PHY proprietary Reg for RMII speed, then go to step10
* generate link_change interrupt and enable io_port

Step10. Done
- if auto-polling_command = “off”, then go to step1, and stay idle
- else go to step2, for next port.

Duplex mode decision rule:

RMII_DUPLEX_MODE = ~((ANAR.8 = 1 and ANLPAR.8 =1:100BASE-TX Full)
or ((ANAR.7 =0 or ANLPAR.7 =0) and ANAR.6 =1 and ANLPAR.6=1))

Note that value 1 denotes half duplex, 0 denotes full duplex.

Speed decision rule:

According to ANLPAR,
- if ~((ANLPAR.9=1 or ANLPAR.8=1 or ANLPAR.7=1)
then speed = 10MHz

2.7 Flow Control

In VT6509, there are three policies of resource management for different objectives:
(1) flow control for full-duplex ports
Objective: slow down the input traffic without incurring any drops.
(2) backpressure for half-duplex ports

Objective: slow down the input traffic with intended collision and the Ethernet backoff mechanism.

(3) drop control for private buffer reservation

Objective: shape the input traffic to be compliant to the buffer reservation discipline.

In the following, we call "congestion control" as aggregation of the above three policies.

The application conditions of resource management policies are as follows:

(1) flow control for full-duplex ports

Enabled only when the flow control feature is enabled and the full-duplex port's party has flow control capability.

Note that as both the flow control feature and the drop control feature are enabled, the drop control feature is not turned on if the full-duplex port's party has flow control capability.

(2) backpressure for half-duplex ports

Enabled only when the flow control feature is enabled and the port is in half-duplex mode.

Note that as both the flow control feature and the drop control feature are enabled, the drop control feature is not turned on if the port is in half-duplex mode.

(3) drop control for private buffer reservation

Enabled when the flow control feature and the drop control feature are enabled, but the full-duplex port's party has not flow control capability.

Also enabled when the flow control feature is disabled and the drop control feature is enabled.

XON/XOFF Window and Drop Window

The flow control and backpressure are based on XON/XOFF window. Within XON window, the source port is in non-congestion status so that any incoming packets are forwarded normally. If an incoming packet violates the constraint of congestion control, the source port will leave the XON window and enter the XOFF window after this enqueue operation is served. Within XOFF window, the source port is in congestion status so that any incoming packet will trigger a congestion-control operation that depends on the used policy. The drop control mechanism, residing in Forwarding Control and enabled by DROP_EN = 1, is based on every ports' congestion windows that are defined as (1) any one is in XOFF window, (2) a destination port is in congestion status if its reservation buffer count == 0.

The XON/XOFF windows and the DROP ON/OFF windows are maintained by TMAC, according to port status and input signals from Queue Control. Figure 6 shows an example XOFF window status (named as XOFF_WINDOW[0]) of port 0, where XOFF[0] and XON[0] is an event generated by queue control to notify this source port should enter XOFF state, and XON[0] are events generated by queue control to notify this source port should enter or leave XOFF state, respectively. If the source port is within XOFF window as an XOFF event arrives, an flow control frame with pause time = 0xffff will be sent out. If the source port is within XON window, the arriving XON event will be ignored.

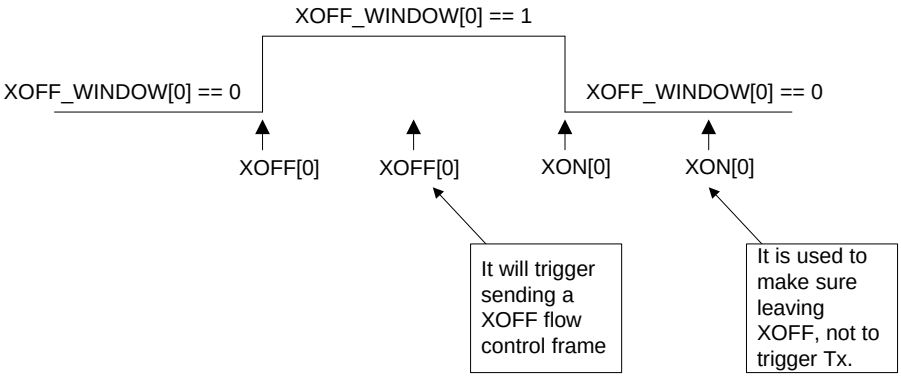

Figure 6. XON/XOFF Window Concept.

Congestion-Control Operations

The starvation control threshold is 9 in VT6509. Let Φ be the Free Memory Count. Let Ω be the Virtual Free Buffer Count, where Ω = Φ - 9. In congestion control, decisions are based on Ω. Within XOFF window, the corresponding congestion-control operations are as follows:

(1) flow control operation

As a unicast or broadcast packet, enqueued by Input Control, violates the congestion control constraints, Queue Control will assert a trigger to its source port's TMAC to send out a flow control frame with pause time = FFFFH. There are two congestion conditions: (1) $\Phi \le \max\{\Psi, 28\}$ and $R_k = 0$, (2) any port is in XOFF window and $R_k = 0$.

Note that even a source port is within XOFF window, an XOFF triggering signal from Queue Control will also make the TMAC to send out a flow control frame of pause time = FFFFH. Although it costs more bandwidth to perform the flow control operation, this duplicated transmission can guarantee the XOFF flow control frame could be received by the party successfully.

(2) backpressure for half-duplex ports

As a non-local incoming packet is received, Input Control will assert a signal of "Non_Local" to notify its TMAC sub-module about the status. TMAC will collide this packet if it is a half-duplex port and it is within XOFF window.

(3) drop control for private buffer reservation

As an incoming packet, coming from a port within DROP_ON window and destining to a port within CONGESTION window, is detected by Forwarding Control in lookup operation, Forwarding Control will unmask its port mask to make it to be dropped by Input Control. Note that the DROP_ON window indicates the time duration in which the drop feature to the specific port is enabled, that is determined by TMAC. Note that the CONGESTION window of port k is turned on when any port is in XOFF window and $R_k = 0$, that is determined by Queue Control.

XON/XOFF Thresholds Selection

In VT6509, the XON threshold is fixed and programmable. Basically, the XON threshold is equal to half of total number of buffers – 10.
The XOFF threshold is adaptive and has two objectives:

(1) Prevent buffers are exhausted by some hungry port for unbalanced traffic condition. So, XOFF threshold >= total # of buffers necessary to be reserved.

(2) Prevent packet drop because of too late response to buffer insufficience condition for balanced traffic condition. In VT6509, we assume the flow control response time to each port is 2 buffers.

The related parameters and XON/XOFF trigger conditions are as follows:

C: number of reserved buffers to prevent touching buffer starvation threshold (Virtual Free Space)
9 < C < D, default C = 10

D: offset of # buffers to tolerate incoming packets before source parties stop (Minimum XOFF Constant Threshold)
Default D = 2x9 + 9 + 1 = 28 for SRAM size of 256KB

£ XFree memory count

£ Z# of buffers necessary to be reserved for private policy.

XOFF condition:

((£ X¡ Ø max{ £ Z, D}+C)) and (R_k = 0)) or (£ X¡ Ø

XON condition:

((£ X> XON_Thred) XON_Thred = (1/2) x (MaxBufferNo + MaxXOFF - 20)

SRAM size	# of buffers	Max XOFF	Min XOFF	XON	Rmax
128KB	64	D+C = 29 D = 19 C = 10	D+C = 29 D = 15 C = 10	43	1
256KB	149	4x9+C = 46 C = 10	D+C = 38 D = 28 C = 10	98	4
512KB	320	6x9+C = 76 C = 20	D+C = 57 D = 37 C = 20	198	6
1MB	661	8x9+C=102 C = 30	D+C = 67 D = 37 C = 30	381	8

2.8 Broadcast Storm Filtering

In VT6509, the feature of broadcast storm filtering is based on the above congestion control, and subjects to incoming broadcast packets. When an incoming broadcast packet is first enqueued by the Input Control of the source port, Queue Control will check if any one of its destination output ports is in XOFF window. If yes, it will trigger flow control to source port. For example, if a broadcast packet comes in from

port 0, it will be forwarded to port1, port2, port3, port4, ... port8 in turn. If any one of ports 1~8 is congested, it will trigger flow control to source port 0, right on it is received and enqueued by Input Control of port 0.
If any port's drop window is turned on, Forwarding Control will drop the incoming broadcast packets to restrict the resource allocation.

2.9 Serial EEPROM interface and Configuration commands

If the chip initialization mode is "EEPROM-init" (by strapping), VT6509 will read the external EEPROM device through EEC, EEIO during the system initialization**.** The signals of EEPROM interface are described as follows:

Name	Type	Description
EEC	O	serial EEPROM interface clock
EEIO	IO	serial EEPROM interface data signal

There are two types of EEPROM commands with the following format:

1. EOF Command: this must be the last command in EEPROM.
   EE_DATA[7:0] = 8'b0000_0000
2. Download Command:
   - EE_DATA[7:6] = 2'b01 ➔ eeprom download data for a specific address
   - EE_DATA[5:0] ➔ sequentially download data length from the following address
   - The next 2 EE_DATA will be Register base and offset address

Note that there must be an 8-bit check sum following the EOF command to make the sum of all commands and this check-sum byte is equal to zero bits 0000-0000.

EEPROM memory layout:

EEPROM Address	Data[7:0]
00H *00H+1* *00H+2* *00H +3 ~* *00H+2+1st_Data_Length(n)*	*- 1st Command[1:0](2'b01) + 1st_Data_Length[5:0] (1 <= n <= 63)* *- CS_Module[7:0]* *- IO_Write Start Address* *- IO_Write_DATA[7:0] *n*
00H+3+1st_Data_Length(n)	- 2nd Command
	- Ending Command[7:0]=8'b0000_0000 ➔ End_of_file
	Check_Sum

Note that EEPROM device id and address must be 7'b1010_001.

2.10 Trunking

There are two trunk groups in VT6509, one is port (0,1), another is port (6,7). Each one can be enabled individually. The trunking load balance algorithm is as follows:

Assume DMAC=12'h000000000002, source_port_id = 4'b0001.

DMAC_forwarding_entry_address = hash algorithm(DMAC) * 2 + resident slot#
(hask key) (0 or 1)

Assume the above value is 12'b000000011010 after lookup for DMAC = 12'h000000000002.

Let dest_bit = DMAC_forwarding_entry_address[4] xor source_port_id[0].
If the DMAC refers to a trunk port of the trunk group (0,1), the selected outgoing trunk port is port 0 + dest_bit.
If the DMAC refers to a trunk port of the trunk group (6,7), the selected outgoing trunk port is port 6 + dest_bit.

3 THE VT6509 SRAM ADDRESS MAPPING TABLE

208 pin/64bit mode/2Bank, (64KX64)X2

Internal 64bit Linear Address	6509 Pad(64bit mode)	6509 Package Address Pin	SRAM Address Pin	SRAM chip select
IA16	SA16 = IA16	SA16 = IA16	N/A	Chip 1 : CE_= 0, CE2_ = SA16, CE2 = 1 Chip 2 : CE_ = 0, CE2_ = 0, CE2 = SA16
IA15	SA15 = IA15	SA15 = IA15	A15 = IA15	
IA14	SA14 = IA14	SA14 = IA14	A14 = IA14	
IA13	SA13 = IA13	SA13 = IA13	A13 = IA13	
IA12	SA12 = IA12	SA12 = IA12	A12 = IA12	
IA11	SA11 = IA11	SA11 = IA11	A11 = IA11	
IA10	SA10 = IA10	SA10 = IA10	A10 = IA10	
IA9	SA9 = IA9	SA9 = IA9	A9 = IA9	
IA8	SA8 = IA8	SA8 = IA8	A8 = IA8	
IA7	SA7 = IA7	SA7 = IA7	A7 = IA7	
IA6	SA6 = IA6	SA6 = IA6	A6 = IA6	
IA5	SA5 = IA5	SA5 = IA5	A5 = IA5	
IA4	SA4 = IA4	SA4 = IA4	A4 = IA4	

IA3	SA3 = IA3	SA3 = IA3	A3 = IA3	
IA2	SA2 = IA2	SA2 = IA2	A2 = IA2	
IA1	SA1 = IA1	SA1 = IA1	A1 = IA1	
IA0	SA0 = IA0	SA0 = IA0	A0 = IA0	

208 pin/64bit mode/1Bank, 64KX64, (32KX32)X2, (64KX32)X2, (128KX32)X2

Internal 64bit linear Address	6509 Pad(64bit mode)	6509 Package Address Pin	SRAM Address Pin	SRAM chip select
IA16	SA16 = IA16	SA16 = IA16	A16 = IA16 N/A for (32kx32)x2, 64kx64, (64kx32)x2	CE_ = 0 CE2 = 1 CE2_ = 0
IA15	SA15 = IA15	SA15 = IA15	A15 = IA15 N/A for (32kx32)x2	
IA14	SA14 = IA14	SA14 = IA14	A14 = IA14	
IA13	SA13 = IA13	SA13 = IA13	A13 = IA13	
IA12	SA12 = IA12	SA12 = IA12	A12 = IA12	
IA11	SA11 = IA11	SA11 = IA11	A11 = IA11	
IA10	SA10 = IA10	SA10 = IA10	A10 = IA10	
IA9	SA9 = IA9	SA9 = IA9	A9 = IA9	
IA8	SA8 = IA8	SA8 = IA8	A8 = IA8	
IA7	SA7 = IA7	SA7 = IA7	A7 = IA7	
IA6	SA6 = IA6	SA6 = IA6	A6 = IA6	
IA5	SA5 = IA5	SA5 = IA5	A5 = IA5	
IA4	SA4 = IA4	SA4 = IA4	A4 = IA4	
IA3	SA3 = IA3	SA3 = IA3	A3 = IA3	
IA2	SA2 = IA2	SA2 = IA2	A2 = IA2	
IA1	SA1 = IA1	SA1 = IA1	A1 = IA1	
IA0	SA0 = IA0	SA0 = IA0	A0 = IA0	

208 pin/32bit mode/1Bank, 32KX32, 64KX32, 128KX32

Internal 64bit linear Address	6509 Pad(64bit mode)	6509 Package Address Pin	SRAM Address Pin	SRAM chip select
IA16	SA16 = IA15	SA16 = IA15	A16 = IA15 N/A for 32kx32, 64kx32	CE_ = 0 CE2 = 1 CE2_ = 0
IA15	SA15 = IA14	SA15 = IA14	A15 = IA14 N/A for 32kx32	
IA14	SA14 = IA13	SA14 = IA13	A14 = IA13	
IA13	SA13 = IA12	SA13 = IA12	A13 = IA12	
IA12	SA12 = IA11	SA12 = IA11	A12 = IA11	
IA11	SA11 = IA10	SA11 = IA10	A11 = IA10	
IA10	SA10 = IA9	SA10 = IA9	A10 = IA9	
IA9	SA9 = IA8	SA9 = IA8	A9 = IA8	
IA8	SA8 = IA7	SA8 = IA7	A8 = IA7	
IA7	SA7 = IA6	SA7 = IA6	A7 = IA6	
IA6	SA6 = IA5	SA6 = IA5	A6 = IA5	

IA5	SA5 = IA4	SA5 = IA4	A5 = IA4	
IA4	SA4 = IA3	SA4 = IA3	A4 = IA3	
IA3	SA3 = IA2	SA3 = IA2	A3 = IA2	
IA2	SA2 = IA1	SA2 = IA1	A2 = IA1	
IA1	SA1 = IA0	SA1 = IA0	A1 = IA0	
IA0	SA0 = 0	SA0 = 0	A0 = SA0 = 0	

SECTION II REGISTER MAP

1. REGISTERS TABLE

base or offset	Register Description	Name	bits	Default Value	R/W
0000H	Chip Configuration				
00-07H	Chip Configuration by strapping: CHIP_CFG[0]: BCAST_BPDU CHIP_CFG[1]: LED_COMB [0] CHIP_CFG[2]: DROP_CONTROL_EN CHIP_CFG[3]: SRAM_TYPE[1] CHIP_CFG[5:4]: CHIP_INIT_MODE[1:0] CHIP_CFG[6] EN_JABBER_CTL CHIP_CFG[7] RETRY_EXCE_COLL CHIP_CFG[15:8]: RMII_SPEED_CFG[7:0] CHIP_CFG[16] TRUNK_MODE[0] CHIP_CFG[17] TRUNK_MODE[2] CHIP_CFG[18]: EN_BACKPRESSURE [0] CHIP_CFG[20:19]: RMII_PORT_FC_EN CHIP_CFG[23:21]: LED_CFG[2:0] CHIP_CFG[24]: HASH_ALG_SEL CHIP_CFG[25]: NO_SWAP_EEPROM_LED CHIP_CFG[26]: EN_AGING CHIP_CFG[27]: DIS_CUT_THRU CHIP_CFG[28]: DIS_AGGRES_BACKOFF CHIP_CFG[29]: COLL_POINT CHIP_CFG[30]: DIS_TEST_MODE CHIP_CFG[31] DIS_LATCH_UP_MODE CHIP_CFG[39:32]: MONITOR_PORT[7:0] CHIP_CFG[41:40]: SNIFFER_PORT[1:0] CHIP_CFG[43]: TRUNK_MODE[1] CHIP_CFG[45]: TRUNK_MODE[3] CHIP_CFG[42]: RMII_SPEED_CFG[8] CHIP_CFG[44]: RMII_SPEED_CFG[9] CHIP_CFG[46]: SRAM_NUM CHIP_CFG[47]: SRAM_TYPE[0] CHIP_CFG[48]: DIS_DROP_STARV CHIP_CFG[51:49]: MII_PORT_CFG CHIP_CFG[52]: SRAM_32BIT_MODE CHIP_CFG[53] DIS_SHORT_PREAM CHIP_CFG[54]: PHY_ADDRESS_SEQ CHIP_CFG[55]: RMII_SPEED_ARB CHIP_CFG[56]: DIS_9_PORT_LED_MODE CHIP_CFG[62:57](reserved with value 1' s) CHIP_CFG[63] DIS_INIT_DBG_MODE Note that LED_CFG and LED_COMB can be programmed on-the-fly by CPU to display the corresponding port status. The other register bits are also programmable by CPU, but they only take effect after soft reset.	CHIP_CFG	[63:0]	by strapping	R/W
0AH	Enable Normal Operation of LED Serial-Out The normal operation (periodic data sending) of LED Serial-Out can be triggered by (1) EEPROM-init complete signal from EEPROM-init module, or (2) software writes value 1 to this register.	EN_LED_OUT	[0]	0	R/W
10H	CPU Soft Reset for the whole switch chip reset For Read, 0: soft reset in progress	CPU_SOFT_RESET	[0]	1	R/W

	1: soft reset done For Write, any value will trigger the whole chip reset. The soft reset is similar to power-on reset for the switch chip, except that it is asserted by writing any value to this register. The CPU soft reset has to take 16 RCLK50 cycles, i.e. 320ns.				
0100H	**Queue control**				
00-01H	Congestion Control Xon Threshold register	XON_THRED	~~[9:0]~~	by strapping	R/W
02H	Output Private Buffer Size	PRIVATE_BUF_SIZE	[4:0]	by strapping	R/W
03H	Selected Output Queue ID	QUEUE_ID	[3:0]	0	R/W
04-05H	Selected Output Queue Length	OUEUE_LENGTH	[9:0]	0	R/W
06-07H	Selected Output Queue Head Pointer	OUEUE_HEAD	[9:0]	3FF (null)	R/O
08-09H	Virtual Free Space	V_FREE_SPACE	[9:0]	by SRAM size	R/W
0A-0BH	Minimum XOFF Constant Threshold	MIN_XOFF_CONST_THRED	[9:0]	by SRAM size	R/W
0200H	**Buffer control**				
00-01H	Free Memory Block Count	FREEMCNT	[9:0]	by SRAM size	R/W
02H	Memory Allocation Bit Mask Byte ID	MASK_ID	[6:0]	0	R/W
03H	Memory Allocation Bit Mask	BUFFER_MASK	[7:0]	0	R/O
0300H	**Forwarding table control**				
00H	sniffer port id	SNIFFER_PID	[3:0]	by strapping	R/W
01-02H	monitor ports bit mask	MONITOR_PM	[8:0]		R/W
03H	monitor ports bit mask 0 => to monitor input&output traffic (default) 1 => to monitor input traffic only 2 => to monitor output traffic only	MONITOR_MODE	[1:0]	0	R/W
04H	auto-aging configuration 0: disable auto-aging (default) 1: enable auto-aging Note: Initialization Control should enable auto-aging for strapping-only initialization.	EN_AUTO_AGE	[0]	0	R/W
05H	inter-aging time (in millisecond) The valid value is 1..255. The default inter-aging time value is 50ms for an entry by default strapping so that the forwarding table of 2K entries (4K slots) can be aged completely in about 5 minutes.	INTER_AGE_TIME	[7:0]	50	R/W
06-07H	aging index 06H: AGING_INDEX[7:0] 07H: AGING_INDEX[10:8] for CPU to trigger aging on specific entry	AGING_INDEX	[10:0]	0	R/W
08H	aging status 0: idle or complete 1: in progress (no matter how the process is triggered by auto-init or by CPU) for CPU to detect status of aging process	AGING_STATUS	[0]	0	R/W
09H	hash algorithm HASH_ALG[0] : 0, crc11, 1, direct map HASH_ALG[1] : 0, scramble, 1, no scramble The default setting 2b' 00 is optimized specially for Smartbit test, each combination may just be as good as others.	HASH_ALG	[1:0]	0 if strapping SD[24]=1, 3 if strapping SD[24]=0	R/W

	HASH_ALG[1:0]=2b' 11: direct map without scramble, hash key = MAC[10:0] HASH_ALG[1:0]=2b' 01: direct map with scramble, hash key = MAC[7:0, 16:18] HASH_ALG[1:0]=2b' 10: CRC map & no scramble, hash key = CRC11{MAC[47:0]} HASH_ALG[1:0]=2b' 00: CRC map with scramble, hash key = CRC11{MAC[47:24,15:0,16:23]} The forwarding table entry format is as follows. [58] static [57:56] age count [49:40] forwarding portmask [39:0] tag Regardless of the hash algorithm, the tag will always use MAC[47:8]. The mapping is as follows. Tag[39:32] ←→ MAC[15:8] Tag[31:24] ←→ MAC[23:16] Tag[23:16] ←→ MAC[31:24] Tag[15:8] ←→ MAC[39:32] Tag[7:0] ←→ MAC[47:40]				
0A-0FH	destination MAC object 0AH: DMAC[47:40], 0BH: DMAC[39:32], .,. 0FH: DMAC[7:0]. A write to this register will make the corresponding CRC-map lower 11-bit hash key and upper 37-bit tag shown in HASH_CONTENT.	DMAC_OBJ	[47:0]	0	R/W
10	calculation status of DMAC' s hash key The HASK_CAL_STATUS register indicates if the content in HASH_KEY registers are valid. 1 for calculation in progress, 0 for calculation complete. The CRC-map result of DMAC_OBJ is shown in this register, 11-bit hash key. The corresponding tag is always DMAC[47:9].	HASH_CAL_STATUS	[0]	0	R/W
11-12H	hash key of DMAC_OBJ The CRC-map result of DMAC_OBJ is shown in this register, 11-bit hash key. The corresponding tag is always DMAC[47:9].	HASH_RESULT	[10:0]	0	R/W
13-14H	port mask for lookup-miss counter 1: to be accounted into LOOKUP_MISS counter 0: not to be accounted LOOKUP_MISS_MASK [7:0] at 13H. LOOKUP_MISS_MASK [9:8] at 14H.	LOOKUP_MISS_MASK	[9:0]	1FF	R/W
15-16H	lookup-miss counter 15H: LOOKUP_MISS [7:0] 16H: LOOKUP_MISS [15:8] This counter is cleared automatically after CPU read.	LOOKUP_MISS	[15:0]	0	R/W
17-18H	hash-conflict counter 17H: HASH_CONFLICT_CNT [7:0] 18H: HASH_CONFLICT_CNT [15:8] This counter is cleared automatically after CPU	HASH_CONFLICT_CNT	[15:0]	0	R/W

	read.				
19-1AH	destination port mask for lookup-miss unicast packets 19H: UCAST_DPM [7:0] for ports 7..0 1AH: UCAST_DPM [9:8] for cpu port & port 8 for unicast packet lookup miss broadcast control	UCAST_DPM	[9:0]	01FF	R/W
1B-1CH	destination port mask for lookup-miss multicast packets 1BH: MCAST_DPM [7:0] 1CH: MCAST_DPM [9:8] for multicast packet lookup miss broadcast control	MCAST_DPM	[9:0]	01FF	R/W
1DH	selection of CPU_PM usage 0: select cpu port to use CPU_PM for lookup 1: select MII port to use CPU_PM for lookup	CPU_PM_USER_SEL	[0]	0	R/W
1E-1FH	Destination Ports Bit Mask for Port 8 (CPU port) 0: no affection on port 8' s incoming packets that will be forwarded with lookup not 0: port 8' s next incoming packets will be forwarded by CPU_PM, rather than lookup. CPU_PM will be set as ' 0' after processing exact ONE port 8' s incoming packet. Note that this function can facilitate the implement of some special system features, e.g. production testing utility, boot diagnosis utility.	CPU_PM	~~[8:0]~~	0	R/W
20H	Reserved Special Internal Control bit 0 –disable early cpu port broadcast filtering The default value 0 is to enable this feature. bit 1 – DISABLE_FAILOVER_PORTS01 The default value 0 is to enable this feature. bit 2 – DISABLE_FAILOVER_PORTS67 The default value 0 is to enable this feature. bit 3 – RUNTIME_TRUNKING_ ENABLE 0: disabled, 1: enabled bit 4 – DROP_CONGESTION_WHEN_STARV bit 6:5 – SWITCH_MAC_MATCH_MODE bit 7 – LOOPUP_OPTION Note that only the bits 0,1,2 are used for normal operation setting, the other bits are used only in Verilog simulation.	SPECIAL_CTL		0	R/W
21H	**CPU port related forwarding configuration** bit 0 – enable forwarding broadcast packets with DMAC=0xffffffffffff to CPU (default = 0 : disable) bit 1 – enable forwarding spanning-tree packets to CPU (default = 0 : disable) bit 2 – enable forwarding unicast packets with DMAC = switch MAC base to CPU (default = 0 : disable) bit 3 – enable forwarding multicast packets with DMAC = 01-80-C2-00-00-02~0F to CPU (default = 0 : disable forwarding to CPU port, i.e. drop these packets) Note that the USER_PM in VT3061 is changed as 10-bit signal from 10 output port enabling bits. The three register bits are used to enable/disable	CPU_FWD_CFG	[3:0]	0	R/W

	forwarding the above three types of frames to the CPU port. If CPU_FWD_CFG[1]=1, i.e. forwarding spanning-tree packets to CPU is disabled, these SPT packets will be broadcasted, just like the case for Ethernet Hub.				
22-23H	**Drop On** Value 1: the corresponding port' s drop feature is enabled. Any incoming packets destining to any one port in congestion will be dropped by Forwarding Control by setting port mask = 0. Value 0: the port' s drop feature is disabled. This signal is from TMAC.	DROP_ON	[8:0]	0	R/O
24-25H	**Congestion Window On** Value 1: the port is in congestion status. Value 0: the port is not in congestion status. This signal is from Queue Control.	CONGEST_ON	[9:0]	0	R/O
26-27H	**XOFF Window** Value 1: the port is in XOFF window. Value 0: the port is in XON window. This signal is from TMAC.	XOFF_WINDOW	[9:0]	0	R/O
30H	**spanning tree state for PORT 0** 2' b00 - blocking state 2' b01 – listening state 2' b10 - learning state 2' b11 – forwarding state (default) The default value of all STP states is "forwarding state". The forwarding operation in each Ethernet port is controlled by its associated spanning tree state. In blocking or listening state, the incoming packets will not trigger any DMAC lookup operation and SMAC learning operation. In learning state, the incoming packets will not trigger DMAC lookup operation, but the SMAC learning operation will be triggered for CRC-OK packets. Only in forwarding state, an incoming packet will trigger DMAC lookup operation while the first 24 bytes are received, and it will trigger the SMAC learning operation while the whole packet is received with good CRC. For the 802.1d spanning tree algorithm, a blocked port for loop avoidance should enter the blocking state so that any incoming packets are filtered without forward. A normal port that does not cause any loop should be in the forwarding state.	PORT0_STP_STATE	[1:0]	3	R/W
31H	**spanning tree state for PORT 1**	PORT1_STP_STATE	[1:0]	3	R/W
32H	**spanning tree state for PORT 2**	PORT2_STP_STATE	[1:0]	3	R/W
33H	**spanning tree state for PORT 3**	PORT3_STP_STATE	[1:0]	3	R/W
34H	**spanning tree state for PORT 4**	PORT4_STP_STATE	[1:0]	3	R/W
35H	**spanning tree state for PORT 5**	PORT5_STP_STATE	[1:0]	3	R/W
36H	**spanning tree state for PORT 6**	PORT6_STP_ST	[1:0]	3	R/W

		ATE			
37H	**spanning tree state for PORT 7**	PORT7_STP_STATE	[1:0]	3	R/W
38H	**spanning tree state for PORT 8**	PORT8_STP_STATE	[1:0]	3	R/W
0400H	**PHY control**				
00H	CPU command port ID Normally, it is the working port ID for CPU direct read/write. It also shows the working port ID for auto-polling debugging mode.	CMD_PORTID	[3:0]	0	R/W
01H	CPU command PHY REG number	PHY_REG_ADDR	[4:0]	0	R/W
02-03H	CPU command R/W data The last data word read from PHY is stored in this register for CPU read. Note that the last data word written to PHY is stored in an internal register that is also located in this address for CPU write, but not readable.	PHYDATA	[15:0]		R/W
04H	CPU command register PHYCMD[3]: enable phy auto-polling PHYCMD[2]: disable phy auto-polling PHYCMD[1]: phy read command. PHYCMD[0]: phy write command. When auto-polling mechanism is enabled, CPU can not read/write PHY devices, even though the PHY (auto-polling) Debug Mode is enabled.	PHYCMD	[3:0]		W/O R/W
05H	PHY status register Bit-0: 0=cmd_idle; 1=cmd_busy Bit-1: 0=cmd_incomplete; 1=cmd_complete Bit-2: 0=autopoll_off; 1=autopoll_on Note that the "cmd" means the command from CPU interface, not from auto-polling control.	PHYSTS	[2:0]	0	R/O
06-07H	Link status change 1' b1=link status changed 1' b0=link status without changed (cleared after cpu read)	LINK_STATUS_CHANGE	[8:0]	0	R/O
08-09H	Link status 1' b0=link down (default) 1' b1=link up	LINK_STATUS	[8:0]	0	R/O R/W
10-18H	PHY Device Address	PHY_ADDR	[4:0]	0	R/W
0500H	**EEPROM control**				
00H	EEPROM word address For a 256-byte EEPROM device, an 8-bit data object is identified with this register. For a 512-byte EEPROM device, an 8-bit data object is identified with this register plus EEDEVADDR[1], vice versa.	EEWDADDR	[7:0]		R/W
01H	EEPROM data The last data byte read from EEPROM is stored in this register for CPU read. Note that the last data byte written to EEPROM is stored in an internal register that is also located in this address for CPU write, but not readable.	EEDATA	[7:0]		R/W
02H	EEPROM device address bit 7-4 : device type id (EEPROM 1010) bit 3-1 : device id bit 0 : r/w command → value 0: write; value 1: read	EEDEVADDR	[7:0]		R/W

03H	EEPROM status register EESTS[0]: value 0: idle, 1: busy (r/w or auto-init in progress) EESTS[1]: value 1: r/w complete without error (cleared by register read) EESTS[2]: value 1: r/w ack error EESTS[3]: value 0: no init error, 1: init err	EESTS	[3:0]		R/O
0600H	**CPU interface**				
00H	interrupt status/clear register bit 0: PHY command complete interrupt bit 1: EEPROM command interrupt bit 2: PHY auto-polling link status change notification bit 3: EEPROM initialization complete interrupt bit 4: interrupt indication for CPU IO port receiving an incoming packet from some Ethernet port, that is ready to be received by CPU via IDE. bit 5: interrupt indication for CPU IO port finishing the transmission of an outgoing packet that was sent by CPU via IDE. bit 6: interrupt indication for re-autonegotiation caused by port speed mismatch Note that writing 1 to the corresponding register bits will clear that interrupts.	IRQSTS	[6:0]	0	R/W
01H	interrupt mask register bit 0: PHY command complete interrupt mask bit 1: EEPROM command interrupt mask bit 2: PHY auto-polling link status change notification mask bit 3: EEPROM initialization complete interrupt mask bit 4: interrupt mask for CPU IO port receiving an incoming packet from some Ethernet port, that is ready to be received by CPU via IDE. bit 5: interrupt mask for CPU IO port finishing the transmission of an outgoing packet that was sent by CPU via IDE. bit 6: interrupt indication for re-autonegotiation caused by port speed mismatch Note: value 0 means “masked”, value 1 means “not masked”. The default is “masked” so that software has to poll IRQSTS to check if there are any pending interrupts.	IRQMASK	[6:0]	0	R/W
02H-04H	sram address register The data object addressed is in unit of 64 bits. The maximum allowable SRAM size is 1MB. For SRAM direct access, the CPU has to (1) set SRAMADDR & SRAMDATA, (2) issue read/write command by SRAMCMD, and (3) check command status by SRAMSTS.	SRAMADDR	[16:0]		R/W
05H-0CH	sram data register	SRAMDATA	[63:0]		R/W
0DH	sram command register 2’ b00 : nop 2’ b01 : read 2’ b10 : write	SRAMCMD	[1:0]		R/W
0EH	sram status register 2’ b01 : read/write command done	SRAMSTS	[1:0]	0	R/O

	2' b10 : busy (read/write in progress) 2' b00 : idle				
0FH	Interrupt Mode INTRQ_MODE == 0 => High Active INTRQ_MODE == 1 => Low Active (default)	INTRQ_MODE	[0]	1	R/W
10H	**Write packet command** 3' b100 : end of frame with the remaining data size = 2 byte 3' b101 : end of frame with the remaining data size = 1 byte 3' b000 : idle 3' b001 : start of frame for the next write 3' b010 : middle of frame for the next write 3' b011 : abort the unfinished packet write CPU should write this command register before repeatedly writing 8 bit packet data to CPU Interface Control via the 8-bit IDE bus. The cpu interface with external is 8bit bus. However the interface between cpuif and cpuioctl is 16bit data bus. The WR_PKT_CMD will be passed to cpuioctl so that it will know all the 16bit are valid or only 8 bit are valid data. For cpu to send packet, the steps are as follows: write WR_PKT_CMD = 3'b001 to notify start of frame, then write first 2 byte data via IDE. write WR_PKT_CMD = 3'b010 to notify middle of frame, then write even number of data via IDE. Leave last two or one byte data. write WR_PKT_CMD = 3'b100 or 3'b101 to notify end of frame with remaining data 2 byte or 1 byte. If it is 2 byte, write the last two byte data via IDE. If it is 1 byte, write the last one byte data first, and then write another byte for dummy padding.	WR_PKT_CMD	[2:0]		W/O R/W
11H	**Packet Abort** Writing 1' b1 to this register will drop an incoming packet ready to be read by CPU. Reading this register will clear this register as 1' b0.	PKT_ABORT	[0]		R/W
12H	**Packet Data Register** Write this register (need to cooperate with WR_PKT_CMD at 0610H and WR_PKT_STATUS at 1910H) to send an incoming-to-switch packet to CPU Input Control. Read this register (need to cooperate with PKT_BYTE_CNT at 1900-1901H and RD_PKT_STATUS at 1902H) to retrieve an outgoing-from-switch packet from CPU Output Control. Note that any read or write to this register will not cause auto-increment of the Address Register to support the continuous packet-byte read/write commands. The cpu interface with external is 8bit bus. However the interface between cpuif and cpuioctl is 16bit data bus. Reading packet data will be served by cpuioctl so that it will know all the 16bit are valid or only 8 bit are valid data. For cpu to read packet, please read one more dummy byte after reading the last byte if the packet length is	PKT_DATA	[7:0]		~~R/W~~

	even.				
13H	**bits [47:40] of switch MAC address [47:0]**	SWITCH_MAC	[7:0]	00H	R/W
14H	**bits [39:32] of switch MAC address [47:0]**	SWITCH_MAC	[7:0]	40H	R/W
15H	**bits [31:24] of switch MAC address [47:0]**	SWITCH_MAC	[7:0]	63H	R/W
16H	**bits [23:16] of switch MAC address [47:0]**	SWITCH_MAC	[7:0]	80H	R/W
17H	**bits [15:8] of switch MAC address [47:0]**	SWITCH_MAC	[7:0]	00H	R/W
18H	**bits [7:0] of switch MAC address [47:0]**	SWITCH_MAC	[7:0]	00H	R/W
19H	**trigger/check SRAM auto test** Write to this register will trigger the SRAM auto-test function. The status of execution of the SRAM auto-test function is reported in this register, defined as follows: bit 2: 1 → in progress, 0 → complete or idle bit 1: 1 → error in single r/w connectivity, 0 → no error in single r/w connectivity bit 0: 1 → error in burst r/w connectivity, 0 → no error in burst r/w connectivity	SRAM_AUTO_TEST	[2:0]	0	R/W
20H	Revision ID	REVISION_ID	[7:0]	0	R/O
1000H	**MAC & I/O Control Module of Port 0**				
00H	configurable preamble bytes	PREAM_CFG	[2:0]	7	R/W
01H	configurable frame gap in di bits for 1st interval	IFG_CFG	[5:0]	32	R/W
02H	Backoff configuration bit 0: CAP mode → DEC' s capture effect (default: CAP=0) bit 1: MBA mode → HP' s capture effect (default: MBA=0) bit 2: EEFAST mode → drop the 2nd collided packet for testing purpose, accelerate the drop event (default: EEFAST=0) bit 3: CRANDOM mode → use another random algorithm (default: CRANDOM=0) bit 4: OFSET → parameter for backoff timer, For OFSET=1, the TMAC will follow the 802.3 standard backoff algorithm. For OFSET=0, the TMAC will select the backoff time for the 1st and 2nd collision as that of the 3rd collision, i.e. the possible the backoff time for the 1st and 2nd collision is ranged from 0 to 7 in unit of slot time. (default: OFSET=1)	BOFFCFG	[4:0]	5' b10000	R/W
03H	MAC media type configuration bit 0: SPD_10M → value 0: 100Mbps, value 1: 10Mbps bit 1: HALF_DPX → 1: half duplex, 0: full duplex bit 2: RCV_FC_DIS → 1: disable receive flow control frame 0: enable receive flow control frame bit 3: XMT_FC_DIS → 1: disable send flow control frame for full-duplex mode 0: enable send flow control frame Note: this is configured automatically by auto-polling control or CPU if auto-polling is disabled to that port. There is no forced mode if auto-polling is enabled to that port.	MACCFG	[3:0]	0	R/W
04H	IO port enable bit 0: input port enable 0: input disabled, (drop incoming packets) 1: input enabled bit 1: output port enable	IO_CFG	[2:0]	0	R/W

	0: output disabled (default) 1: output enabled bit 2: output port dequeue hold 1: dequeue disabled 0: dequeue enabled (default)				
05-06H	Granted 1st free buffer Addr in Input Control bit [10:0]: buffer address (ID*3 + 40h) bit [11]: buffer address valid (default: invalid) bit [12]: buffer free (available for the new incoming packet)	FREE_BUF_ADDR1	[12:0]	0	R/O
07-08H	Granted 2nd free buffer Addr in Input Control bit [10:0]: buffer address (ID*3 + 40h) bit [11]: buffer address valid (default: invalid) bit [12]: buffer free (available for the new incoming packet)	FREE_BUF_ADDR2	[12:0]	0	R/O
09-0AH	Event mask for counters EVENT_MASK[0]: rx CRC good packets for Counter 1 EVENT_MASK[1]: rx flow control frames for Counter 1 --- EVENT_MASK[2]: rx a runt packet for Counter 2 EVENT_MASK[3]: rx an over-size packet for Counter 2 EVENT_MASK[4]: rx a regular-size but CRC error packet for Counter 2 --- EVENT_MASK[5]: FIFO over-run without buffer starvation for Counter 3 EVENT_MASK[6]: FIFO over-run with buffer starvation for Counter 3 EVENT_MASK[7]: intended drop local packets for Counter 3 EVENT_MASK[8]: drop broadcast packets due to congestion for Counter 3 EVENT_MASK[9]: drop unicast packets due to congestion for Counter 3 --- EVENT_MASK[10]: tx packets excluding flow control packets for counter 4 EVENT_MASK[11]: tx flow control packets for counter 4 --- EVENT_MASK[12]: FIFO under-run for Counter 5 EVENT_MASK[13]: collision for Counter 5 EVENT_MASK[14]: first 16-bit packet data late (can not catch up preamble) for counter 5	EVENT_MASK	[14:0]	7FFF	R/W
10H-11H	received good packet count (Counter 1) It is cleared automatically after CPU read.	RCV_GOOD_PKT	[15:0]	0	R/O
12H-13H	received bad packet count (Counter 2) It is cleared automatically after CPU read.	RCV_BAD_PKT	[15:0]	0	R/O
14H-15H	drop packet counter(Counter 3) It is cleared automatically after CPU read.	DROP_PKT	[15:0]	0	R/O
16H-17H	sent good packet count(Counter 4) ~~It is cleared automatically after CPU read.~~	XMT_GOOD_PKT	[15:0]	0	R/O
18H-19H	sent bad packet counter(Counter 5) It is cleared automatically after CPU read.	XMT_BAD_PKT	[15:0]	0	R/O

1AH-1BH	sent abort packet counter(Counter 6) (abort due to excessive transmission fails) It is cleared automatically after CPU read.	XMT_ABORT_PKT	[15:0]	0	R/O
1C-1DH	dequeued 1st buffer Addr in Output Control bit [10:0]: buffer address (ID*3 + 40h) bit [11]: buffer address valid (default: invalid)	DEQUEUED_BUF_ADDR1	[11:0]	0	R/O
1E-1FH	dequeued 2nd buffer Addr in Output Control bit [10:0]: buffer address (ID*3 + 40h) bit [11]: buffer address valid (default: invalid)	DEQUEUED_BUF_ADDR2	[11:0]	0	R/O
20H	Internal state of Input Control (reserved for Weipin' s definition) (only valid for port 0)	INPUT_CTL_STATE	[7:0]	0	R/O
21H	Internal state of Output Control (reserved for Weipin' s definition) (only valid for port 0)	OUTPUT_CTL_STATE	[7:0]	0	R/O
1100H	MAC & I/O Control Module of Port 1	same as Port 0			
1200H	**MAC & I/O Control Module of Port 2**	same as Port 0			
1300H	**MAC & I/O Control Module of Port 3**	same as Port 0			
1400H	**MAC & I/O Control Module of Port 4**	same as Port 0			
1500H	**MAC & I/O Control Module of Port 5**	same as Port 0			
1600H	**MAC & I/O Control Module of Port 6**	same as Port 0			
1700H	**MAC & I/O Control Module of Port 7**	same as Port 0			
1800H	**MAC & I/O Control Module of Port 8**	same as Port 0			
1900H	**CPU I/O Control Module**				
00H	**CPU packet read byte count register bits [7:0]** CPU can check the incoming (ready to be received by CPU via IDE) packet length via the 11-bit register PKT_BYTE_CNT [10:0] before starting to read it.	PKT_BYTE_CNT	[7:0]	0	R/O
01H	**CPU packet read byte count register bits [10:8]**	PKT_BYTE_CNT	[10:8]	0	R/O
02H	**CPU packet read status register** 2' b00: idle or packet read (by CPU) in progress 2' b01: packet read (by CPU) complete without error 2' b10: packet read (by CPU) with error (CPU needs to read the same packet again)	RD_PKT_STATUS	[1:0]	0	R/O
03H	**Packet source port ID** CPU can check the incoming packet' s source port ID via the 3-bit register PKT_SRC_PORT before starting to read it. It is useful to the spanning tree algorithm.	PKT_SRC_PORT	[3:0]	0	R/O
04H	**CPU IO port configuration register** bit 0 : input port enable → 1: input enable, 0: input disable bit 1 : output port enable → 1: output enable, 0: output disable bit 2 : output port dequeue hold → 1: dequeue disabled, 0: dequeue enabled	~~CPUIO_CFG~~	[2:0]	0	R/W
05-06H	Granted (at most one) free buffer Addr in Input Control Bit [10:0]: buffer address (ID*3 + 40h) Bit [11]: buffer address valid	FREE_BUF_ADDR	[11:0]	0	R/O
07-08H	dequeued buffer Addr in Output Control bit [10:0]: buffer address (ID*3 + 40h) bit [11]: buffer address valid (default: invalid)	DEQUEUED_BUF_ADDR	[11:0]	0	R/O
10H	**CPU packet write status register**	WR_PKT_STAT	[2:0]	0	R/O

	bits [1:0] : packet write status 2' b00: idle or packet write in progress 2' b01: CPU sent packet successfully 2' b10: CPU sent packet unsuccessfully (CPU needs to re-write the packet again) Note that CPU packets have highest priority and will not be dropped by Forwarding Control. bit 2: CPU Input Control is ready for CPU to write packets (It can be ready only after setting CPUIO_CFG[0] = 1.) 0: not ready (default) 1: ready Note that if CPU IO is in XOFF window, it will be not ready with WR_PKT_STATUS[2]==1. CPU IO always uses flow control mechanism with displaying XON/XOFF status in XOFF_WINDOW[9] of Forwarding Control.	US			

Note: By using EEPROM initialization mode, all internal registers can be accessed.

SECTION III ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS

Parameter	Min	Max	Unit
Ambient operating temperature	0	70	^oC
Case temperature	0	100	^oC
Storage temperature	-55	125	^oC
Input voltage	-0.5	5.5	Volts
Output voltage (V_{CC} = 3.1 - 3.6V)	-0.5	V_{CC} + 0.5	Volts

Note: Stress above the conditions listed may cause permanent damage to the device. Functional operation of this device should be restricted to the conditions described under operating conditions.

DC CHARACTERISTICS

TA-0-70^{o}C, V_{CC}=3.3V+/-5%, GND=0V

Symbol	Parameter	Min	Max	Unit	Condition
V_{IL}	Input low voltage	-0.50	0.8	V	
V_{IH}	Input high voltage	2.0	V_{CC}+0.5	V	
V_{OL}	Output low voltage	-	0.45	V	I_{OL}=4.0mA
V_{OH}	Output high voltage	2.4	-	V	I_{OH}=-1.0mA
I_{IL}	Input leakage current	-	+/-10	uA	$0<V_{IN}<V_{CC}$
I_{OZ}	Tristate leakage current	-	+/-20	uA	$0.45<V_{OUT}<V_{CC}$
I_{CC}	Power supply current	-	**TBD**	mA	

AC CHARACTERISTICS

AC timing specifications provided are based on external zero-pf capacitance load. Min/Max cases are based on the following table:

Parameter	Min	Max	Unit
3.3V power (Vcc)	3.135	3.465	V
Temperature	0	95	^{o}C

- SRAM interface Timing Characteristics

	SETUP	HOLD	MIN	MAX	UNIT
SA output delay			2	7	
SD input	2.5	1.5			
SD output delay			2	7	
SADS			2	7	
SCS0			2	7	
SWE			2	7	

- RMII Interface Timing Characteristics

Parameter	min	type	max	unit	condition
RCLK50 cycle time			20	ns	
RXD CRS_DV setup time	4	-	-	ns	to RCLK50 rising edge
RXD CRS_DV hold time	2	-	-	ns	to RCLK50 rising edge
TXD TX_EN output delay	3	-	12	ns	to RCLK50 rising edge

- Management Interface (MI) Timing Characteristics

Parameter	min	typ	max	unit	condition
MDC cycle time	-	400	-	ns	
MDC high time	180	200	220	ns	
MDC low time	180	200	220	ns	
MDIO setup time (source by PHY)	30	-	-	ns	to MDC rising edge
MDIO hold time (source by PHY)	0	-	-	ns	to MDC rising edge
MDIO output delay (source by vt3061)	200	-	300	ns	to MDC rising edge

- EEPROM Interface Timing Characteristics

Parameter	min	typ	max	unit	condition

EEC clock frequency		0	78.12	-	kHz
Clock high time	-	6.4	-	µs	
Clock low time	-	6.4	-	µs	
Start Condition setup time		6.4	-	-	µs
Start Condition hold time	6.4	-	-	µs	
Stop Condition setup time		6.4	-	-	µs
Stop Condition hold time	6.4	-	-	µs	
Read Data In setup time	0	-	-		to EEC rising edge
Read Data In hold time	0	-	-		to EEC falling edge
EEIO Data out delay	2.6	-	3.0	µs	to EEC falling edge
Write Cycle time	-	11.4	-	ms	

- CPU interface IO Timing Characteristics

PARAMETER	MIN	MAX	UNIT
IOR/IOW falling to IOR/IOW rising	70		ns
IOR/IOW rising to IOR/IOW falling	25		ns
HD valid to IOR/IOW falling	25		ns
IOW data setup(write data valid to IOW rising)	20		ns
IOW data hold(IOW rising to write data invalid)	10		ns
IOR data setup(read data valid to IOR rising)	20		ns
IOR data hold(IOW rising to read data invalid)	5		ns
HCS falling to IOR/IOW falling (HCS setup time)	4		ns
HCS rising to IOR/IOW rising (HCS hold time)	4		ns

PACKAGE MECHANICAL SPECIFICATIONS