AX11025
Single Chip Microcontroller with TCP/IP and 10/100M Fast Ethernet MAC/PHY

Document No: AX11025/V1.09/06/14/2011

Features

- **MCU**
  - 8-bit pipelined RISC, single cycle per instruction with maximum operating frequency of 100Mhz (100 MIPS)
  - 100% software compatible with standard 8051/80390
  - 4 GPIO ports of 8 bits
  - 2 external interrupt sources with 2 priority levels
  - Support power management unit, programmable watchdog timer, and 3 16-bit timer/counters
  - Debug port for connecting to In-Circuit Emulation (ICE) adaptor
  - 5 channels of programmable counter array
- **On-chip Program and Data Memory**
  - Embed 512KB Flash memory, and 16KB SRAM for program code mirroring. The external program memory can grow up to 2MB without bank select
  - Support initial Flash memory programming via UART or ICE adaptor, the so-called In System Programming (ISP)
  - Support reprogrammable boot code and In Application Programming (IAP) to update run-time firmware or boot code through Ethernet or UART (US Patent Pending)
  - Support boot loader to shadow program code on to internal 16KB SRAM and external SRAM for high performance applications
  - Embed 32KB SRAM for data memory, expandable up to 2MB via external SRAM without bank select
- **Buffer Management**
  - Innovative-shared memory architecture to allow external program and data memory to share the same SRAM memory chip with flexible memory space allocation
  - Embed DMA engine and memory arbiter. Support 3 DMA channels for high performance data movement needed for network protocol stack processing
- **On-chip 10/100M Fast Ethernet MAC and PHY**
  - Integrate IEEE 802.3 10BASE-T/100BASE-TX compatible Fast Ethernet MAC and PHY with dedicated 12KB SRAM for Ethernet packet buffering. Support full-duplex and half-duplex operations. Provide optional MII interface (for HomePNA and HomePlug)
  - Support twisted pair crossover detection and auto-correction (HP Auto-MDIX)
  - Support wakeup via Link-up, Magic packet, Wakeup frame, external input pin or UART
- **TCP/IP**
  - Build in TCP/IP accelerator in hardware to improve network transfer throughput. Support IP/TCP/UDP/ICMP/IGMP checksum and ARP in hardware
  - Support TCP, UDP, ICMP, IPv4, DHCP, BOOTP, ARP, DNS, SMTP, SNTP, uPNP, PPPoE and HTTP in software
- **Communication Interface**
  - 3 UART interface (with 1 supporting 921.6Kbps and Modem control)
  - 1 I2C interface (master and slave mode)
  - SPI/Micro wire interface (3 masters or 1 slave mode)
  - 1 1-Wire controller interface (master mode)
  - 1 CAN 2.0 controller interfac**e**
  - 10/100 Ethernet PHY interface

- Support network boot over Ethernet using BOOTP and TFTP
- Integrate on-chip voltage regulator and require single power supply of 3.3V only
- Integrate on-chip oscillator and PLL. Require only one 25Mhz crystal to operate
- Integrate on-chip power-on reset circuit
- 128-pin LQFP RoHS package
- Operating temperature: -40 to 85˚C

*IEEE is a registered trademark of the Institute of Electrical and Electronic Engineers, Inc.

*All other trademarks and registered trademark are the property of their respective holders.

Product Description

The AX11025, Single Chip Micro-controller with TCP/IP and 10/100M Fast Ethernet MAC/PHY, is a System-on-Chip (SoC) solution which offers a high performance embedded micro-controller and rich communication peripherals for wide varieties of application which need access to the LAN or Internet. With built-in network protocol stack, the AX11025 provides very cost effective networking solution to enable simple, easy, and low cost Internet connection capability for many applications such as consumer electronics, networked home appliances, industrial equipments, security systems, remote data collection equipments, remote control, remote monitoring, and remote management.

Always contact ASIX Electronics for possible updates before starting a design.

This data sheet contains new products information. ASIX Electronics reserves the rights to modify product specification without notice. No liability is assumed as a result of the use of this product. No rights under any patent accompany the sale of the product.

Target Applications

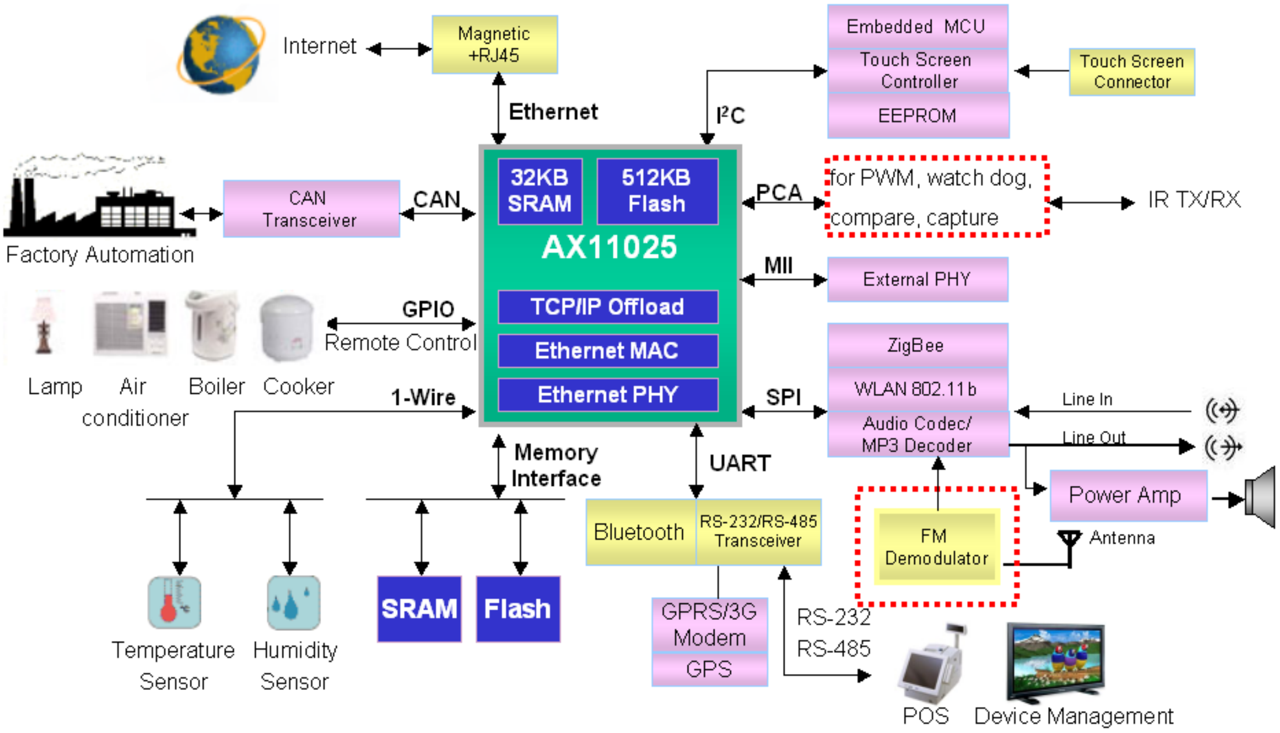


Figure 1: Target Application Diagram

Typical System Block Diagrams

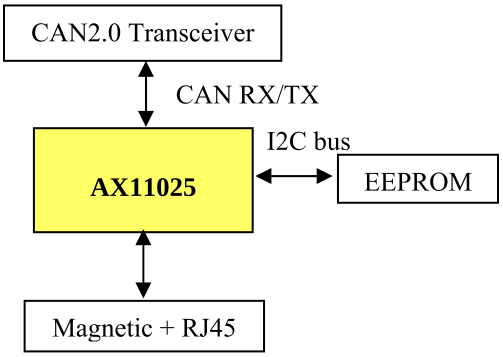


Figure 2: CAN to Ethernet Converter

DISCLAIMER

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of ASIX. ASIX may make changes to the product specifications and descriptions in this document at any time, without notice.

ASIX provides this document "as is" without warranty of any kind, either expressed or implied, including without limitation warranties of merchantability, fitness for a particular purpose, and non-infringement.

Designers must not rely on the absence or characteristics of any features or registers marked "reserved", "undefined" or "NC". ASIX reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Always contact ASIX to get the latest document before starting a design of ASIX products.

TRADEMARKS

ASIX, the ASIX logo are registered trademarks of ASIX Electronics Corporation. All other trademarks are the property of their respective owners.

Table of Contents

1.0 INTRODUCTION 11
1.1 GENERAL DESCRIPTION 11
1.2 AX11025 BLOCK DIAGRAM 11
1.3 AX11025 PINOUT DIAGRAM 12
1.4 SIGNAL DESCRIPTION 13
2.0 FUNCTION DESCRIPTION 21
2.1 CLOCK GENERATION 21
2.2 RESET GENERATION 21
2.3 VOLTAGE REGULATOR 21
2.4 CPU CORE AND DEBUGGER 22
2.4.1 CPU Core 22
2.4.2 Debugger 22
2.5 ON-CHIP FLASH MEMORY 24
2.6 MEMORY ARBITER AND BOOT LOADER 25
2.6.1 Boot Loader 27
2.6.2 Memory Arbiter 27
2.6.3 Flash Programming Controller 27
2.7 DMA ENGINE 28
2.8 INTERRUPT CONTROLLER 29
2.9 WATCHDOG TIMER 30
2.10 POWER MANAGEMENT UNIT 31
2.10.1 PMM 31
2.10.2 STOP Mode 31
2.11 TIMERS AND COUNTERS 32
2.12 UARTS 33
2.12.1 UART 0 and UART 1 33
2.12.2 UART 2 33
2.13 GPIOS 34
2.14 TCP/IP OFFLOAD ENGINE 35
2.15 10/100M ETHERNET MAC 36
2.16 10/100M ETHERNET PHY 37
2.17 PROGRAMMABLE COUNTER ARRAY 38
2.18 I2C CONTROLLER 38
2.19 1-WIRE CONTROLLER 39
2.20 SPI CONTROLLER 40
2.21 CAN CONTROLLER 41
3.0 MEMORY MAP DESCRIPTION 43
3.1 I2C CONFIGURATION EEPROM MEMORY MAP 43
3.1.1 Length (0x00) 44
3.1.2 Flag (0x01) 44
3.1.3 Multi-function Pin Setting (0x02~0x03) 45
3.1.4 Programmable Output Driving Strength (0x04) 47
3.1.5 Node ID (0x06~0x0B) 48
3.1.6 Maximum Packet Size (0x0C~0x0D) 48
3.1.7 Primary/Secondary PHY Type and PHY ID (0x0E~0x0F) 48
3.1.8 Pause Frame Low Water and High Water Mark (0x10~0x11) 48
3.1.9 TOE TX VLAN Tag (0x14~0x15) 49
3.1.10 TOE RX VLAN Tag (0x16~0x17) 49
3.1.11 TOE ARP Cache Timeout (0x18) 49
3.1.12 TOE Source IP Address (0x19~0x1C) 49
3.1.13 TOE Subnet Mask (0x1D~0x20) 49

3.1.14 TOE L4 DMA Transfer Gap (0x21) ... 49
3.2 Program Memory Map ... 50
3.3 External Data (xDATA) Memory Map ... 51
3.4 Internal Data Memory and SFR Register Map ... 51
4.0 DETAILED FUNCTION DESCRIPTION ... 53
4.1 Clock Generation ... 53
4.2 Reset Generation ... 54
4.3 Voltage Regulator ... 55
4.4 CPU Core and Debugger ... 56
4.4.1 CPU Core SFR Register Map ... 57
4.4.2 CPU Core SFR Register Description ... 58
4.4.3 Memory Allocation ... 59
4.4.4 Performance Improvement ... 66
4.4.5 Debugger ... 72
4.5 On-Chip Flash Memory ... 73
4.6 Memory Arbiter & Boot Loader ... 85
4.6.1 Boot Loader ... 85
4.6.2 Memory Arbiter ... 88
4.6.3 Program and Data Memory Address Mapping ... 89
4.6.4 Flash Memory Address Re-mapping for the Lower 16KB Boot Sector ... 91
4.6.5 Flash Memory Access in Program Code Shadow Mode ... 94
4.6.6 Flash Programming Controller ... 95
4.7 DMA Engine ... 96
4.7.1 DMA Transfers for Ethernet Packet Receive and Transmit ... 96
4.7.2 Software DMA ... 97
4.7.3 DMA Arbitration ... 102
4.8 Interrupt Controller ... 104
4.8.1 Interrupt Controller SFR Register Map ... 104
4.9 Watchdog Timer ... 110
4.9.1 Watchdog SFR Register Map ... 110
4.9.2 Watchdog Interrupt ... 111
4.9.3 Watchdog Timer Reset ... 112
4.9.4 Simple Timer ... 112
4.9.5 System Monitor ... 112
4.9.6 Clock Control ... 113
4.9.7 Timed Access Register ... 114
4.10 Power Management Unit ... 115
4.10.1 Power Management Unit SFR Register Map ... 115
4.10.2 Power Management Mode ... 116
4.10.3 STOP Mode ... 118
4.11 Timers and Counters ... 120
4.11.1 Timers 0, 1, 2 Related SFR Register Map ... 120
4.11.2 Timer 0, 1 ... 120
4.11.3 Timer 2 ... 127
4.11.4 Millisecond Timer ... 129
4.12 UARTs ... 130
4.12.1 UART 0, 1 SFR Register Map ... 130
4.12.2 UART 0 ... 130
4.12.3 UART 1 ... 134
4.12.4 UART 2 ... 138
4.13 GPIO ... 148
4.13.1 GPIO SFR Register Map ... 148
4.14 TCP/IP Offload Engine ... 151
4.14.1 TOE SFR Register Map ... 154
4.14.2 L2_Engine Function Description ... 155
4.14.3 L3_Engine Function Description ... 161
4.14.4 L4_Engine Function Description ... 163

4.14.5 Packet Buffer Ring in xDATA Memory of 1T 80390 CPU *166*
4.14.6 Packet Format in Packet Buffer Ring *169*
4.15 10/100M ETHERNET MAC 179
4.15.1 10/100M Ethernet MAC SFR Register Map *180*
4.15.2 Ethernet MAC Receive Filtering *181*
4.15.3 Ethernet MAC Packet Transmit *186*
4.15.4 Ethernet MAC Buffer Management *188*
4.15.5 Magic Packet and Wakeup Frame *190*
4.16 10/100M ETHERNET PHY 197
4.16.1 MII Station Management Function *198*
4.16.2 10/100M Ethernet PHY SFR Register Map *198*
4.17 PROGRAMMABLE COUNTER ARRAY 204
4.17.1 Programmable Counter Array SFR Register Map *204*
4.17.2 PCA Timer/Counter *205*
4.17.3 Compare/Capture Modules *206*
4.18 I2C CONTROLLER 215
4.18.1 I2C Controller SFR Register Map *216*
4.18.2 I2C Slave Mode Function Description *222*
4.19 1-WIRE CONTROLLER 224
4.19.1 1-Wire Controller SFR Register Map *224*
4.20 SPI CONTROLLER 230
4.20.1 SPI SFR Register Map *232*
4.20.2 SPI Slave Mode Function Description *238*
4.21 CAN CONTROLLER 242
4.21.1 CAN SFR Register Map *248*
4.21.2 Basic Mode Operation *250*
4.21.3 Extended Mode Operation *257*

5.0 ELECTRICAL SPECIFICATIONS **273**

5.1 DC CHARACTERISTICS 273
5.1.1 Absolute Maximum Ratings *273*
5.1.2 Recommended Operating Condition *273*
5.1.3 Leakage Current and Capacitance *274*
5.1.4 DC Characteristics of 3.3V I/O Pins *274*
5.1.5 DC Characteristics of 3.3V with 5V Tolerant I/O Pins *274*
5.1.6 DC Characteristics of Voltage Regulator *275*
5.2 POWER CONSUMPTION 276
5.3 POWER-UP SEQUENCE 277
5.4 AC TIMING CHARACTERISTICS 278
5.4.1 Clock Timing *278*
5.4.2 External Memory Interface Timing *279*
5.4.3 I2C Interface Timing *283*
5.4.4 SPI Interface Timing *284*
5.4.5 1-Wire Interface Timing *286*
5.4.6 Programmable Counter Array Interface Timing *289*
5.4.7 Timer 0/1/2 Interface Timing *290*
5.4.8 10/100M Ethernet PHY Interface Timing *291*
5.4.9 MII Timing *292*
5.4.10 Station Management Timing *293*

6.0 PACKAGE INFORMATION **294**

7.0 ORDERING INFORMATION **295**

8.0 REVISION HISTORY **295**

List of Figures

FIGURE 1: TARGET APPLICATION DIAGRAM 2
FIGURE 2: CAN TO ETHERNET CONVERTER 2
FIGURE 3: AX11025 BLOCK DIAGRAM 11
FIGURE 4: AX11025 PINOUT DIAGRAM 12
FIGURE 5: TYPICAL DEBUGGER AND HARDWARE ASSISTED DEBUGGER (HAD2) SYSTEM DIAGRAM 23
FIGURE 6: SYSTEMS REQUIRING ONLY 512K BYTES OF PROGRAM FLASH AND 32K BYTES OF DATA MEMORY 25
FIGURE 7: SYSTEMS REQUIRING 512K BYTES PROGRAM FLASH AND OVER 32K BYTES DATA MEMORY 25
FIGURE 8: SYSTEMS REQUIRING OVER 512K BYTES PROGRAM FLASH AND OVER 32K BYTES DATA MEMORY 26
FIGURE 9: SYSTEMS REQUIRING 512K BYTES PROGRAM FLASH AND OVER 32K BYTES DATA MEMORY FOR HIGH PERFORMANCE 26
FIGURE 10: FLASH MEMORY PROGRAMMING SYSTEM CONFIGURATION 28
FIGURE 11: WATCHDOG TIMER BLOCK DIAGRAM 30
FIGURE 12: TIMERS 0, 1, AND 2 BLOCK DIAGRAM 32
FIGURE 13: I/O BUFFER OF RXD0 PIN OF UART 0 AND RXD1 PIN OF RXD1 33
FIGURE 14: THE I/O BUFFER OF GPIO PINS 34
FIGURE 15: INTERNAL DATA PATH DIAGRAM OF 10/100M ETHERNET MAC, 10/100 ETHERNET PHY, AND MII INTERFACE 36
FIGURE 16: INTERNAL CONTROL MUX FOR STATION MANAGEMENT INTERFACE 37
FIGURE 17: PROGRAMMABLE COUNTER ARRAY BLOCK DIAGRAM 38
FIGURE 18: I2C CONTROLLER BLOCK DIAGRAM 39
FIGURE 19: 1-WIRE CONTROLLER BLOCK DIAGRAM 39
FIGURE 20: SPI CONTROLLER BLOCK DIAGRAM 40
FIGURE 21: CAN CONTROLLER BLOCK DIAGRAM 42
FIGURE 22: THE PROGRAM MEMORY MAP OF 1T 80390 CPU CORE (PROGRAM NON-SHADOW MODE) 50
FIGURE 23: THE PROGRAM MEMORY MAP OF 1T 80390 CPU CORE (PROGRAM SHADOW MODE) 50
FIGURE 24: THE EXTERNAL DATA MEMORY MAP OF 1T 80390 CPU CORE 51
FIGURE 25: THE INTERNAL MEMORY MAP OF 1T 80390 CPU CORE 51
FIGURE 26: AX11025 CLOCK GENERATION BLOCK DIAGRAM 53
FIGURE 27: AX11025 RESET GENERATION BLOCK DIAGRAM 54
FIGURE 28: AX11025 RESET TIMING DIAGRAM 54
FIGURE 29: CPU CORE BLOCK DIAGRAM 56
FIGURE 30: STACK BYTES ORDER 65
FIGURE 31: ON-CHIP FLASH MEMORY BLOCK DIAGRAM 73
FIGURE 32: AUTOMATIC CHIP ERASE ALGORITHM FLOWCHART 76
FIGURE 33: AUTOMATIC SECTOR ERASE ALGORITHM FLOWCHART 77
FIGURE 34: ERASE SUSPEND/ERASE RESUME FLOWCHART 78
FIGURE 35: AUTOMATIC PROGRAMMING ALGORITHM FLOWCHART 79
FIGURE 36: DATA# POLLING ALGORITHM 81
FIGURE 37: TOGGLE BIT ALGORITHM 83
FIGURE 38: BOOT LOADER, MEMORY ARBITER & FLASH PROGRAMMING CONTROLLER BLOCK DIAGRAM 85
FIGURE 39: CPU PROGRAM MEMORY MAP (EXT_PROG_SRAM_EN = 0) 89
FIGURE 40: CPU PROGRAM MEMORY MAP (EXT_PROG_SRAM_EN = 1) 89
FIGURE 41: CPU PROGRAM MEMORY AND XDATA MEMORY MAP (EXT_PROG_SRAM_EN = 1) 90
FIGURE 42: CPU PROGRAM MEMORY AND XDATA MEMORY MAP (EXT_PROG_SRAM_EN = 1) 90
FIGURE 43: CPU PROGRAM MEMORY AND XDATA MEMORY MAP (EXT_PROG_SRAM_EN = 0) 91
FIGURE 44: FLASH MEMORY ADDRESS WITHOUT RE-MAPPING, SFR REGISTER CSREPR [FARM] = 0 (DEFAULT) 92
FIGURE 45: FLASH MEMORY ADDRESS RE-MAPPING ENABLED, SFR REGISTER CSREPR [FARM] = 1 92
FIGURE 46: DMA ENGINE BLOCK DIAGRAM 96
FIGURE 47: RING-AWARE SOFTWARE DMA EXAMPLE 98
FIGURE 48: EXAMPLE: ETHERNET PACKET RECEIVE DMA TRANSFER ONLY (RECEIVING A 1500-BYTE PACKET) 103
FIGURE 49: EXAMPLE: ETHERNET PACKET RECEIVE AND TRANSMIT DMA TRANSFERS SIMULTANEOUSLY (RECEIVING AND TRANSMITTING A 1500-BYTE PACKET) 103
FIGURE 50: EXAMPLE: ETHERNET PACKET RECEIVE AND TRANSMIT AND SOFTWARE DMA TRANSFERS SIMULTANEOUSLY (RECEIVING AND TRANSMITTING A 1500-BYTE PACKET, AND SOFTWARE COPYING A 200-BYTES DATA BLOCK) 103

FIGURE 51: WATCHDOG TIMER BLOCK DIAGRAM..110
FIGURE 52: AX11025 OPERATING MODE TRANSITION DIAGRAM..115
FIGURE 53: TIMER/COUNTER 0, MODE 0: 13-BIT TIMER/COUNTER..123
FIGURE 54: TIMER/COUNTER 0, MODE 1: 16-BIT TIMER/COUNTER..123
FIGURE 55: TIMER/COUNTER 0, MODE 2: 8-BIT TIMER/COUNTER WITH AUTO-RELOAD..124
FIGURE 56: TIMER/COUNTER 0, MODE 3: TWO 8-BIT TIMERS/COUNTERS..124
FIGURE 57: TIMER/COUNTER 1, MODE 0: 13-BIT TIMERS/COUNTERS..125
FIGURE 58: TIMER/COUNTER 1, MODE 1: 16-BIT TIMERS/COUNTERS..125
FIGURE 59: TIMER/COUNTER 1, MODE 2: 8-BIT TIMER/COUNTER WITH AUTO-RELOAD..126
FIGURE 60: TIMER 2 BLOCK DIAGRAM IN TIMER MODE..128
FIGURE 61: TIMER 2 BLOCK DIAGRAM AS UART0 BAUD RATE GENERATOR..128
FIGURE 62: UART 0 BLOCK DIAGRAM..131
FIGURE 63: UART 0, MODE 0 TRANSMIT TIMING DIAGRAM..132
FIGURE 64: UART 0, MODE 1 TRANSMIT TIMING DIAGRAM..132
FIGURE 65: UART 0, MODE 2 TRANSMIT TIMING DIAGRAM..133
FIGURE 66: UART 0, MODE 3 TRANSMIT TIMING DIAGRAM..133
FIGURE 67: UART 1 BLOCK DIAGRAM..134
FIGURE 68: UART 1, MODE 0 TRANSMIT TIMING DIAGRAM..136
FIGURE 69: UART 1, MODE 1 TRANSMIT TIMING DIAGRAM..136
FIGURE 70: UART1, MODE 2 TRANSMIT TIMING DIAGRAM..137
FIGURE 71: UART 1, MODE 3 TRANSMIT TIMING DIAGRAM..137
FIGURE 72: UART 2 BLOCK DIAGRAM..138
FIGURE 73: PORTS PIN LOGIC..149
FIGURE 74: DATA REGISTER ACCESSED BY READ-MODIFY-WRITE INSTRUCTIONS..149
FIGURE 75: PORTS WRITE TIMING DIAGRAM..149
FIGURE 76: PORTS READ TIMING DIAGRAM..149
FIGURE 77: TOE BLOCK DIAGRAM..151
FIGURE 78: ARP CACHE SRAM MEMORY MAP..156
FIGURE 79: THE EXTERNAL DATA (XDATA) MEMORY OF CPU..166
FIGURE 80: THE CONTENT OF BUFFER DESCRIPTOR PAGE (BDP)..167
FIGURE 81: EXAMPLE RING STRUCTURE OF RECEIVE PACKET BUFFER RING..168
FIGURE 82: EXAMPLE RING STRUCTURE OF TRANSMIT PACKET BUFFER RING..168
FIGURE 83: ICMP PACKET FORMAT IN RPBR..169
FIGURE 84: IGMP PACKET FORMAT IN RPBR..170
FIGURE 85: UDP PACKET FORMAT IN RPBR..171
FIGURE 86: TCP PACKET FORMAT IN RPBR..173
FIGURE 87: NON-IP-TYPE PACKET FORMAT IN RPBR..173
FIGURE 88: ICMP PACKET FORMAT IN TPBR..175
FIGURE 89: IGMP PACKET FORMAT IN TPBR..175
FIGURE 90: UDP PACKET FORMAT IN TPBR..176
FIGURE 91: TCP PACKET FORMAT IN TPBR..178
FIGURE 92: NON-IP-TYPE PACKET FORMAT IN TPBR..178
FIGURE 93: 10/100M ETHERNET MAC BLOCK DIAGRAM..179
FIGURE 94: MULTICAST FILTER ARRAY HASHING ALGORITHM..183
FIGURE 95: MULTICAST FILTER ARRAY BIT MAPPING..184
FIGURE 96: ETHERNET PACKET FORMAT..186
FIGURE 97: ETHERNET PACKET BUFFER DATA STRUCTURE IN ETHERNET MAC..188
FIGURE 98: TRANSMIT/RECEIVE BUFFER RING STRUCTURE..189
FIGURE 99: PAUSE FRAME..189
FIGURE 100: EXAMPLE MAGIC PACKET FORMAT..190
FIGURE 101: 10/100M ETHERNET PHY BLOCK DIAGRAM..197
FIGURE 102: MII STATION MANAGEMENT FRAME FORMAT..198
FIGURE 103: PROGRAMMABLE COUNTER ARRAY BLOCK DIAGRAM..204
FIGURE 104: PCA TIMER/COUNTER..205
FIGURE 105: PCA CAPTURE MODE..206
FIGURE 106: PCA SOFTWARE TIMER MODE (COMPARE MODE)..207
FIGURE 107: PCA HIGH-SPEED OUTPUT MODE..208
FIGURE 108: PCA PULSE WIDTH MODULATOR MODE..209
FIGURE 109: I2C CONTROLLER BLOCK DIAGRAM..215

FIGURE 110: TRANSMITTING DATA TO AN I2C SLAVE DEVICE..221
FIGURE 111: I2C READ DATA 221
FIGURE 112: 1-WIRE CONTROLLER BLOCK DIAGRAM 224
FIGURE 113: SPI CONTROLLER BLOCK DIAGRAM 230
FIGURE 114: SPI TIMING DIAGRAM..231
FIGURE 115: COMMAND FRAME FORMAT IN SPI SLAVE MODE..240
FIGURE 116: CAN BLOCK DIAGRAM..243
FIGURE 117: CAN STANDARD DATA FRAME (11-BITS IDENTIFIER FIELD) FORMAT 243
FIGURE 118: CAN EXTENDED DATA FRAME (29-BITS IDENTIFIER FIELD) FORMAT 244
FIGURE 119: CAN REMOTE FRAME (11-BITS IDENTIFIER FIELD SHOWN, 29-BITS IDENTIFIER ALSO SUPPORTED) FORMAT
....245
FIGURE 120: CAN ERROR FRAME FORMAT...246
FIGURE 121: CAN OVERLOAD FRAME FORMAT...247
FIGURE 122: EXAMPLE BIT TIME STRUCTURE OF CAN SERIAL DATA 254
FIGURE 123: ARBITRATION LOST CAPTURE REGISTER CONTENT INTERPRETATION 262
FIGURE 124: ARBITRATION LOST EXAMPLE (RESULTANT ALC = 0X08)...262
FIGURE 125: SINGLE FILTER MODE FOR RECEIVING STANDARD FRAME FORMAT 266
FIGURE 126: SINGLE FILTER MODE FOR RECEIVING EXTENDED FRAME FORMAT..267
FIGURE 127: DUAL FILTER MODE FOR RECEIVING STANDARD FRAME FORMAT...268
FIGURE 128: DUAL FILTER MODE FOR RECEIVING EXTENDED FRAME FORMAT 269
FIGURE 129: POWER-UP SEQUENCE TIMING DIAGRAM AND TABLE 277
FIGURE 130: XTL25P CLOCK TIMING DIAGRAM AND TABLE 278
FIGURE 131: LB_CLK CLOCK TIMING DIAGRAM AND TABLE 278
FIGURE 132: CPU PROGRAM READ AND PROGRAM WRITE TIMING DIAGRAM AND TABLE 279
FIGURE 133: CPU DATA READ AND DATA WRITE TIMING DIAGRAM AND TABLE 280
FIGURE 134: DMA DATA READ AND DATA WRITE TIMING DIAGRAM AND TABLE 281
FIGURE 135: BOOT LOADER READING FLASH AND WRITING SRAM TIMING DIAGRAM AND TABLE 282
FIGURE 136: SPI MASTER CONTROLLER TIMING DIAGRAM AND TABLE 284
FIGURE 137: SPI SLAVE CONTROLLER TIMING DIAGRAM AND TABLE..285
FIGURE 138: 1-WIRE RESET PULSE AND PRESENCE PULSE TIMING DIAGRAM AND TABLE 286
FIGURE 139: 1-WIRE WRITE AND READ TIME SLOT TIMING DIAGRAM AND TABLE..287
FIGURE 140: 1-WIRE STPZ RESET AND READ WRITE TIMING DIAGRAM AND TABLE..288
FIGURE 141: ECI TIMING DIAGRAM AND TABLE...289
FIGURE 142: CEX[4:0] TIMING DIAGRAM AND TABLE 289
FIGURE 143: TM_CK[2:0] TIMING DIAGRAM AND TABLE..290
FIGURE 144: TM_GT[2:0] TIMING DIAGRAM AND TABLE 290
FIGURE 145: 10/100M ETHERNET PHY TRANSMITTER WAVEFORM AND SPEC...291

List of Tables

TABLE 1: PINOUT DESCRIPTION...13
TABLE 2: INTERRUPT CONTROLLER SUMMARY 29
TABLE 3: I2C CONFIGURATION EEPROM MEMORY MAP 43
TABLE 4: THE SFR REGISTER MAP 52
TABLE 5: CPU CORE SFR REGISTER MAP...57
TABLE 6: ON-CHIP FLASH MEMORY SECTOR STRUCTURE 73
TABLE 7: ON-CHIP FLASH MEMORY COMMAND DEFINITIONS 75
TABLE 8: WRITE OPERATION STATUS 80
TABLE 9: ON-CHIP FLASH MEMORY READ PROTECTION 84
TABLE 10: BLOCK NUMBER SETTING FOR PROGRAM SHADOW MODE 87
TABLE 11: SOFTWARE DMA AND MILLISECOND TIMER RELATED SFR REGISTER MAP 98
TABLE 12: SOFTWARE DMA AND MILLISECOND TIMER REGISTER MAP..99
TABLE 13: INTERRUPTS FLAG SUMMARY 104
TABLE 14: INTERRUPT CONTROLLER SFR REGISTER MAP 104
TABLE 15: WATCHDOG TIMER SFR REGISTER MAP...110
TABLE 16: WATCHDOG BITS AND ACTIONS..112
TABLE 17: TIMED ACCESS REGISTERS..114

TABLE 18: POWER MANAGEMENT UNIT SFR REGISTER MAP 115
TABLE 19: TIMERS 0, 1, 2 PIN DESCRIPTION 120
TABLE 20: TIMERS 0, 1, 2 RELATED SFR REGISTER MAP 120
TABLE 21: TIMER 2 MODE OF OPERATION 127
TABLE 22: MILLISECOND TIMER DIVIDER RATIO 129
TABLE 23: UART 0, 1 SFR REGISTER MAP 130
TABLE 24: UART 2 SFR REGISTER MAP 141
TABLE 25: UART2 INTERRUPT IDENTIFICATION REGISTER 144
TABLE 26: GENERAL PURPOSE I/O PORTS PINS DESCRIPTION 148
TABLE 27: GPIO SFR REGISTER MAP 148
TABLE 28: READ-MODIFY-WRITE INSTRUCTIONS 148
TABLE 29: TOE OPERATION MODES 153
TABLE 30: TOE SFR REGISTER MAP 154
TABLE 31: TOE REGISTER MAP 155
TABLE 32: DA FIELD GENERATION RULE IN TRANSMIT DIRECTION 157
TABLE 33: L2_ENGINE RX TRUTH TABLE 158
TABLE 34: L2_ENGINE TX TRUTH TABLE 159
TABLE 35: 10/100M ETHERNET MAC SFR REGISTER MAP 180
TABLE 36: 10/100M ETHERNET MAC REGISTER MAP 181
TABLE 37: PACKET FILTERING DURING REMOTE-WAKEUP ENABLE MODE 185
TABLE 38: 10/100 ETHERNET PHY SFR REGISTER MAP 198
TABLE 39: EMBEDDED 10/100M ETHERNET PHY REGISTER MAP 200
TABLE 40: PROGRAMMABLE COUNTER ARRAY SFR REGISTER MAP 204
TABLE 41: PCA TIMER/COUNTER INPUT SOURCES AND REFERENCE TIMING TICK 205
TABLE 42: PULSE WIDTH MODULATOR FREQUENCY 209
TABLE 43: PCA MODULE MODES WITHOUT INTERRUPT ENABLED 212
TABLE 44: PCA MODULE MODES WITH INTERRUPT ENABLED 213
TABLE 45: I2C CONTROLLER SFR REGISTER MAP 216
TABLE 46: I2C CONTROLLER REGISTER MAP 217
TABLE 47: REFERENCE COMMAND INSTRUCTIONS IN I2C SLAVE MODE 222
TABLE 48: 1-WIRE CONTROLLER SFR REGISTER MAP 224
TABLE 49: 1-WIRE CONTROLLER REGISTER MAP 225
TABLE 50: SPI CONTROLLER SFR REGISTER MAP 232
TABLE 51: SPI CONTROLLER REGISTER MAP 233
TABLE 52: COMMAND INSTRUCTION IN SPI SLAVE MODE 239
TABLE 53: CAN CONTROLLER SFR REGISTER MAP 248
TABLE 54: CAN CONTROLLER REGISTER MAP IN BASIC MODE 249
TABLE 55: CAN CONTROLLER REGISTER MAP IN EXTENDED MODE 249
TABLE 56: I2C MASTER CONTROLLER TIMING TABLE 283
TABLE 57: I2C SLAVE CONTROLLER TIMING TABLE 283
TABLE 58: 10/100M ETHERNET PHY RECEIVER SPEC 291

1.0 Introduction

1.1 General Description

The AX11025, Single Chip Micro-controller with TCP/IP and 10/100M Fast Ethernet MAC/PHY, is a System-on-Chip (SoC) solution which offers a high performance embedded micro-controller and rich communication peripherals for wide varieties of application which need access to the LAN or Internet. With built-in network protocol stack, the AX11025 provides very cost effective networking solution to enable simple, easy, and low cost Internet connection capability for many applications such as consumer electronics, networked home appliances, industrial equipments, security systems, remote data collection equipments, remote control, remote monitoring, and remote management.

The AX11025 needs only a 25Mhz crystal to operate and its internal operating frequency is programmable from 25Mhz, 50Mhz, and 100Mhz, depending on system performance and power consumption trade-off. The AX11025 integrates an internal voltage regulator that requires only single power supply of 3.3V to operate, and an internal power-on-reset circuitry that simplifies the external reset circuitry on PCB. The package is 128-pin low-profile LQFP RoHS package and the operating temperature range is -40 to 85°C. Please refer to ordering information for part number details.

1.2 AX11025 Block Diagram

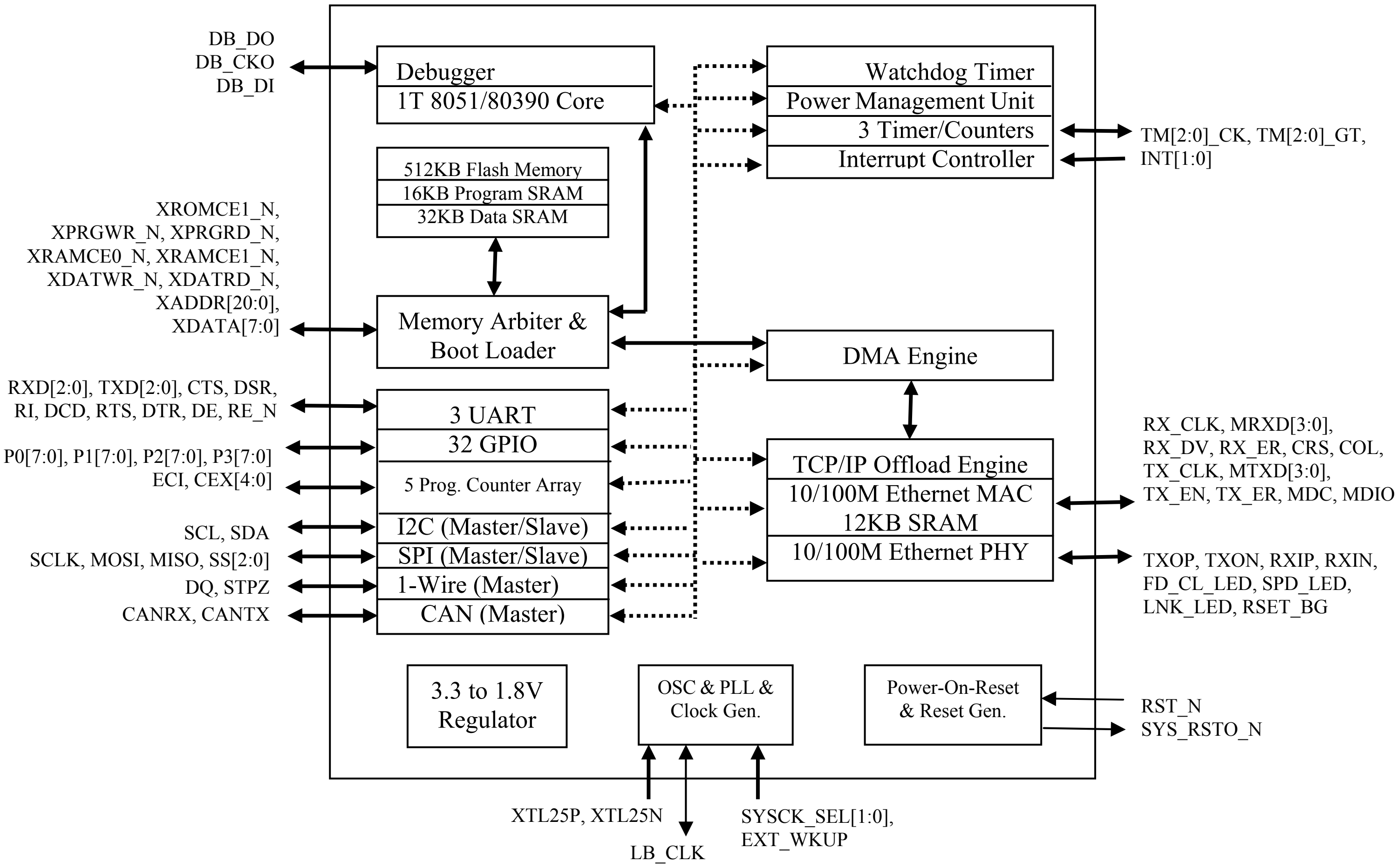


Figure 3: AX11025 Block Diagram

1.3 AX11025 Pinout Diagram

The AX11025 is housed in a 128-pin LQFP package.

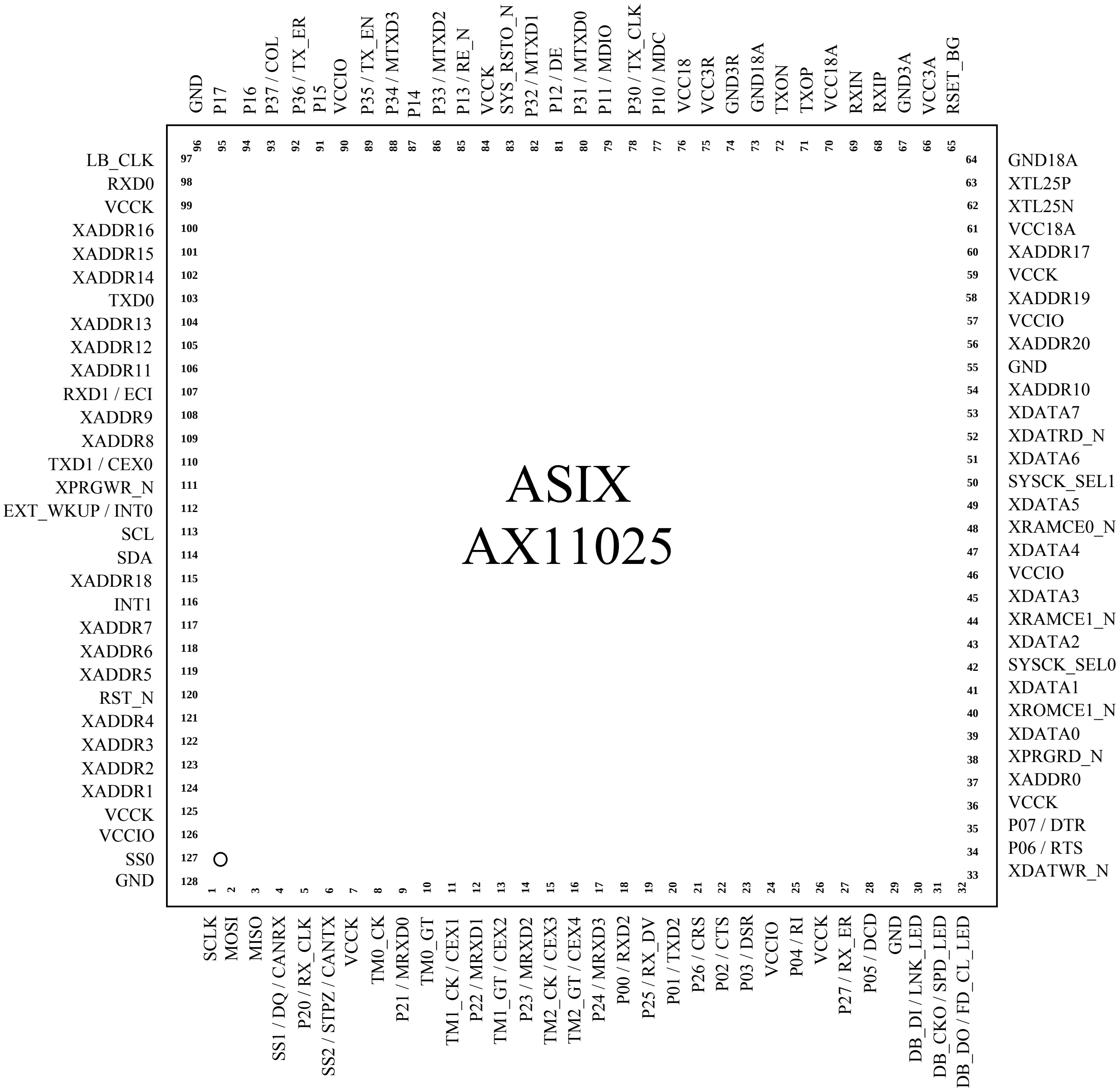


Figure 4: AX11025 Pinout Diagram

1.4 Signal Description

The following abbreviations apply to the following pin description table. Please note some I/O pins with multiple signal definitions on the same pin may have different pin attribute in the "Type" column for different signal definition. For example, pin 18 can be defined as P00, LA8 or RXD2. In the case of P00 the Type = B5/T/4m/8m/PU, while in the case of LA8 the Type = O5/4m/8m, and in the case of RXD2 the Type = I5. In other words, the PU (internal pull-up) only takes effective during P00 signal mode, and LA8 or RXD2 signal modes will not have the PU.

I18	Input, 1.8V	**4m**	4mA driving strength
I3	Input, 3.3V	**8m**	8mA driving strength
I5	Input, 3.3V with 5V tolerant	**PU**	Internal Pull-Up (75K)
O18	Output, 1.8V	**PD**	Internal Pull-Down (75K)
O3	Output, 3.3V	**P**	Power and ground pin
O5	Output, 3.3V with 5V tolerant	**S**	Schmitt Trigger
B3	Bi-directional I/O, 3.3V	**T**	Tri-state
B5	Bi-directional I/O, 3.3V with 5V tolerant	**AB**	Analog Bi-directional I/O
		AO	Analog Output

Table 1: Pinout Description

Pin Name	Type	Pin No	Pin Description
External Memory Interface			
XROMCE1_N	O3/4m	40	External program Flash memory chip enable, active low.
XRAMCE0_N	O3/4m	48	External program/data SRAM memory chip enable 0, active low. Note that the addressable space of external SRAM chip using XRAMCE0_N is programmable by the ASCS bits in I2C EEPROM offset 0x01. In other words, this allows users to choose from 64K bytes, 128K bytes, 256K bytes, 512K bytes, or 1024K bytes of SRAM chip whichever is most cost effective. See section 3.1.2 for details.
XRAMCE1_N	O3/4m	44	External program/data SRAM memory chip enable 1, active low.
XPRGWR_N	O3/4m/8m	111	External program Flash memory write enable, active low. Note that the output driving strength is programmable, by MI_ODS bit in I2C EEPROM offset 0x04. See section 3.1.4 for details.
XPRGRD_N	O3/4m/8m	38	External program Flash memory read enable, active low. Note that the output driving strength is programmable, by MI_ODS bit in I2C EEPROM offset 0x04. See section 3.1.4 for details.
XDATWR_N	O3/4m/8m	33	External program/data SRAM memory write enable, active low. Note that the output driving strength is programmable, by MI_ODS bit in I2C EEPROM offset 0x04. See section 3.1.4 for details.
XDATRD_N	O3/4m/8m	52	External program/data SRAM memory read enable, active low. Note that the output driving strength is programmable, by MI_ODS bit in I2C EEPROM offset 0x04. See section 3.1.4 for details.
XADDR [20:0]	O3/4m/8m	56, 58, 115, 60, 100, 101, 102, 104, 105, 106, 54, 108, 109, 117, 118, 119, 121, 122, 123, 124, 37	External program/data memory address bus. The output driving strength is programmable, by MI_ODS bit in I2C EEPROM offset 0x04. See section 3.1.4 for details.

<table>
<tr><td>XDATA [7:0]</td><td>B3/4m/8m</td><td>53, 51,
49, 47,
45, 43,
41, 39</td><td>External program/data memory’s data bus.
The XDATA [6:0] are also being used as chip configuration pins whose value will be latched during chip hardware reset, i.e., RST_N being low (but not software reset or software reboot). Their definition are as follows,
<table>
<tr><th>Pin Name</th><th>Pin Definition during chip power-on reset</th><th>Description</th></tr>
<tr><td rowspan="2">XDATA [0]</td><td rowspan="2">EXT_PROG_SRAM_EN</td><td>For high performance application, pull up with 10Kohm to enable program shadow mode, which will automatically copy program code from Flash memory to external SRAM before CPU starts running.</td></tr>
<tr><td>Pull down with 10Kohm to disable program shadow mode when the external SRAM is only used as data memory or when no external SRAM is used.</td></tr>
<tr><td>XDATA [1]</td><td>LB_MOD</td><td>This determines how LB_CLK is used. Please refer to LB_CLK signal description. For normal operation, please pull up with 10Kohm during chip hardware reset.</td></tr>
<tr><td>XDATA [2]</td><td>SYNC_BUS</td><td>This determines how LB_CLK is used. Please refer to LB_CLK signal description. For normal operation, please pull down with 10Kohm during chip hardware reset.</td></tr>
<tr><td>XDATA [3]</td><td>TEST_SPEEDUP</td><td>Please pull down with 10Kohm for normal operation.</td></tr>
<tr><td rowspan="2">XDATA [4]</td><td rowspan="2">BURN_FLASH_EN</td><td>Pull up with 10Kohm to temporarily enable Flash programming via UART0. This will put the CPU in reset state during Flash programming.</td></tr>
<tr><td>Pull down with 10Kohm to allow the CPU to run normally after reset and disable Flash programming via UART0.</td></tr>
<tr><td rowspan="2">XDATA [5]</td><td rowspan="2">BURN_FLASH_921K</td><td>Pull up with 10Kohm when the BURN_FLASH_EN is also pulled up to enable Flash memory programming at higher speed as 921.6Kbps baud rate. When the BURN_FLASH_EN is pulled down, this has no effect.</td></tr>
<tr><td>Pull down with 10Kohm when the BURN_FLASH_EN is also pulled up to enable Flash memory programming at normal speed as 115.2Kbps baud rate. When the BURN_FLASH_EN is pulled down, this has no effect.</td></tr>
<tr><td rowspan="2">XDATA [6]</td><td rowspan="2">I2C_BOOT_DIS</td><td>Pull up with 10Kohm if the optional I2C EEPROM is not used for storing configuration data.</td></tr>
<tr><td>Pull down with 10Kohm if the I2C EEPROM is used for storing configuration data.</td></tr>
</table>
Note that the output driving strength is programmable, by MI_ODS bit in I2C EEPROM offset 0x04. See section 3.1.4 for details.</td></tr>
<tr><th colspan="4">CPU Debugger/Interrupt/Timers/GPIO Interface</th></tr>
<tr><td>DB_DI</td><td>I5/PU</td><td>30</td><td>CPU debugger data input.
Note that this is a multi-function pin (DB_DI/LNK_LED), depending on the setting of DBG_PSEL bit in I2C EEPROM offset 0x01, see section 3.1.2 for details.</td></tr>
</table>

DB_CKO	O5/8m	31	CPU debugger clock output. Note that this is a multi-function pin (DB_CKO/SPD_LED), depending on the setting of DBG_PSEL bit in I2C EEPROM offset 0x01, see section 3.1.2 for details.
DB_DO	O5/8m	32	CPU debugger data output. Note that this is a multi-function pin (DB_DO/FD_CL_LED), depending on the setting of DBG_PSEL bit in I2C EEPROM offset 0x01, see section 3.1.2 for details.
INT [1:0]	I5/PU	116, 112	Interrupt inputs, active low or falling edge trigger. Note that the INT0 is a dual-function input pin sharing with EXT_WKUP pin.
TM[2:0]_CK	I5	15, 11, 8	Timer 2, 1, 0 external clock input (only TM0_CK has internal pull-up). Note that these are multi-function pins (TM2_CK/CEX3, TM1_CK/CEX1), depending on the setting of TM_PSEL bits in I2C EEPROM offset 0x03, see section 3.1.3 for details.
TM[2:0]_GT	I5	16, 13, 10	Timer 2, 1, 0 external gate control input (only TM_GT has internal pull-up). Note that these are multi-function pins (TM2_GT/CEX4, TM1_GT/CEX2), depending on the setting of TM_PSEL bits in I2C EEPROM offset 0x03, see section 3.1.3 for details.
P0 [7:0]	B5/T/4m/8m/PU	35, 34, 28, 25, 23, 22, 20, 18	Port 0 general purpose input and output pins. Note that these are multi-function pins (P07/DTR, P06/RTS, P05/DCD, P04/RI, P03/DSR, P02/CTS, P01/TXD2, P00/RXD2), depending on the setting of P0_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by P0_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
P1 [7:0]	B5/T/4m/8m/PU	95, 94, 91, 87, 85, 81, 79, 77	Port 1 general-purpose input and output pins. Note that these are multi-function pins (P13/RE_N, P12/DE, P11/MDIO, P10/MDC), depending on the setting of P1_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by P1_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
P2 [7:0]	B5/T/4m/8m/PU	27, 21, 19, 17, 14, 12, 9, 5	Port 2 general-purpose input and output pins. Note that these are multi-function pins (P27/RX_ER, P26/CRS, P25/RX_DV, P24/MRXD3, P23/MRXD2, P22/MRXD1, P21/MRXD0, P20/RX_CLK), depending on the setting of P2_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by P2_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
P3 [7:0]	B5/T/4m/8m/PU	93, 92, 89, 88, 86, 82, 80, 78	Port 3 general-purpose input and output pins. Note that these are multi-function pins (P37/COL, P36/TX_ER, P35/TX_EN, P34/MTXD3, P33/MTXD2, P32/MTXD1, P31/MTXD0, P30/TX_CLK), depending on the setting of P3_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by P3_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
UART Interface			
RXD0	B5/4m/PU	98	UART 0 serial receive data.
TXD0	O5/4m	103	UART 0 serial transmit data.
RXD1	B5/4m/PU	107	UART 1 serial receive data. Note that this is a multi-function pin (RXD1/ECI), depending on the setting of UIT_PSEL bits in I2C EEPROM offset 0x03, see section 3.1.3 for details.
TXD1	O5/4m/8m	110	UART 1 serial transmit data. Note that this is a multi-function pin (TXD1/CEX0), depending on the setting of UIT_PSEL bits in I2C EEPROM offset 0x03, see section 3.1.3 for details. The output driving strength is programmable, by PCA_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
RXD2	I5	18	UART 2 serial receive data. Note that this is a multi-function pin (P00/RXD2), depending on the setting of P0_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details.

TXD2	O5/4m/8m	20	UART 2 serial transmit data. Note that this is a multi-function pin (P01/TXD2), depending on the setting of P0_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by P0_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
CTS	I5	22	UART 2 clear to send. Note that this is a multi-function pin (P02/CTS), depending on the setting of P0_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details.
DSR	I5	23	UART 2 data set ready. Note that this is a multi-function pin (P03/DSR), depending on the setting of P0_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details.
RI	I5	25	UART 2 ring indicator. Note that this is a multi-function pin (P04/RI), depending on the setting of P0_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details.
DCD	I5	28	UART 2 data carriers detect. Note that this is a multi-function pin (P05/DCD), depending on the setting of P0_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details.
RTS	O5/4m/8m	34	UART 2 request to send. Note that this is a multi-function pin (P06/RTS), depending on the setting of P0_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by P0_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
DTR	O5/4m/8m	35	UART 2 data terminal ready. Note that this is a multi-function pin (P07/DTR), depending on the setting of P0_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by P0_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
DE	O5/4m/8m	81	UART 2 transceiver driver output enable. Note that this is a multi-function pin (P12/DE), depending on the setting of P1_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by P1_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
RE_N	O5/4m/8m	85	UART 2 transceiver receiver output enable, active low. Note that this is a multi-function pin (P13/RE_N), depending on the setting of P1_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by P1_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
Serial Interface			
SCL	B5/4m/8m/PU	113	I2C serial clock line for operating in either master or slave mode. Note that the output driving strength is programmable, by I2C_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
SDA	B5/4m/8m/PU	114	I2C serial data line for operating in either master or slave mode. Note that the output driving strength is programmable, by I2C_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
SS0	B5/T/4m	127	SPI slave select 0. This is a tri-stateable output when operating in SPI master mode or an input when operating in SPI slave mode. When operating in SPI master mode, it needs an external pulled-up resistor.
SS[2:1]	O5/T/4m/8m	6, 4	SPI slave select 2, 1. These are tri-stateable output (an external pulled-up resistor needed) and used in SPI master mode only. Note that these are multi-function pins (SS2/STPZ/CANTX, SS1/DQ/CANRX), depending on the setting of SPI_PSEL bits in I2C EEPROM offset 0x03, see section 3.1.3 for details. The output driving strength is programmable, by SPI_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.

SCLK	B5/T/4m/8m	1	SPI clock. This is a tri-stateable output when operating in SPI master mode or an input when operating in SPI slave mode. In SPI master mode operating at mode 0 or 2, user should pull low this pin with external resistor, while operating at mode 1 or 3, user should pull up this pin with external resistor. Note that the output driving strength is programmable, by SPI_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
MISO	B5/T/4m/8m	3	SPI master input slave output line. This is used to receive serial data when the SPI controller is configured as SPI master or to transmit serial data when it is configured as SPI slave. When operating in SPI slave mode, this is a tri-stateable output, which needs an external pulled-up resistor. Note that the output driving strength is programmable, by SPI_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
MOSI	B5/T/4m/8m	2	SPI master output slave input line. This is used to transmit serial data when the SPI controller is configured as SPI master or to receive serial data when it is configured as SPI slave. When operating in SPI master mode, this is a tri-stateable output, which needs an external pulled-up resistor. Note that the output driving strength is programmable, by the SPI_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
DQ	B5/4m/8m	4	1-Wire serial data input and output. This is an open-drain pin, which needs an external pulled-up resistor or a strong pull-up through a PMOS transistor. Note that this is a multi-function pin (SS1/DQ/CANRX), depending on the setting of SPI_PSEL bits in I2C EEPROM offset 0x03, see section 3.1.3 for details. The output driving strength is programmable, by SPI_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
STPZ	O5/4m/8m	6	1-Wire strong pull-up is used for device with a stiff power supply for high current application. This is active low. Note that this is a multi-function pin (SS2/STPZ/CANTX), depending on the setting of SPI_PSEL bits in I2C EEPROM offset 0x03, see section 3.1.3 for details. The output driving strength is programmable, by SPI_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
CANRX	I5	4	CAN serial receive data. Note that this is a multi-function pin (SS1/DQ/CANRX), depending on the setting of SPI_PSEL bits in I2C EEPROM offset 0x03, see section 3.1.3 for details.
CANTX	O5	6	CAN serial transmit data. Note that this is a multi-function pin (SS2/STPZ/CANTX), depending on the setting of SPI_PSEL bits in I2C EEPROM offset 0x03, see section 3.1.3 for details. The output driving strength is programmable, by SPI_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
Programmable Counter Array Interface			
ECI	I5	107	Programmable counter array external clock input. Note that this is a multi-function pin (RXD1/ECI), depending on the setting of UIT_PSEL bits in I2C EEPROM offset 0x03, see section 3.1.3 for details.
CEX [4:0]	B5/4m/8m	16, 15, 13, 11, 110	Programmable counter array module 4~0 input and output. Note that these are multi-function pins (TM2_GT/CEX4, TM2_CK/CEX3, TM1_GT/CEX2, TM1_CK/CEX1, TXD1/CEX0), depending on the setting of UIT_PSEL and TM_PSEL bits in I2C EEPROM offset 0x03, see section 3.1.3 for details. The output driving strength is programmable, by the PCA_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
Ethernet PHY Interface			
RXIP	AB	68	Receive differential data input positive pin for 10BASE-T/100BASE-TX in MDI mode or transmit differential data output positive pin in MDIX mode.
RXIN	AB	69	Receive differential data input negative pin for 10BASE-T/100BASE-TX in MDI mode or transmit differential data output negative pin in MDIX mode.
TXOP	AB	71	Transmit differential data output positive pin for 10BASE-T/100 BASE-TX in MDI mode or receive differential data input positive pin in MDIX mode.
TXON	AB	72	Transmit differential data output negative pin for 10BASE-T/100 BASE-TX in MDI mode or receive differential data input negative pin in MDIX mode.

RSET_BG	AO	65	For Ethernet PHY's internal biasing. Please connect to GND3A through a 12.1Kohm +/-1% resistor.
LNK_LED	O5/8m	30	Link status LED indicator. This pin drives low continuously when the Ethernet link is up and drives low and high in turn (blinking) when Ethernet PHY is in receiving or transmitting state. Note that this is a multi-function pin (DB_DI/LNK_LED), depending on the setting of DBG_PSEL bit in I2C EEPROM offset 0x01, see section 3.1.2 for details.
SPD_LED	O5/8m	31	Ethernet speed LED indicator. This pin drives low when the Ethernet PHY is in 100BASE-TX mode and drives high when in 10BASE-T mode. Note that this is a multi-function pin (DB_CKO/SPD_LED), depending on the setting of DBG_PSEL bit in I2C EEPROM offset 0x01, see section 3.1.2 for details.
FD_CL_LED	O5/8m	32	Full duplex and collision detected LED indicator. This pin drives low when the Ethernet PHY is in full-duplex mode and drives high when in half duplex mode. When in half duplex mode and the Ethernet PHY detects collision, it will be driven low. Note that this is a multi-function pin (DB_DO/FD_CL_LED), depending on the setting of DBG_PSEL bit in I2C EEPROM offset 0x01, see section 3.1.2 for details.
MII and Station Management Interface			
MDC	O5/4m/8m/PU	77	Station management clock output and the timing reference for MDIO. All data transferred on MDIO are synchronized to the rising edge of this clock. The frequency of MDC is 1.5MHz. Note that this is a multi-function pin (P10/MDC), depending on the setting of P1_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by the P1_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
MDIO	B5/T/4m/8m/PU	79	Station management data input/output. Serial data input/output transferred from/to the PHYs. The transfer protocol conforms to the IEEE 802.3u MII spec. Note that this is a multi-function pin (P11/MDIO), depending on the setting of P1_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by the P1_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
RX_CLK	I5	5	Receive clock. RX_CLK is received from PHY to provide timing reference for the transfer of MRXD [3:0], RX_DV, and RX_ER signals on receive direction of MII interface. Note that this is a multi-function pin (P20/RX_CLK), depending on the setting of P2_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details.
MRXD [3:0]	I5	17, 14, 12, 9	Receive data. MRXD [3:0] is driven synchronously with respect to RX_CLK by PHY. Note that these are multi-function pins (P24/MRXD3, P23/MRXD2, P22/MRXD1, P21/MRXD0), depending on the setting of P2_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details.
RX_DV	I5	19	Receive data valid. RX_DV is driven synchronously with respect to RX_CLK by PHY. It is asserted high when valid data is present on MRXD [3:0]. Note that this is a multi-function pin (P25/RX_DV), depending on the setting of P2_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details.
RX_ER	I5	27	Receive error. RX_ER is driven synchronously with respect to RX_CLK by PHY. It is asserted high for one or more RX_CLK periods to indicate to the MAC that an error has detected. Note that this is a multi-function pin (P27/RX_ER), depending on the setting of P2_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details.
COL	I5	93	Collision detected. COL is driven high by PHY when the collision is detected. Note that this is a multi-function pin (P37/COL), depending on the setting of P3_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details.

CRS	I5	21	Carrier sense. CRS is asserted high asynchronously by the PHY when either transmit or receive medium is non-idle. Note that this is a multi-function pin (P26/CRS), depending on the setting of P2_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details.
TX_CLK	I5	78	Transmit clock. TX_CLK is received from PHY to provide timing reference for the transfer of MTXD [3:0], TX_EN and TX_ER signals on transmit direction of MII interface. Note that this is a multi-function pin (P30/TX_CLK), depending on the setting of P3_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details.
MTXD [3:0]	O5/4m/8m	88, 86, 82, 80	Transmit data. MTXD [3:0] is transitioned synchronously with respect to the rising edge of TX_CLK. Note that these are multi-function pins (P34/MTXD3, P33/MTXD2, P32/MTXD1, P31/MTXD0), depending on the setting of P3_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by the P3_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
TX_EN	O5/4m/8m	89	Transmit enable. TX_EN is transitioned synchronously with respect to the rising edge of TX_CLK. TX_EN is asserted high to indicate a valid MTXD [3:0]. Note that this a multi-function pin (P35/TX_EN), depending on the setting of P3_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by the P3_ODS bit in I2C Configuration EEPROM offset 0x04 See section 3.1.4 for details.
TX_ER	O5/4m/8m	92	Transmit coding error. TX_ER is transitioned synchronously with respect to the rising edge of TX_CLK. When asserted high for one or more TX_CLK, the PHY shall emit one or more code-groups that are not part of the valid data or delimiter set somewhere in the frame being transmitted. Note that this a multi-function pin (P36/TX_ER), depending on the setting of P3_PSEL bits in I2C EEPROM offset 0x02, see section 3.1.3 for details. The output driving strength is programmable, by the P3_ODS bit in I2C Configuration EEPROM offset 0x04. See section 3.1.4 for details.
Misc. Pins			
RST_N	I5/PU/S	120	Chip reset input, active low. This is the external reset source used to reset this chip. This input feeds to the internal power-on reset circuitry, which then provides the main reset source of this chip.
SYS_RSTO_N	O5/4m	83	Chip system reset output, active low. This output reflects the actual reset state of internal CPU. When low the internal CPU is in reset state, when high the CPU is in normal functional state.
XTL25P	O18	63	25Mhz crystal or oscillator clock output. The recommended reference frequency is 25Mhz +/- 0.005% (i.e. 25Mhz +/- 1250hz).
XTL25N	I18	62	25Mhz crystal or oscillator clock input. The recommended reference frequency is 25Mhz +/- 0.005% (i.e. 25Mhz +/- 1250hz).

<table>
<tr><td>LB_CLK</td><td>B5/8m</td><td>97</td><td>The LB_CLK can be used to provide alternative clock source for the system logic or to provide operating system clock output to peripheral devices. When used as clock source, the input frequency should be as close to 25/50/100 MHz as possible such as 24/48/96Mhz so that the internal timer/counter can work properly. The mode of operation is determined by LB_MOD and SYNC_BUS pin level during chip hardware reset.
<table>
<tr><th>LB_MOD</th><th>SYNC_BUS</th><th>LB_CLK</th></tr>
<tr><td>Pulled-up</td><td>Pulled-up</td><td>The LB_CLK instead of internal 100Mhz PLL is the clock source for operating system clock. In this case, user can provide 24/48/96Mhz clock input to this pin. Also, the SYSCK_SEL should be set to 00/01/11 accordingly so that the internal timer/counter can work properly.</td></tr>
<tr><td>Pulled-down</td><td>Pulled-up</td><td>The LB_CLK is a clock output, which provides the operating system clock of this chip to the external peripheral devices.</td></tr>
<tr><td>Pulled-up</td><td>Pulled-down</td><td>The LB_CLK is not used. In this case, please add a pulled-up resistor to this pin such that it draws minimum current.</td></tr>
</table></td></tr>
<tr><td>SYSCK_SEL[1:0]</td><td>I3</td><td>50, 42</td><td>Operating system clock frequency selection:
00: Set the operating CPU clock to 25Mhz.
01: Set the operating CPU clock to 50Mhz.
10: Reserved.
11: Set the operating CPU clock to 100Mhz.</td></tr>
<tr><td>EXT_WKUP</td><td>I5/PU</td><td>112</td><td>External remote-wakeup trigger input pin, rising edge.
Note that the EXT_WKUP is a dual-function input pin sharing with INT0 pin.</td></tr>
<tr><td colspan="4"><b>On-chip Regulator Pins</b></td></tr>
<tr><td>VCC3R</td><td>P</td><td>75</td><td>3.3V power supply to on-chip 3.3V to 1.8V voltage regulator.</td></tr>
<tr><td>GND3R</td><td>P</td><td>74</td><td>Ground pin of on-chip 3.3V to 1.8V voltage regulator.</td></tr>
<tr><td>VCC18</td><td>P</td><td>76</td><td>1.8V voltage output of on-chip 3.3V to 1.8V voltage regulator. Please add 1uF capacitor between VCC18 and GND3R.</td></tr>
<tr><td colspan="4"><b>Power and Ground Pins</b></td></tr>
<tr><td>VCCK</td><td>P</td><td>7, 26, 36, 59, 84, 99, 125</td><td>Digital core power, 1.8V.</td></tr>
<tr><td>VCCIO</td><td>P</td><td>24, 46, 57, 90, 126</td><td>Digital I/O power, 3.3V.</td></tr>
<tr><td>GND</td><td>P</td><td>29, 55, 96, 128</td><td>Digital ground for core and I/O.</td></tr>
<tr><td>VCC18A</td><td>P</td><td>61, 70</td><td>Analog power for oscillator, PLL, and Ethernet PHY differential I/O pins, 1.8V</td></tr>
<tr><td>GND18A</td><td>P</td><td>64, 73</td><td>Analog ground for oscillator, PLL, and Ethernet PHY differential I/O pins.</td></tr>
<tr><td>VCC3A</td><td>P</td><td>66</td><td>Analog power for bandgap, 3.3V.</td></tr>
<tr><td>GND3A</td><td>P</td><td>67</td><td>Analog ground for bandgap.</td></tr>
</table>

2.0 Function Description

2.1 Clock Generation

The AX11025 integrates an internal 25Mhz oscillator, which allows the chip to operate cost effectively with just an external 25Mhz crystal. The 25Mhz oscillator provides reference clock to the internal PLL circuit, which generate a free-run 100Mhz clock source for system logic and a 125Mhz clock source for the internal Ethernet PHY use. The operating system clock is derived from the 100Mhz clock source from PLL and is programmable between 25Mhz, 50Mhz, and 100Mhz, based on the setting of SYSCK_SEL [1:0] input pins. The users can trade off between system performance and power consumption to decide the best operating system clock frequency.

The AX11025 supports a deep power-down mode (CPU STOP mode) where the internal 25Mhz crystal oscillator and PLL circuit can be completely disabled to consume minimum power. The AX11025 also supports the Power Management Mode (PMM) where the operating system clock frequency is reduced to 1/100 of the original frequency (i.e., 0.25Mhz, 0.5Mhz, and 1Mhz) to reduce power consumption during PMM mode.

The AX11025 also has an external clock source input pin called LB_CLK, which can be used as alternative clock source for system logic. This is typically used when more accurate baud rate generation for UART0/1/2 is needed. For more details on chip clock configuration and distribution, please refer to section 4.1.

2.2 Reset Generation

The AX11025 integrates an internal power-on-reset circuit, which can simplify the external reset circuitry on PCB design. The power-on-reset circuit shall generate a reset pulse to reset system logic after 1.8V core power ramping up to 1.2V (typical threshold). The external hardware reset input pin, RST_N, is fed directly to the input of power-on-reset circuit and can also be used as additional hardware reset source to reset the system logic.

If the internal power-on-reset circuit is used as main reset source, user shall connect RST_N pin to a simple RC reset, which shall generate a low level of at least 4 msec intervals after 1.8V core power ramping up to 1.8V to correctly reset the system logic. If the system has a dedicated reset source connecting to RST_N, this reset source shall also generate a low level of at least 4 msec intervals after 1.8V core power ramping up to 1.8V to correctly reset the system logic.

The optional system reset output pin, SYS_RSTO_N, is a reset indication signal to be used by users who need to control the reset of peripheral devices working with AX11025 in system. This reset output reflects the actual reset state of internal CPU. For more details on chip reset distribution, please refer to section 4.2 and section 5.3.

2.3 Voltage Regulator

The AX11025 contains an internal 3.3V to 1.8V low-dropout-voltage and low-standby-current voltage regulator. The internal regulator provides up to 240mA of driving current for the 1.8V core/analog power of the chip to satisfy the worst-case power consumption scenario. Also for the purpose of lowering power consumption in deep power-down mode or PMM mode, the internal regulator can operate in standby mode to consume less current when the required driving current is less than 30mA. For more details on voltage regulator DC characteristic, please refer to section 5.1.6.

2.4 CPU Core and Debugger

2.4.1 CPU Core

The 1T 8051/80390 CPU core of AX11025 is an ultra high performance, speed optimized, 8-bit embedded controller dedicated for operation with fast (on-chip) and slow (off-chip) memories. The CPU core has been designed with a special concern about performance to power consumption ratio. The CPU core is 100% binary-compatible with the industry standard 8051 8-bit micro-controller. The CPU core can address up to 2 M bytes of linear program space. The CPU core has Pipelined RISC architecture, which can be 10 times faster compared to standard architecture and executes 100 million instructions per second when operating in 100Mhz. The main features of 1T 8051/80390 CPU core are listed below, for more details, please refer to section 4.4.

- 100% software compatible with industry standard 8051
- Maximum operating clock frequency of 100M Hz
- Pipelined RISC architecture enables to execute instructions 10 times faster compared to standard 8051
  - ○ 21-bit FLAT program addressing mode – 80C390 instructions set
  - ○ 16-bit LARGE program addressing mode – 80C51 instructions set
- 24 times faster multiplication
- 12 times faster addition
- 256 bytes of internal (on-chip) Data Memory
- Up to 2M bytes of Program Memory
  - ○ On-chip SRAM used for mirrored program: 0 to 16K bytes
  - ○ On-chip Flash memory used for program: 0 to 512K bytes in FLAT mode
  - ○ Off-chip Flash memory used for program: 512K bytes to 2M bytes in FLAT mode
- Up to 2M bytes of External Data Memory
  - ○ On-chip SRAM used for External Data Memory: 0 to 32K bytes
  - ○ Off-chip SRAM used for External Data Memory: 32K bytes to 2M bytes
- User programmable Program Memory wait states for wide range of memory speed
- User programmable External Data Memory wait states for wide range of memory speed
- Dedicated address/data/controlled bus to allow easy and glueless connection to external Flash/SRAM memory

2.4.2 Debugger

The Debugger inside AX11025 provides an in-circuit emulator feature and it is used to connect to an external In-Circuit-Emulation (ICE) adaptor board, which manages communication between the Debugger inside AX11025 and the Debug Software on a PC. As shown in Figure 5, the Hardware Assisted Debugger (HAD2) is the ICE adaptor board.

The HAD2 is a small hardware adapter that manages communication between the Debugger inside AX11025 and an USB port of the host PC running Debug Software. The USB communication interface to target host PC is at USB Full speed and its power supply comes directly from the USB port.

The Debug Software is a Windows based application. It is fully compatible with all existing 8051/80390 C compilers and Assemblers. The Debug Software allows user to work in two major modes: software simulator mode and hardware debugger mode. Those two modes assure software validation in simulation mode and then real-time debugging of developed software inside AX11025 using debugger mode. Once loaded, the program may be observed in Source Window, run at full-speed, single stepped by machine or C level instructions, or stopped at any of the breakpoints. For more detailed description about the Debug Software, please refer to “AX110xx Software User Guide”.

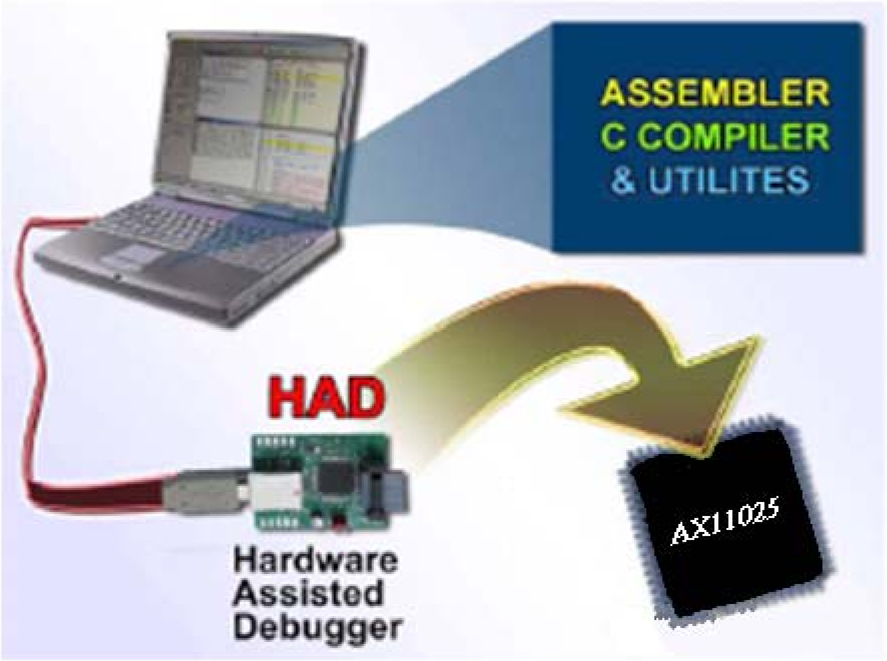


Figure 5: Typical Debugger and Hardware Assisted Debugger (HAD2) System Diagram

The main features of Debugger inside AX11025 are listed below,

- Processor execution control
  - ○ Run, Halt
  - ○ Reset
  - ○ Step into instruction
  - ○ Skip Instruction
- Read-write all processor contents
  - ○ Program Counter (PC)
  - ○ Program Memory
  - ○ Internal (direct) Data Memory
  - ○ Special Function Registers (SFRs)
  - ○ External Data Memory
- Code execution breakpoints - one real-time PC breakpoint
- Hardware execution watch-points
  - ○ Two at Internal (direct) Data Memory
  - ○ Two at Special Function Registers (SFRs)
  - ○ Two at External Data Memory
- Hardware watch-points activated at a
  - ○ certain address by any write into memory
  - ○ certain address by any read from memory

- ○ certain address by write into memory a required data
- ○ certain address by read from memory a required data

- Unlimited number of software watch-points
  - ○ Internal (direct) Data Memory
  - ○ Special Function Registers (SFRs)
  - ○ External Data Memory
- Unlimited number of software breakpoints - Program Memory (PC)
- Automatic adjustment of debug data transfer speed rate between HAD and CPU core
- Communication interface - DTAG three wire communication

2.5 On-Chip Flash Memory

The AX11025 embeds an on-chip Flash memory of 512K bytes. The main features of the Flash memory are listed below,

- Requires only 3.3V power for read, erase and program operations
- Fast read access time: 55ns
- Command register architecture
  - ■ Byte programming time: 9us (typical)
  - ■ Sector Erase (Sector structure 16K Byte x 1, 8K Byte x 2, 32K Byte x1, and 64K Byte x7)
- Auto Erase (chip & sector) and Auto Program
  - ■ Automatically erase any combination of sectors with Erase Suspend capability
  - ■ Automatically program and verify data at specified address
- Erase Suspend/Erase Resume
  - ■ Suspends sector erase operation to read data from, or program data to, any sector that is not being erased, and then resumes the erase operation.
- Status Reply
  - ■ Data# Polling & Toggle bit for detection of program and erase operation completion.
- Sector protection
  - ■ Hardware method to disable any combination of sectors from program or erase operations
  - ■ Temporary sector unprotect allows code changes in previously locked sectors.
- 100,000 minimum erase/program cycles
- 20 years data retention
- Program code download protection in hardware to disable Debugger access for preventing unauthorized program code downloading.

For more detailed description, please refer to section 4.5.

2.6 Memory Arbiter and Boot Loader

The external memory interface of AX11025 allows simple and easy connections for up to one external Flash memory chip and two external SRAM chips without any glue logic. The external Flash memory provides additional program code storage for those applications, which require more than 512K bytes of program code size. The external SRAM chips provide data memory expansion for those applications, which require more than 32K bytes of data storage. The external SRAM chips can also function as program code storage in “program code shadow” mode for applications, which require higher performance.

The addressable memory space of first external SRAM chip using chip enable pin, XRAMCE0_N, is programmable (via the ASCS bits in I2C Configuration EEPROM offset 0x01, see section 3.1.2 for details), which allows user to choose from 64K bytes, 128K bytes, 256K bytes, 512K bytes, or 1024K bytes of SRAM chip whichever is most cost effective.

Depending on system applications and requirements, following scenarios of Flash memory and SRAM configurations can be possibly used, as shown in Figure 6, Figure 7, Figure 8, and Figure 9.

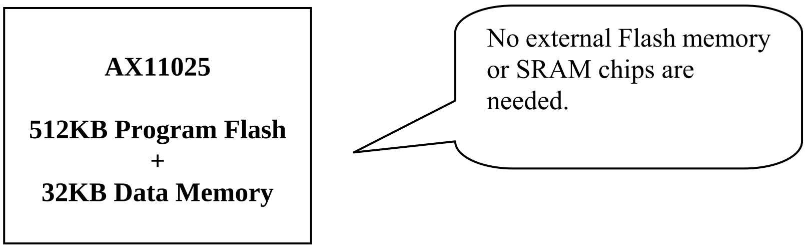


Figure 6: Systems requiring only 512K bytes of Program Flash and 32K bytes of Data Memory

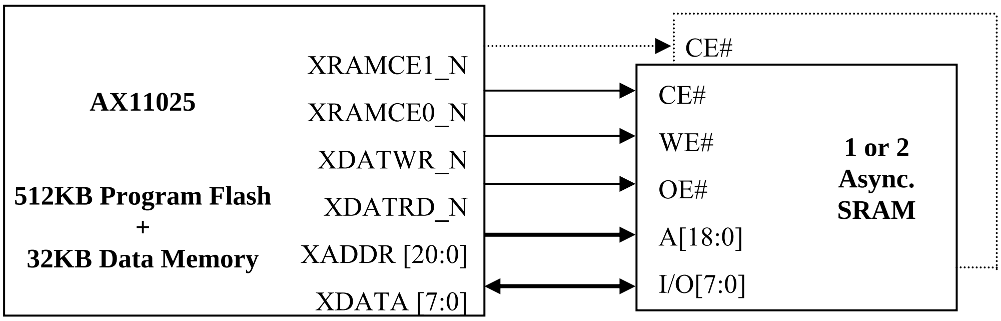


Figure 7: Systems requiring 512K bytes Program Flash and over 32K bytes Data Memory

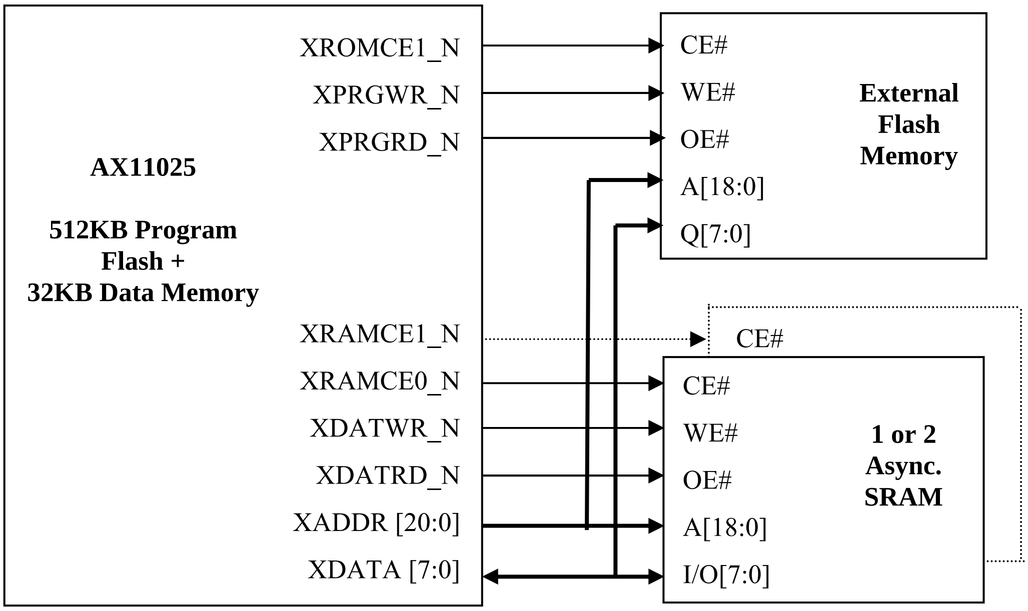


Figure 8: Systems Requiring over 512K bytes Program Flash and over 32K bytes Data Memory

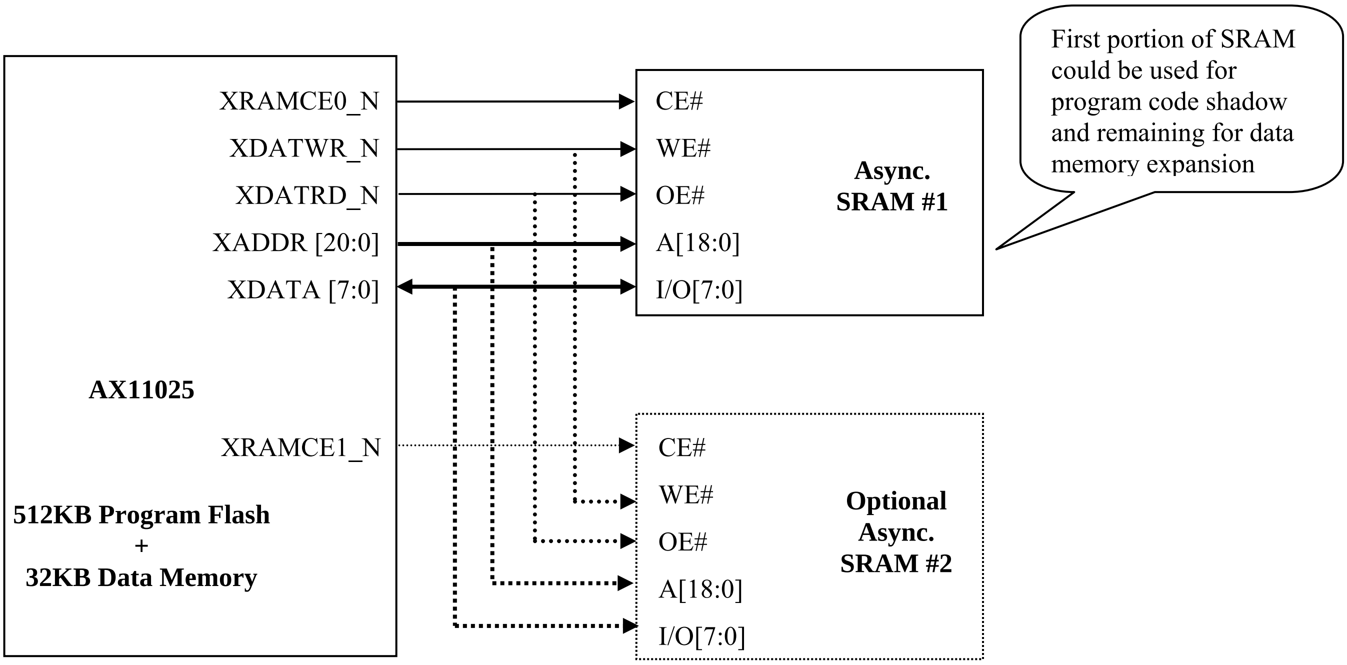


Figure 9: Systems requiring 512K bytes Program Flash and over 32K bytes Data Memory for High Performance

The memory arbiter and boot loader of AX11025 support three major functions - Boot loader, Memory arbiter, and Flash programming controller, as described in following sections.

2.6.1 Boot Loader

The boot loader shall activate right after hardware reset (either power-on-reset or RST_N input) or software reboot command (via SFR register CSREPR). It shall automatically perform copying the program code from Flash memory to on-chip 16KB SRAM for "program code mirroring", and upon enabled (via EXT_PROG_SRAM_EN pin), also automatically perform copying the program code from on-chip/off-chip Flash memory to off-chip SRAM for "program code shadow" mode for high performance applications.

The "program code mirroring" allows the program code residing on on-chip Flash memory space 0~16K bytes to be mirrored to on-chip 16Kbytes SRAM before the 1T 80390 CPU starts running. This on-chip 16Kbytes SRAM located at program memory space 0~16K bytes of the 1T 80390 CPU will be used to execute program code with 0 wait state to achieve top performance of 100 MIPS. During time of firmware update via Ethernet or UART, the 16K bytes of mirrored program code on SRAM shall perform Flash write commands to write new firmware into the Flash memory. This allows the program code being executed continuously while the Flash memory is being updated.

The "program code shadow" mode allows both the program code residing on on-chip Flash memory space 16K~512K bytes and the program code residing on off-chip Flash memory to be shadowed to off-chip SRAM chips before the 1T 80390 CPU starts running. The off-chip SRAM chips located at program memory space 16K~2M bytes of the 1T 80390 CPU can then be used to execute program code with 0 or 1 wait state. For more details, please refer to section 4.6.

2.6.2 Memory Arbiter

The memory arbiter manages Program memory and External Data (xDATA) memory bus access. It arbitrates the access of xDATA memory between 1T 80390 CPU and the Direct Memory Access (DMA) engine, and upon enabling "program code shadow", it also redirects the 1T 80390 CPU program access from Flash memory to off-chip SRAM chips.

The xDATA memory access could come from 1T 80390 CPU, the DMA from TCP/IP Offload Engine (TOE), and DMA for software DMA. The arbitration priority is that, the 1T 80390 CPU's access to Program memory and xDATA memory has the highest priority, the DMA for software DMA is given the second priority, and the DMA for TOE is the last.

Upon enabling "program code shadow" mode, the memory arbiter shall redirect 1T 80390 CPU's program access from Flash memory to external SRAM chips. Upon enabled, the memory arbiter also allows the external SRAM chips to be used as "program code shadow" and "xDATA memory expansion" purposes at the same time (shared memory architecture). Therefore, allowing 1T program code execution as well as xDATA memory expansion capability cost effectively on the same SRAM chips. For more details, please refer to section 4.6.

2.6.3 Flash Programming Controller

The Flash programming controller supports In-System-Programming (ISP) for both on-chip Flash memory of AX11025 and off-chip Flash memory on PCB via UART 0 interface of AX11025. When enabled (via BURN_FLASH_EN pin), it allows on-chip/off-chip Flash memory to be programmed by ASIX's Flash Programming utilities software on a PC with a standard RS-232 port, as shown in Figure 10. The link speed of AX11025's UART 0 used for communicating to the PC's RS-232 port can be chosen to be either 921.6K or 115.2K bps (via BURN_FLASH_921K pin). When developing software for AX11025 or manufacturing the system with AX11025 on it, the ASIX's Flash Programming utilities software can provide easy and fast Flash memory update capability.

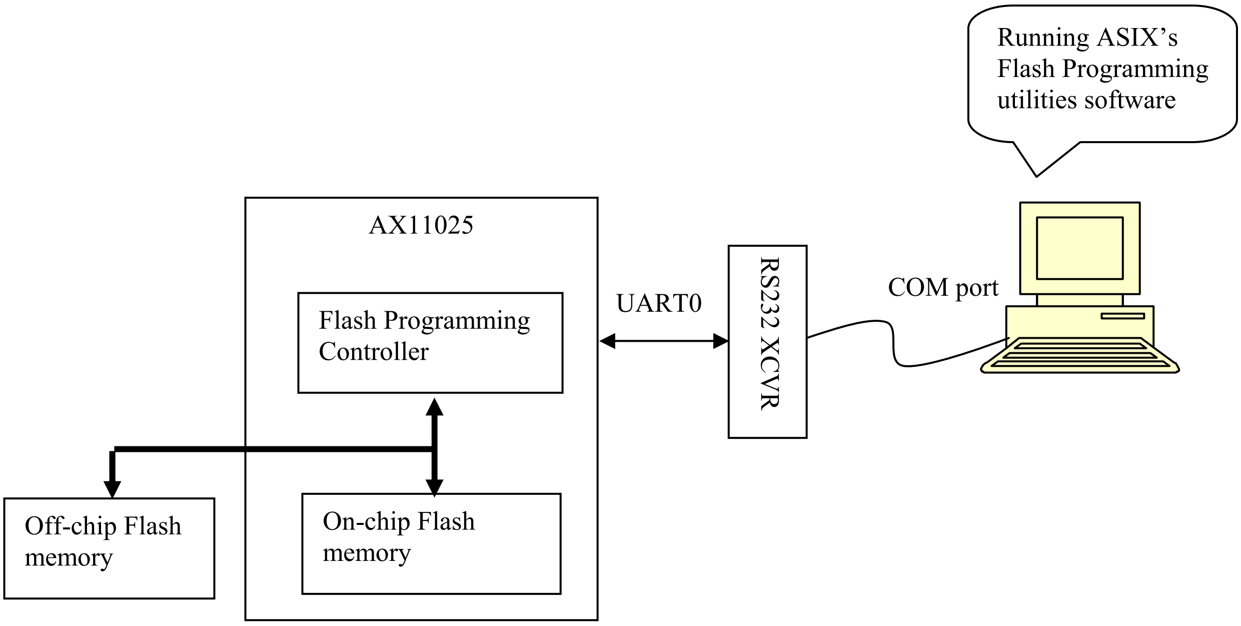

Running ASIX's Flash Programming utilities software
AX11025
Flash Programming Controller
UART0
RS232 XCVR
COM port
Off-chip Flash memory
On-chip Flash memory

Figure 10: Flash Memory Programming System Configuration

During Flash programming process, the Flash Programming Controller (FPC) in AX11025 shall receive commands from Flash Programming utilities software through the UART 0 interface. The commands received are in form of packets from which FPC will decode, execute, and then acknowledge the result back to the software utilities. The command handshaking structure is simple and flexible to simplify the FPC design while at the same time addressing the long programming time, complex programming procedures, command compatibility issues of Flash memory. For more details, please refer to section 4.6.

2.7 DMA Engine

The direct memory access (DMA) engine of AX11025 handles External Data (xDATA) memory read and write access for TCP/IP Offload Engine (TOE) as well as bulk data copy for software DMA.

The TOE can receive packets from Ethernet MAC and store them in xDATA memory via DMA write access, or it can transmit packets to Ethernet MAC from xDATA memory via DMA read access.

The DMA engine also can support software DMA, which performs bulk data copy from one region of xDATA memory to another region in a timely manner, based on software configuration. The hardware based DMA engine can greatly reduce the time spending in bulk data movement very often needed in network protocol stack processing, and, hence, help achieve better performance on micro-controller computing power. For more details, please refer to section 4.7.

2.8 Interrupt Controller

The interrupt controller of AX11025 supports 2 external interrupt pins, INT0 and INT1, with each having two levels of interrupt priority control. They can be in high or low-level priority group (setting via SFR register IP, EIP). The 2 external interrupt pins can be activated at low level or by a falling edge.

As shown in Table 2 below, the interrupt controller also supports various interrupt requests internal to the AX11025, again each having two levels of interrupt priority control. For more details, prefer to section 4.8.

Interrupt Sources	Function Description	Active level	Vector	Natural Priority
INT 0	The external interrupt input pin, INT0	Active low or falling edge	0x03	1
Timer 0	The internal Timer 0 interrupt request		0x0B	2
INT 1	The external interrupt input pin, INT1	Active low or falling edge	0x13	3
Timer 1	The internal Timer 1 interrupt request		0x1B	4
UART 0	The internal UART 0 interrupt request		0x23	5
Timer 2	The internal Timer 2 interrupt request		0x2B	6
UART 1	The internal UART 1 interrupt request		0x33	7
INT 2	The internal DMA transfer interrupt request for TOE/software DMA mode, please set to high priority		0x3B	8
INT 3	The internal programmable counter array interrupt request		0x43	9
INT 4	The internal peripheral interrupt request for TOE, MAC/PHY, I2C, SPI, 1-Wire, UART2, CAN, etc.		0x4B	10
INT 5	The internal software DMA complete and millisecond timer timeout interrupt		0x53	11
INT 6	The wake-up interrupt request (resume from CPU STOP mode)		0x5B	12
Watchdog	Internal watchdog interrupt		0x63	13

Table 2: Interrupt Controller Summary

2.9 Watchdog Timer

The watchdog timer of AX11025 is a user programmable clock counter that can serve as:

- A time-base generator
- An event timer
- System supervisor

As shown in Figure 11, the watchdog timer is driven by the main system clock, which is supplied to a series of dividers. The divider output is selectable, and determines interval between timeouts. When the timeout is reached, an interrupt flag will be set, and if enabled, a reset will occur (to reset CPU core). The interrupt flag will cause an interrupt to occur if its individual enable bit is set and the global interrupt enable is set. The reset and interrupt are discrete functions that may be acknowledged or ignored, together or separately for various applications. For more details, please refer to section 4.9.

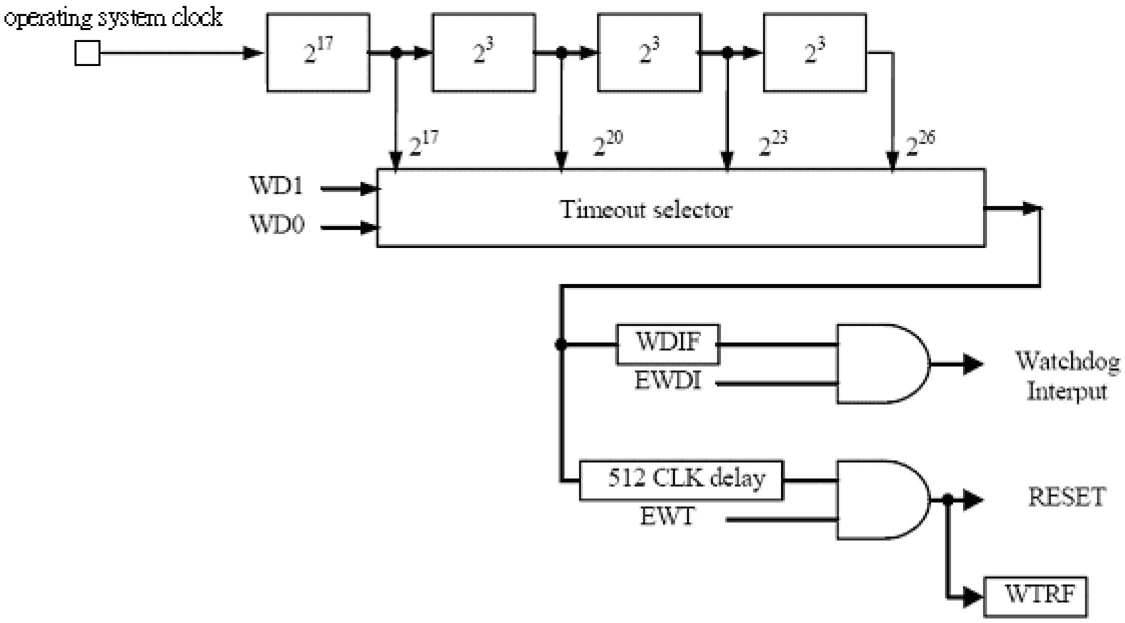


Figure 11: Watchdog Timer Block Diagram

2.10 Power Management Unit

The power management unit of AX11025 supports two power conservation modes - Power Management Mode (PMM) and STOP mode.

2.10.1 PMM

When entering the PMM (via SFR register PCON) from full speed mode, most system logic of AX11025 shall run at slower clock frequency (1/100 of original clock frequency) to reduce power consumption. The PMM is entered and exited by setting the PMM bit (PCON.0) by software. The PMM mode also supports the "switchback" feature using SWB bit (PCON.2). The "switchback" feature of PMM allows the AX11025 to almost immediately return to the full speed mode from PMM, upon acknowledgement of an interrupt or a falling edge on a serial port receiver pin. The following events can trigger AX11025 switchback to full speed mode from PMM:

- Receive interrupt on external interrupt pin, INT0 or INT1
- Detect falling-edge transition (start bit) on RXD0 pin of UART 0 or RXD1 pin of UART 1
- Transmit buffer loaded on UART0 or UART1
- Watchdog timer reset

In addition, the following events can also trigger AX11025 switchback to full speed mode from PMM, via INT 6:

- Receive rising-edge signal on external remote-wakeup trigger input pin, EXT_WKUP, if enabled
- Receive Magic packet from Ethernet, if enabled
- Receive pre-defined Wakeup frame from Ethernet, if enabled
- Detect link-up signal from the embedded Ethernet PHY, if enabled
- Detect link-up signal from secondary PHY, if enabled
- Detect falling-edge transition (start bit) on RXD2 pin or falling-edge transition on RI pin of UART 2, if enabled

2.10.2 STOP Mode

When entering the STOP mode (via SFR register PCON), the main system clock for most system logic of AX11025 shall be completely disabled to further reduce power consumption. The AX11025 supports entering the STOP mode with options of internal 25Mhz crystal oscillator and PLL circuit either still running or completely disabled via TOFFOP bit (Flag.1) in I2C Configuration EEPROM offset 0x01. The lowest power consumption that AX11025 can enter is the STOP mode with 25Mhz crystal oscillator and PLL circuit completely disabled.

The software can enter the STOP mode from full speed mode or PMM by setting the STOP bit (PCON.1). After entering the STOP mode, no processing is possible, timers are stopped, and no serial communication is possible. A NOP instruction has to be added after an instruction that sets STOP bit. The NOP is added because of pipelining architecture of 1T 80390 CPU. The CPU operation will be postponed on the instruction that sets the STOP bit.

If the STOP mode is entered with 25Mhz oscillator and PLL completely disabled, the STOP mode can be exited in following ways:

- Receive rising-edge signal on external remote-wakeup trigger pin, EXT_WKUP, if enabled
- Detect falling-edge transition (start bit) on RXD2 pin or falling-edge transition on RI pin of UART 2, if enabled
- Receive hardware reset on RST_N pin (CPU operation will resume execution at address 0x00_0000)

If the STOP mode is entered with 25Mhz oscillator and PLL still running, the STOP mode can be exited in following ways, depending on software configuration before entering the STOP mode:

- Receive rising-edge signal on external remote-wakeup trigger pin, EXT_WKUP, if enabled
- Receive Magic packet from Ethernet, if enabled

- Receive pre-defined Wakeup frame from Ethernet, if enabled
- Detect link-up signal from the embedded Ethernet PHY, if enabled
- Detect link-up signal from secondary PHY, if enabled
- Detect falling-edge transition (start bit) on RXD2 pin or falling-edge transition on RI pin of UART 2, if enabled
- Receive hardware reset on RST_N pin (CPU operation will resume execution at address 0x00_0000)

Note that above trigger events use the non-clocked interrupt, INT6, to wake up 1T 80390 CPU and to re-enable the main system clock. The clocked interrupts such as the watchdog timer, internal timers, and serial ports (UART0/1) do not operate in STOP mode, therefore, can't be used as a trigger event to wake up from STOP mode. The 1T 80390 CPU operations will resume with the fetching of the interrupt vector associated with the interrupt that caused the exit from STOP mode. When the interrupt service routine will complete, RETI returns the program to the instruction immediately following the one that invoked the STOP mode. For more detailed description, please refer to section 4.10.

2.11 Timers and Counters

The AX11025 contains three 16-bit timer/counters, namely, Timer 0, Timer 1, and a fully compatible with the standard 8052 Timer 2, and one dedicated Millisecond Timer which is programmable with 1ms resolution for software use.

In the "timer mode", timer registers are incremented every 12 or 4 operating system clock periods when appropriate timer is enabled. In the "counter mode" the timer registers are incremented every falling transition on their corresponding input pins: TM0_CK, TM1_CK, or TM2_CK. The input pins are sampled every operating system clock period. The Timers 0, 1, 2 block diagram is shown in figure below. For more details, please refer to section 4.11.

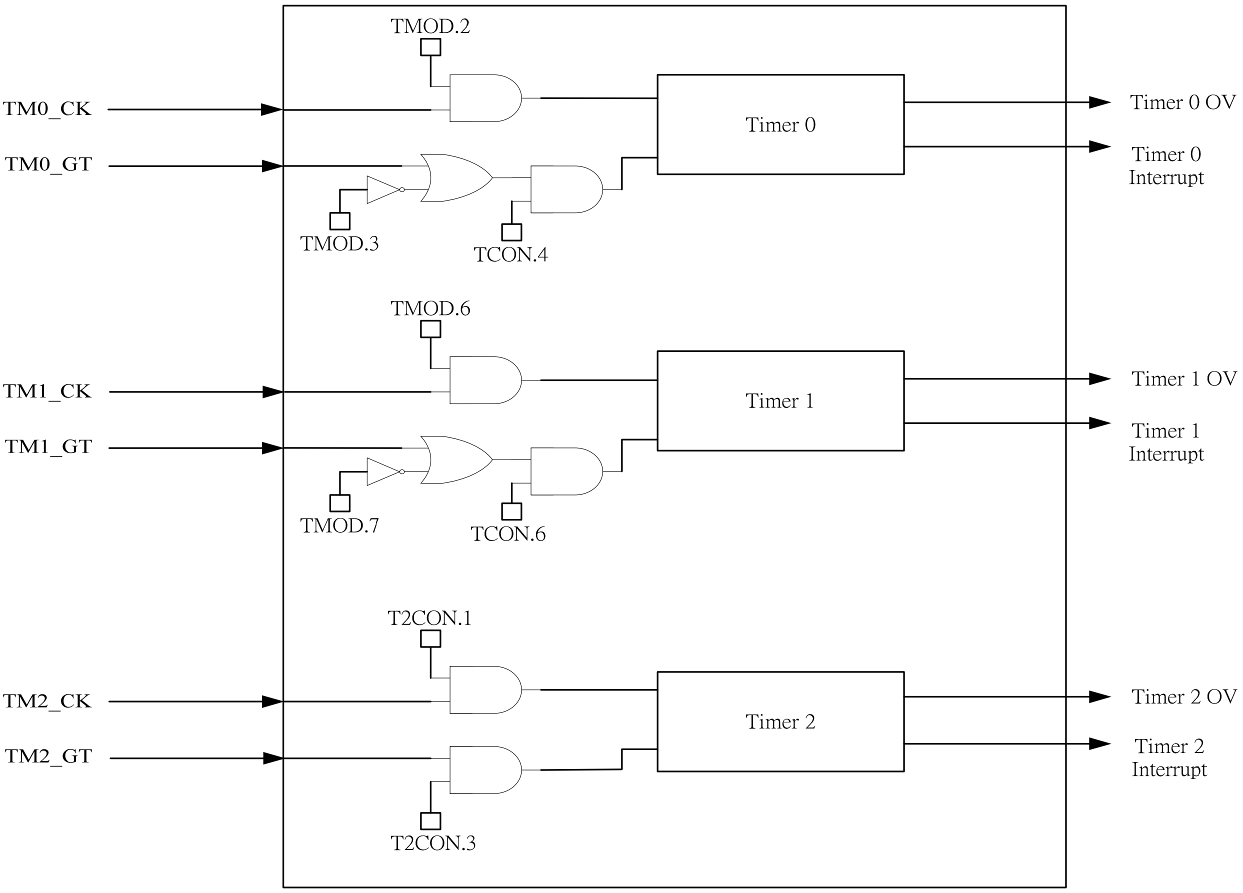


Figure 12: Timers 0, 1, and 2 Block Diagram

2.12 UARTs

The AX11025 contains 3 UART interfaces, namely, UART 0, UART 1, and UART 2.

2.12.1 UART 0 and UART 1

The UART 0 and UART 1 of AX11025 have the same functionality as standard 8051 UARTs. Each is full duplex, meaning it can transmit and receive concurrently. Each is receive double-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register.

UART 0 can operate in following 4 modes:

- Mode 0, synchronous mode
- Mode 1, 8-bit UART, variable baud rate, Timer 1 or Timer 2 clock source
- Mode 2, 9-bit UART, fixed baud rate
- Mode 3, 9-bit UART, variable baud rate, Timer 1 or Timer 2 clock source

UART 1 can operate in following 4 modes:

- Mode 0, synchronous mode
- Mode 1, 8-bit UART, variable baud rate, Timer 1 clock source
- Mode 2, 9-bit UART, fixed baud rate
- Mode 3, 9-bit UART, variable baud rate, Timer 1 clock source

The Figure 13 below shows the I/O buffer of RXD0/1 pin of UART 0/1, the RXD0/1 pin is tri-stated when RXD0/1_out is high. For more details, please refer to section 4.12.

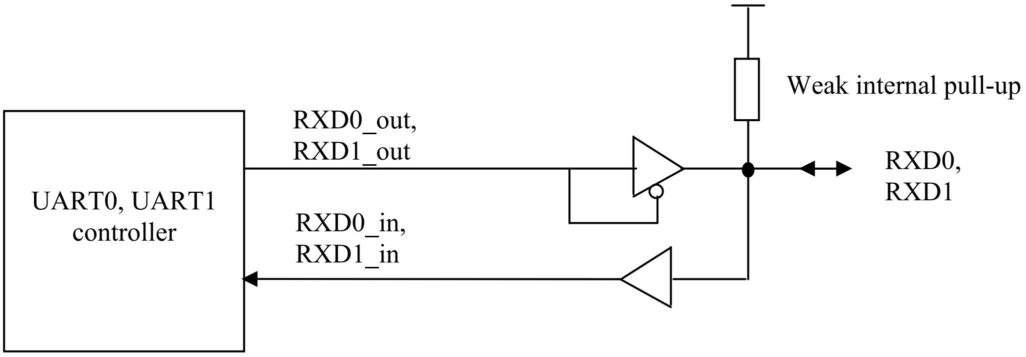


Figure 13: I/O Buffer of RXD0 pin of UART 0 and RXD1 pin of RXD1

2.12.2 UART 2

The UART 2 of AX11025 is designed to be maximally compatible with standard 16550. It can communicate with MODEM or other external device (e.g. computer) by using RS-232 protocol. The UART 2 has 16-bytes deep transmit/receive FIFO and its transfer rate can be up to 921600 bps. The UART 2 includes a programmable baud rate generator capable of dividing the operating system clock by (27*N), where N = 1~65535, for generating wide range of baud rate for the internal transmitter/receiver logic. The main features of UART 2 are listed below,

- 16 bytes deep receive and transfer FIFO

- Support up to 921600 bps baud
- Detection of bad data in the receiver FIFO
- Full-duplex asynchronous channel
- Automatic send data control (ASDC) for automatically transmitter/receiver enable control for RS-485
- Modem control functions (CTS, RTS, DSR, DTR, RI and DCD)
- Fully programmable serial interface
  - Even, odd, no parity bit generation and detection
  - 5, 6, 7, 8 data bit
  - 1, 1.5, 2 stop bit generation
- Line break generation and detection
- Internal diagnostic capabilities (loopback controls, break, parity, overrun and framing error)
- Transmit, receive, line status, and data set interrupts independently controlled
- Complete status reporting capabilities
- Remote wakeup by detecting falling-edge transition (start bit) on RXD2 pin or falling-edge transition on RI pin

For more details, please refer to section 4.12.

2.13 GPIOs

The AX11025 supports four 8-bit bi-directional, open-drain, general purpose input and output ports, namely, P0 [7:0], P1 [7:0], P2 [7:0] and P3 [7:0]. Each port bit can be individually accessed by bit addressable instructions. The driving strength of the GPIO ports is programmable (4mA or 8mA, via I2C Configuration EEPROM offset 0x04, see section 3.1.4 for details). The Figure 14 below shows the I/O buffer of GPIO pins. For example, the P00 pin is tri-stated when P00_out is high. For more details, please refer to section 0.

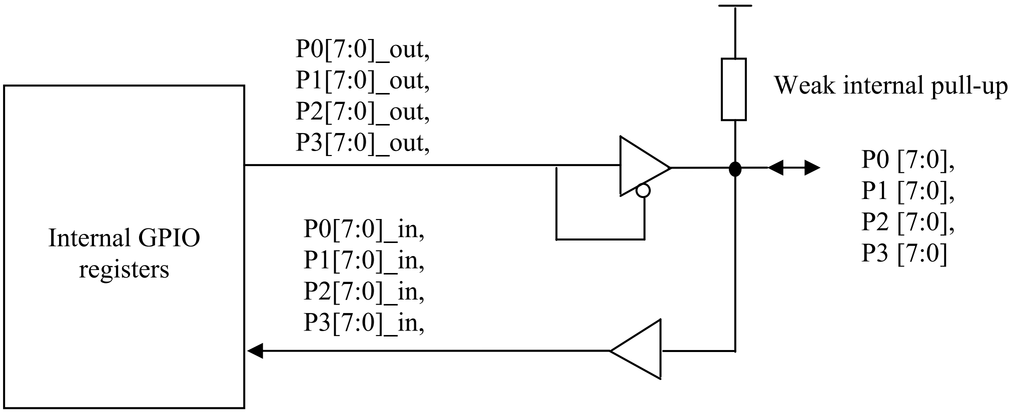


Figure 14: The I/O Buffer of GPIO Pins

2.14 TCP/IP Offload Engine

The TCP/IP Offload Engine (TOE) of AX11025 supports some network layer 2 to 4 header processing functions in hardware. The layer-2 function of TOE interfaces to Ethernet MAC, while its layer-4 function interfaces to DMA engine for receiving/transmitting network packets to/from xDATA memory of AX11025. The TOE can operate in two different modes - "Non-Transparent" mode and "Transparent" mode.

When TOE operating in "Transparent" mode, it supports following features,

- VLAN ID filtering for received packets, if enabled
- On-the-fly IPv4 packet header checksum check and generation (with or without PPPoE header, RFC2516)
- Received packet filtering for IPv4 packets with error header checksum
- On-the-fly TCP and UDP segment checksum check and generation
- On-the-fly ICMP and IGMP message checksum check and generation
- Received packet filtering for TCP/UDP/ICMP/IGMP packets with error checksum

When TOE operating in "Non-Transparent" mode, it supports following features,

- Layer-2 functions (the recognizable packet types are Ethernet II encapsulation (RFC894), IEEE 802.2/802.3 SNAP encapsulation (RFC1042), IEEE 802.2/802.3 encapsulation, and NetWare 802.3 RAW encapsulation)
  - Ethernet MAC frame header parsing and encapsulation, including DA, SA, Length/Type, VLAN Tag fields.
  - ARP Cache:
    - When receiving, automatically learns the source IP address and SA of the received Ethernet MAC frames into ARP Cache SRAM
    - When transmitting, automatically sends out ARP-Request packet when the ARP Cache is not found
    - Upon receiving ARP-Request packet, automatically responds with ARP-Reply packet and updates ARP Cache
    - Upon receiving ARP-Reply packet, automatically updates ARP Cache
    - Software programmable timeout value for ARP Cache Timeout
    - ARP Cache SRAM is software accessible
  - VLAN ID filtering for received packets and VLAN Tag insertion for transmit packets, if enabled
  - Received packet filtering for ARP-Request packet
  - Remove layer 2 header of receive IPv4-type packets before forwarding up to Layer-3 function
  - Append layer 2 header of transmit IPv4-type packets from Layer-3 function before passing down to Ethernet MAC
- Layer-3 functions:
  - IPv4 header parsing, including version, header length, total length, protocol, header checksum, source IP address, destination IP address fields
  - On-the-fly IPv4 header checksum check and generation (only when without PPPoE header bytes)
  - Received packet filtering for IPv4 packets with version not equal to 4 or error header checksum
  - Received packet filtering for IPv4 packets with wrong destination IP address (not equal to owned IP address, and not equal to broadcast IP address, and not equal to multicast IP address) and wrong source IP address (equal to broadcast IP address, or equal to multicast IP address)
- Layer 4 functions:
  - On-the-fly TCP and UDP segment checksum check and generation
  - On-the-fly ICMP and IGMP message checksum check and generation

- Received packet filtering for TCP/UDP/ICMP/IGMP packets with error checksum

For more detailed description on TOE, please refer to section 4.14.

2.15 10/100M Ethernet MAC

The 10/100M Ethernet MAC of AX11025 supports 802.3 and 802.3u MAC sub-layer functions as listed below,

- MAC frame receive and transmit through MII interface
- With dedicated receive buffer of 8K bytes SRAM and transmit buffer of 4K bytes SRAM
- Flow-control support in full-duplex mode by monitoring receive buffer usage to compare with high water mark and low water mark for triggering flow control
- Received MAC frame CRC check and transmit MAC frame CRC generation
- Received packet filtering for broadcast, multicast, unicast, or CRC error MAC frames, etc. if enabled
- Support collision-detection, exponential backoff, packet retransmission, and backpressure in half-duplex mode
- Support Magic packet, predefined Wakeup frame, and Ethernet PHY linkup remote-wakeup mode. Upon detecting wakeup event, it can awake the AX11025 up from PMM or STOP mode
- Provides additional media-independent interface (MII) interface for interfacing external HomePlug PHY or HomePNA PHY functions

The 10/100M Ethernet MAC interfaces to both MII interface of the embedded 10/100M Ethernet PHY and the external MII interface I/O pins. The selection between the two MII interfaces is controlled by software. Figure 15 shows the data path diagram of the 10/100M Ethernet MAC, the embedded 10/100M Ethernet PHY, and external MII interface. For more detailed description, please refer to section 4.15.

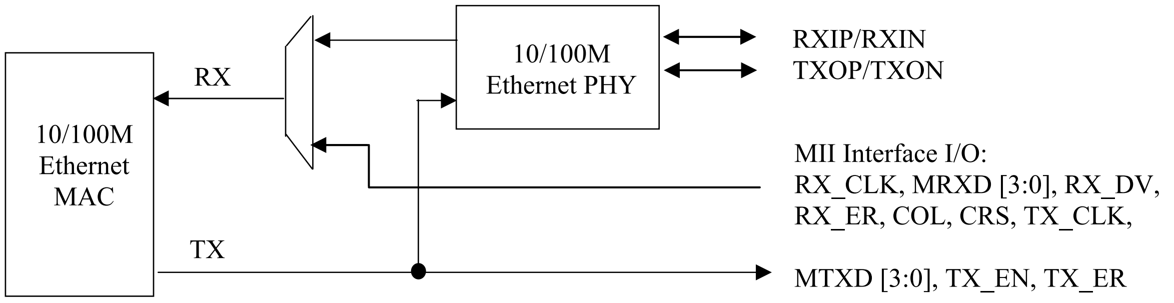


Figure 15: Internal Data path Diagram of 10/100M Ethernet MAC, 10/100 Ethernet PHY, and MII Interface

The 10/100M Ethernet MAC also provides a two-wire, serial interface to connect to a managed PHY device for the purposes of controlling the PHY and gathering status from the PHY. This interface allows communicating with multiple PHY devices at the same time by identifying the managed PHY with 5-bit, unique PHY ID. The PHY ID of the embedded 10/100 Ethernet PHY is being pre-assigned to "1_0000".

Figure 16 shows the internal control mux for the station management interface. When doing read, the "mdin" signal will select from embedded 10/100 Ethernet PHY only if PHY ID matches with "1_0000", otherwise, it will always select from external MDIO pin of the AX11025.

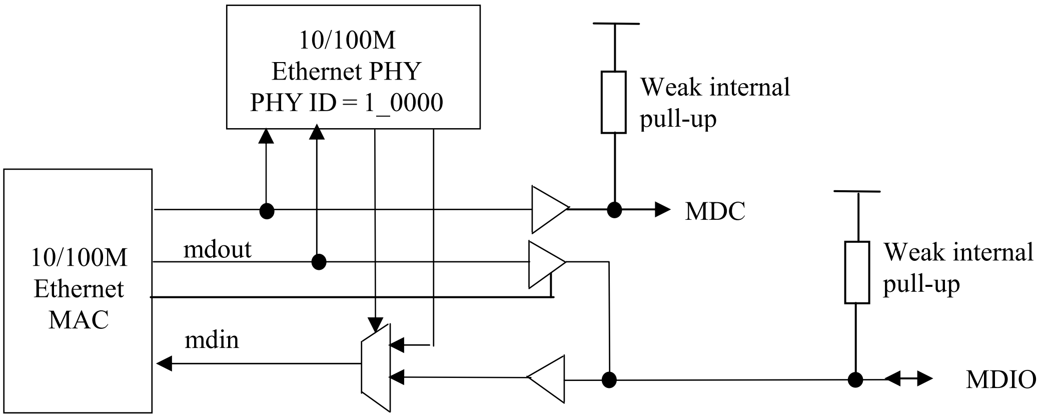


Figure 16: Internal Control Mux for Station Management Interface

2.16 10/100M Ethernet PHY

The 10/100 Ethernet PHY of AX11025 is compliant with IEEE 802.3 and IEEE 802.3u standards. It contains an on-chip crystal oscillator, PLL-based clock multiplier, and digital phase-locked loop for data/timing recovery. It provides over-sampling mixed-signal transmit drivers complying with 10/100BASE-TX transmit wave-shaping / slew rate control requirements. It has robust mixed-signal loop adaptive equalizer for receiving signal recovery.

- Support full-duplex mode, half-duplex mode, and auto-negotiation
- Support twisted pair crossover detection and auto-correction (Auto-MDIX)
- DSP-based adaptive line equalizer, providing superior immunity to near end crosstalk and inter-symbol interference
- Fully compliant with 100BASE-TX, and 10BASE-T PMD level standards (IEEE 802.3u and IEEE 802.3)
- DSP-controlled symbol timing recovery circuit
- Baseline wander corrective circuits to compensate data dependent offset due to AC coupling transformers
- Over-sampling mixed-signal transmit driver complies with 10/100BASE-TX transmit wave-shaping/slew-control requirements

For more detailed description, please refer to section 4.16.

2.17 Programmable Counter Array

The programmable counter array (PCA) present on the AX11025 is a special 16-bit timer that has five 16-bit capture/compare modules. It provides more timing capabilities with less CPU intervention than the standard timer/counter. Its advantages include reduced software overhead and improved accuracy.

As shown Figure 17 below, the PCA have 6 I/O pins, one external clock input pin, ECI, and five capture/compare signal pins, CEX [4:0]. The PCA consists of a dedicated timer/counter, which serves as the time base for an array of five compare/capture modules. Each of the five modules can be programmed in any of the following modes: rising and/or falling edge capture, software timer, high speed output, and pulse width modulator (PWM). For more details, please refer to section 4.17.

The PCA timer/counter uses operating system clock, Timer 0 overflow, and ECI, to generate reference clock for capture/compare modules. The 5 capture/compare modules use CEX [4:0] pins to communicate to external resource. The output driving strength of CEX [4:0] is programmable (4mA or 8mA, by PCA_ODS bit in I2C Configuration EEPROM offset 0x04, see section 3.1.4 for details).

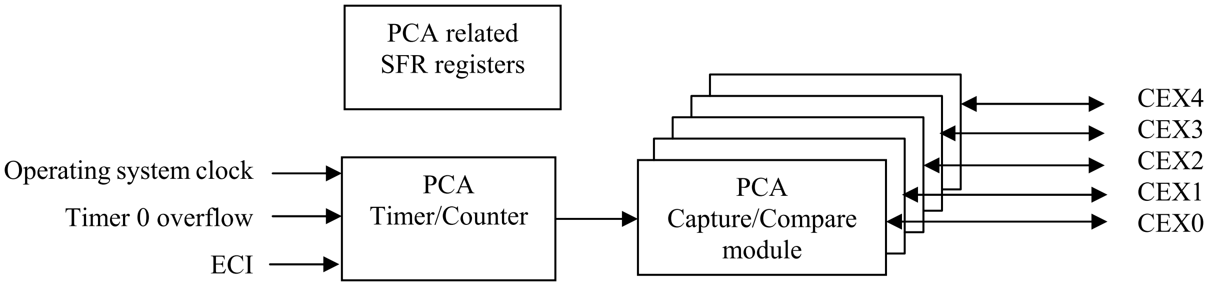


Figure 17: Programmable Counter Array Block Diagram

2.18 I2C Controller

The I2C controller of AX11025 supports Standard-mode (100K bps) and Fast-mode (400K bps), but not High-speed mode (3.4M bps) of the standard I2C bus spec. As shown in Figure 18, the I2C controller consists of an I2C master controller to support communication to external I2C devices, an I2C slave controller to support communication to external micro-controller with I2C master, and an I2C boot loader to support communication to external I2C EEPROM being used for storing chip configuration data. The output driving strength of I2C pins, SCL and SDA, is programmable (4mA or 8mA, by I2C_ODS bit in I2C Configuration EEPROM offset 0x04, see section 3.1.4 for details).

The I2C master controller is compatible with I2C bus protocol. It provides eight registers to fully control and monitor I2C bus transaction, and it has separate receive and transmit registers to hold data for transactions between AX11025 and the external I2C devices. The I2C master controller also provides arbitration for multi-master operation scenario and reports the arbitration status. Also, the I2C master controller accepts the SCL being extended low by the slow I2C slave devices as additional wait state indication during data or acknowledge cycles.

The I2C slave controller allows an external micro-controller with I2C master to communicate with AX11025. It provides an I2C device ID register to allow flexible assignment of AX11025 with any I2C device address for either 7-bit or 10-bit address mode, and can automatically filter I2C bus transactions not belonging to AX11025 in hardware. The I2C slave controller can extend the SCL line low when it needs additional wait state to respond to the external I2C master's bus transaction. The I2C slave controller supports 6 flexible command instructions for the external micro-controller to access the internal registers and memory resources of AX11025. For more details, please refer to section 4.18.

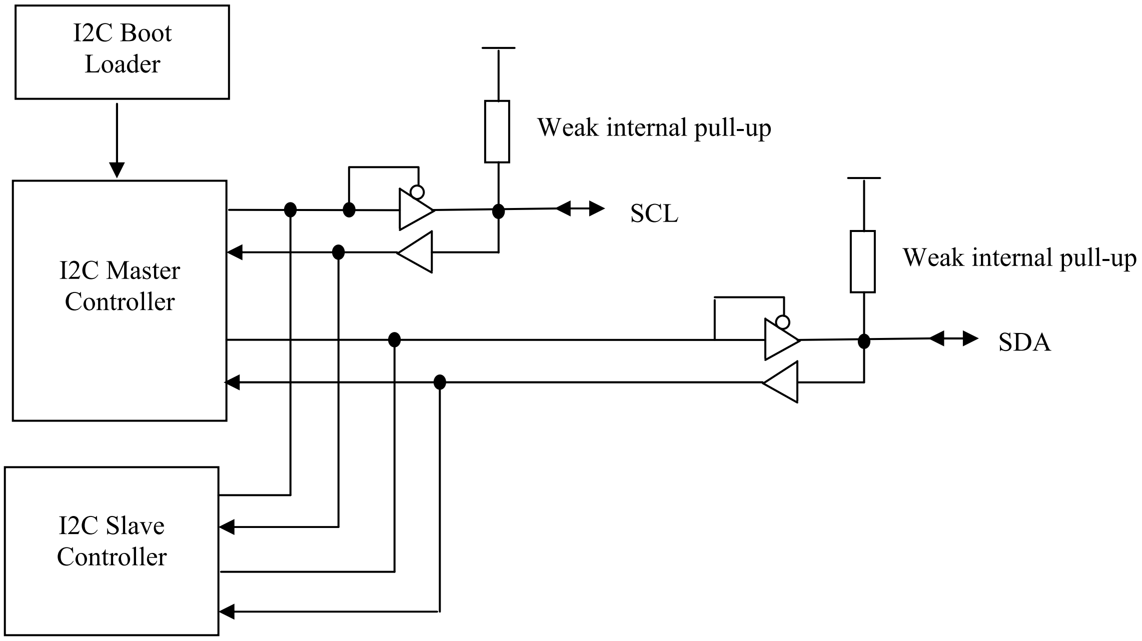


Figure 18: I2C Controller Block Diagram

The I2C boot loader is used to load chip configuration data from external I2C EEPROM. It is activated after hardware reset (either power-on-reset or RST_N input) or via the software reload command (via I2CCTR register). The detailed memory map of I2C EEPROM is described in section 3.1. The use of external I2C EEPROM is optional, when not used, the I2C_BOOT_DIS pin should be pulled up during chip hardware reset, in that case, the reset value listed in I2C EEPROM memory map shall be used by this chip by default.

2.19 1-Wire Controller

The 1-Wire controller of AX11025 is a master-mode controller that controls the communication with multiple external 1-Wire devices. The data transmissions on 1-Wire bus are bit-asynchronous and half-duplex mode only. The 1-Wire controller provides some registers for software to easily perform reading/writing data from/to the 1-Wire devices without having to deal with time-consuming bus timing and control sequences on 1-Wire bus. It supports Standard mode, Standard – Long line mode, and Overdrive mode to work with various 1-Wire devices.

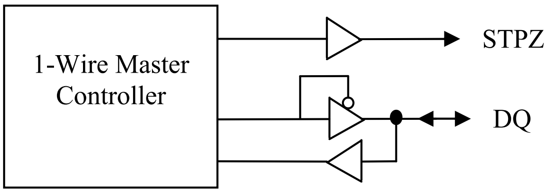


Figure 19: 1-Wire Controller Block Diagram

The 1-Wire controller also supports Search ROM Accelerator, which relieves CPU from any single bit operations on the 1-Wire Bus. As shown in Figure 19 above, it also provides a strong pull-up control pin, STPZ, for the case of large loading or long line conditions. The DQ is an open-drain pin, which needs an external pulled-up resistor or a strong pull-up through a PMOS transistor. The driving strength of DQ and STPZ is programmable (4mA or 8mA, by SPI_ODS bit in I2C Configuration EEPROM offset 0x04, see section 3.1.4 for details).

2.20 SPI Controller

The serial peripheral interface (SPI) controller of AX11025 provides a full-duplex, synchronous serial communication interface (4 wires) to flexibly work with numerous peripheral devices or micro-controller with SPI. As shown in Figure 20, the SPI controller consists of a SPI master controller with 3 slave select pins, SS0, SS1, SS2, to connect up to 3 SPI devices, and a SPI slave controller to support communication with external micro-controller with SPI master. The driving strength of SCLK, MISO, MOSI, SS1, and SS2 is programmable (4mA or 8mA, by SPI_ODS bit in I2C Configuration EEPROM offset 0x04, see section 3.1.4 for details).

The SPI master controller supports 4 types of interface timing mode, namely, Mode 0, Mode 1, Mode 2, and Mode 3 to allow working with most SPI devices available. Please refer to section 4.20 for detailed description of the 4 timing modes. It supports variable length of transfer word up to 32 bits per software command or even extended length of transfer word for a long burst transfer by keeping slave select pins active. It supports either MSB or LSB first data transfer, and the operating SPI clock, SCLK, is programmable by software and can be run up to 14 MHz when operating system clock is at 100MHz.

The SPI slave controller allows an external micro-controller with SPI master to communicate with AX11025. It supports 2 types of interface timing mode, namely, mode 0 and mode 3. In slave mode, only MSB first data transfer is supported and only the slave select pin, SS0, is used. The SPI slave controller supports 8 flexible command instructions for the external micro-controller to access the internal registers and memory resources of AX11025. It contains a 16-bytes FIFO to hold receive/transmit data on SPI interface and the SPI clock can be run up to 6 MHz when operating system clock is at 100MHz. For more details, please refer to section 4.20.

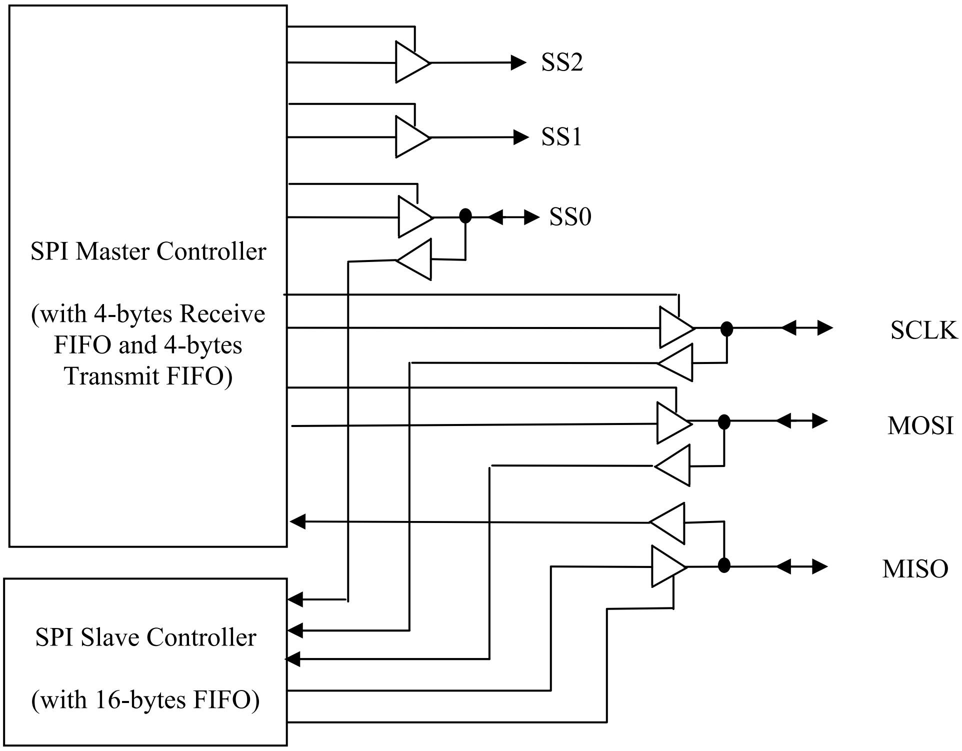


Figure 20: SPI Controller Block Diagram

2.21 CAN Controller

The Controller Area Network (CAN) controller of AX11025 supports a serial, asynchronous, multi-master communication bus protocol for connecting microcontrollers in automotive and general industrial automation applications. The CAN is a two-wire, half duplex, high-speed network system and is well suited for high speed applications using short messages. Its robustness, reliability and the large following from the semiconductor industry are some of the benefits with CAN.

The CAN bus is a broadcast type of bus. That means all nodes can "hear" all transmissions. There is no way to send a message to just a specific node, all nodes will invariably pick up all traffic. The CAN controller hardware, however, provides local filtering so that each node may react only on the interesting messages. The bus uses Non-Return-To-Zero (NRZ) with bit stuffing. The modules are connected to the CAN bus in Wire-AND fashion; “Recessive” bits (mostly logic level “1”) are overwritten by “Dominant” bits (mostly logic level “0”).

The data messages transmitted from any node on a CAN bus do not contain addresses of either the transmitting node, or of any intended receiving node. Instead, the content of the message is labeled by an identifier that is unique throughout the network. The unique identifier also determines the priority of the message. The lower the numerical value of the identifier, the higher the priority. This allows arbitration if two (or more) nodes compete for access to the bus at the same time. The higher priority message is guaranteed to gain bus access as if it were the only message being transmitted. Lower priority messages are automatically re-transmitted in the next bus cycle, or in a subsequent bus cycle if there are still other higher priority messages waiting to be sent.

As shown in Figure 21, the CAN controller of AX11025 uses two signal pins (CANRX and CANTX) to interface to external CAN transceiver. These pins can be enabled by SPI_PSEL bits in I2C Configuration EEPROM offset 0x03, see section 3.1.3 for details. The CANTX output driving strength is either 4 or 8mA, set by SPI_ODS bit in I2C Configuration EEPROM offset 0x04.

The main features of CAN controller are listed below,

- CAN 2.0B protocol compatible
- Support Standard frame (11-bits Identifier field) and Extended frame (29-bits Identifier field) format messages
- Supports two modes of operation: Basic mode (for Standard frame) and Extended mode (for Standard and Extended frames)
- Support 4 message types, Data frame, Remote frame, Error frame and Overload frame, as shown in Figure 117 ~ Figure 121. Data frame and Remote frame are processed by software; Error frame and Overload frame are processed by hardware
- Contain 64-bytes FIFO as receive buffer. When receive buffer is almost full (less than 23-bytes space left), support flow control in hardware by automatically sending out Overload frame to other nodes right after receiving Data frame. This can temporarily stop other nodes from sending more messages to avoid receive buffer overflow.
- Support 10-bytes transmit buffer in Basic mode or 13-bytes transmit buffer in Extended mode
- Data length from 0 to 8 bytes
- Programmable baud rate generator up to 1 Mbps to handle bit timing on CAN bus. Synchronized to the bit stream on the CAN-bus on a recessive to dominant edge during bus idle indicating Start-of-Frame, and resynchronized on further transitions during the reception of a message. Provides programmable time segment to compensate for the propagation delay times and phase shift and to define the sample point and the number of samples to be taken within a bit time
- Support acceptance filter to filter unwanted CAN message frames in hardware
- Support error detection, error signaling, and fault confinement in hardware
- Automatic retransmission of corrupted message as soon as the bus is idle again
- Extended mode extensions:
    - Arbitration lost interrupts with report of detailed bit position

- Error counters with read/write access and programmable error warning limit
- Error interrupts with report of CAN bus error types
- Listen only mode (no acknowledge, no active error flags)
- Acceptance filter extension
- Number of available messages within the RX FIFO

For more details, please refer to section 4.21.

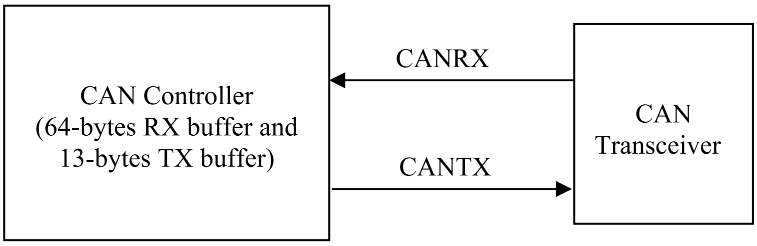


Figure 21: CAN Controller Block Diagram

3.0 Memory Map Description

3.1 I2C Configuration EEPROM Memory Map

The I2C Configuration EEPROM uses a serial EEPROM with I2C interface with at least 128x8 (1K bits), for example, Atmel AT24C01. The 7-bit device address of the I2C Configuration EEPROM should be 1010000b for AX11025. The I2C Configuration EEPROM is used to store some hardware and software default setting for the chip. These setting will be loaded into the chip by the I2C master controller right after the deactivation of chip reset or through software issuing reload command in I2C controller. Table 3 below shows the memory map of I2C Configuration EEPROM. Note that if I2C EEPROM is not used, then I2C_BOOT_DIS pin should be pulled up during chip reset, and the reset value of each offset address listed in this section shall be used by the AX11025 by default.

<table>
<tr><th>EEPROM Offset</th><th colspan="8">Descriptor</th></tr>
<tr><td>0x00</td><td colspan="8">Length</td></tr>
<tr><td>0x01</td><td colspan="8">Flag</td></tr>
<tr><td>0x03~0x02</td><td colspan="4">Multi-function Pin Setting 1</td><td colspan="4">Multi-function Pin Setting 0</td></tr>
<tr><td>0x04</td><td colspan="8">Programmable Output Driving Strength</td></tr>
<tr><td>0x05</td><td colspan="8">Reserved = 0x00</td></tr>
<tr><td>0x0B~0x06</td><td>Node ID 5</td><td colspan="2">Node ID 4</td><td>Node ID 3</td><td>Node ID 2</td><td colspan="2">Node ID 1</td><td>Node ID 0 (0x06)</td></tr>
<tr><td>0x0D~0x0C</td><td colspan="4">Maximum Packet Size 1</td><td colspan="4">Maximum Packet Size 0 (0x0C)</td></tr>
<tr><td>0x0F~0x0E</td><td colspan="4">Secondary PHY Type and PHY ID</td><td colspan="4">Primary PHY Type and PHY ID (0x0E)</td></tr>
<tr><td>0x11~0x10</td><td colspan="4">Pause Frame Low Water Mark</td><td colspan="4">Pause Frame High Water Mark (0x10)</td></tr>
<tr><td>0x13~0x12</td><td colspan="4">Reserved = 0x00</td><td colspan="4">Reserved = 0x87</td></tr>
<tr><td>0x15~0x14</td><td colspan="4">TOE TX VLAN Tag 1</td><td colspan="4">TOE TX VLAN Tag 0 (0x14)</td></tr>
<tr><td>0x17~0x16</td><td colspan="4">TOE RX VLAN Tag 1</td><td colspan="4">TOE RX VLAN Tag 0 (0x16)</td></tr>
<tr><td>0x18</td><td colspan="8">TOE ARP Cache Timeout</td></tr>
<tr><td>0x1C~0x19</td><td colspan="2">TOE Source IP Address 3</td><td colspan="2">TOE Source IP Address 2</td><td colspan="2">TOE Source IP Address 1</td><td colspan="2">TOE Source IP Address 0 (0x19)</td></tr>
<tr><td>0x20~0x1D</td><td colspan="2">TOE Subnet Mask 3</td><td colspan="2">TOE Subnet Mask 2</td><td colspan="2">TOE Subnet Mask 1</td><td colspan="2">TOE Subnet Mask 0 (0x1D)</td></tr>
<tr><td>0x21</td><td colspan="8">TOE L4 DMA Transfer Gap</td></tr>
<tr><td>0x2F~0x22</td><td colspan="8">Reserved for Hardware future use</td></tr>
<tr><td>0x7F~0x30</td><td colspan="8">Reserved for Software and Driver Settings</td></tr>
</table>

Table 3: I2C Configuration EEPROM Memory Map

3.1.1 Length (0x00)

This field determines the number of bytes (not including the length byte itself) to be loaded by the I2C master controller from I2C Configuration EEPROM after chip reset. Please set to 0x21. Note that setting any value larger than 0x2F will be changed to 0x2F by I2C master controller, i.e., it will only load the content between 0x00~0x2F for the hardware use in that case.

3.1.2 Flag (0x01)

Bit	7	6	5	4	3	2	1	0
Name	DBG_PSEL	ASCS			ACB	RCB	TOFFOP	F10HD
Reset Value	1	111			1	1	0	0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>F10HD</td><td>Force embedded Ethernet PHY to operate at 10M Half-Duplex mode.
1: Force the embedded Ethernet PHY to operate in 10Mbps half-duplex mode.
0: The embedded Ethernet PHY will base on auto negotiation to determine mode of operation.</td></tr>
<tr><td>1</td><td>TOFFOP</td><td>Turn OFF 25Mhz Oscillator and PLL circuit during STOP mode
1: To turn off the 25Mhz oscillator and PLL circuit to reduce power consumption during Stop mode.
0: To keep 25Mhz Oscillator and PLL circuit free run during Stop mode.</td></tr>
<tr><td>2</td><td>RCB</td><td>Remove CRC Bytes of RX Ethernet packet.
1: Ethernet MAC removes the CRC bytes of received Ethernet packet before forwarding to CPU.
0: The CRC bytes are not removed.</td></tr>
<tr><td>3</td><td>ACB</td><td>Append CRC Bytes of TX Ethernet packet.
1: The CRC byte of transmitted Ethernet packet are generated and appended by the Ethernet MAC.
0: The CRC bytes are not appended.</td></tr>
<tr><td>6:4</td><td>ASCS</td><td>Addressable Space of external SRAM Chip Select pin, “XRAMCE0_N”.
<table>
<tr><th>Value</th><th>Addressable Space</th></tr>
<tr><td>000</td><td>0~64 Kbytes</td></tr>
<tr><td>001</td><td>0~128 Kbytes</td></tr>
<tr><td>010</td><td>0~256 Kbytes</td></tr>
<tr><td>011</td><td>0~512 Kbytes</td></tr>
<tr><td>100</td><td>0~1024 Kbytes</td></tr>
<tr><td>101~110</td><td>Reserved</td></tr>
<tr><td>111</td><td>0~16384 Kbytes</td></tr>
</table></td></tr>
<tr><td>7</td><td>DBG_PSEL</td><td>CPU Debugger Pin Select. This selects the desired function (CPU debugger pins or Ethernet LED pins) of below multi-function pins, which users want to enable.
<table>
<tr><th>Pin #</th><th>DBG_PSEL = 0</th><th>DBG_PSEL = 1</th></tr>
<tr><td>30</td><td>DB_DI</td><td>LNK_LED</td></tr>
<tr><td>31</td><td>DB_CKO</td><td>SPD_LED</td></tr>
<tr><td>32</td><td>DB_DO</td><td>FD_CL_LED</td></tr>
</table></td></tr>
</table>

3.1.3 Multi-function Pin Setting (0x02~0x03)

Multi-function Pin Setting 0 (0x02)

Bit	7	6	5	4	3	2	1	0
Name	P3_PSEL		P2_PSEL		P1_PSEL		P0_PSEL	
Reset Value	00		00		00		00	

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>1:0</td><td>P0_PSEL</td><td>GPIO Port 0 Pin Select. This selects the desired function (port 0 or UART2) of below multi-function pins, which users want to enable.
<table>
<tr><th>Pin #</th><th>P0_PSEL = 00/01</th><th>P0_PSEL = 10</th><th>P0_PSEL = 11</th></tr>
<tr><td>18</td><td>P00</td><td>Reserved</td><td>RXD2</td></tr>
<tr><td>20</td><td>P01</td><td>Reserved</td><td>TXD2</td></tr>
<tr><td>22</td><td>P02</td><td>Reserved</td><td>CTS</td></tr>
<tr><td>23</td><td>P03</td><td>Reserved</td><td>DSR</td></tr>
<tr><td>25</td><td>P04</td><td>Reserved</td><td>RI</td></tr>
<tr><td>28</td><td>P05</td><td>Reserved</td><td>DCD</td></tr>
<tr><td>34</td><td>P06</td><td>Reserved</td><td>RTS</td></tr>
<tr><td>35</td><td>P07</td><td>Reserved</td><td>DTR</td></tr>
</table></td></tr>
<tr><td>3:2</td><td>P1_PSEL</td><td>GPIO Port 1 Pin Select. This selects the desired function (port 1 or MII) of below multi-function pins, which users want to enable.
<table>
<tr><th>Pin #</th><th>P1_PSEL = 00/01</th><th>P1_PSEL = 10</th><th>P1_PSEL = 11</th></tr>
<tr><td>77</td><td>P10</td><td>Reserved</td><td>MDC</td></tr>
<tr><td>79</td><td>P11</td><td>Reserved</td><td>MDIO</td></tr>
<tr><td>81</td><td>P12</td><td>Reserved</td><td>DE</td></tr>
<tr><td>85</td><td>P13</td><td>Reserved</td><td>RE_N</td></tr>
<tr><td>87</td><td>P14</td><td>Reserved</td><td>P14</td></tr>
<tr><td>91</td><td>P15</td><td>Reserved</td><td>P15</td></tr>
<tr><td>94</td><td>P16</td><td>Reserved</td><td>P16</td></tr>
<tr><td>95</td><td>P17</td><td>Reserved</td><td>P17</td></tr>
</table></td></tr>
<tr><td>5:4</td><td>P2_PSEL</td><td>GPIO Port 2 Pin Select. This selects the desired function (port 2 or MII) of below multi-function pins, which users want to enable.
<table>
<tr><th>Pin #</th><th>P2_PSEL = 00/01</th><th>P2_PSEL = 10</th><th>P2_PSEL = 11</th></tr>
<tr><td>5</td><td>P20</td><td>Reserved</td><td>RX_CLK</td></tr>
<tr><td>9</td><td>P21</td><td>Reserved</td><td>MRXD0</td></tr>
<tr><td>12</td><td>P22</td><td>Reserved</td><td>MRXD1</td></tr>
<tr><td>14</td><td>P23</td><td>Reserved</td><td>MRXD2</td></tr>
<tr><td>17</td><td>P24</td><td>Reserved</td><td>MRXD3</td></tr>
<tr><td>19</td><td>P25</td><td>Reserved</td><td>RX_DV</td></tr>
<tr><td>21</td><td>P26</td><td>Reserved</td><td>CRS</td></tr>
<tr><td>27</td><td>P27</td><td>Reserved</td><td>RX_ER</td></tr>
</table></td></tr>
<tr><td>7:6</td><td>P3_PSEL</td><td>GPIO Port 3 Pin Select. This selects the desired function (port 3 or MII) of below multi-function pins, which users want to enable.
<table>
<tr><th>Pin #</th><th>P3_PSEL = 00/01</th><th>P3_PSEL = 10</th><th>P3_PSEL = 11</th></tr>
<tr><td>78</td><td>P30</td><td>Reserved</td><td>TX_CLK</td></tr>
<tr><td>80</td><td>P31</td><td>Reserved</td><td>MTXD0</td></tr>
<tr><td>82</td><td>P32</td><td>Reserved</td><td>MTXD1</td></tr>
<tr><td>86</td><td>P33</td><td>Reserved</td><td>MTXD2</td></tr>
<tr><td>88</td><td>P34</td><td>Reserved</td><td>MTXD3</td></tr>
<tr><td>89</td><td>P35</td><td>Reserved</td><td>TX_EN</td></tr>
<tr><td>92</td><td>P36</td><td>Reserved</td><td>TX_ER</td></tr>
<tr><td>93</td><td>P37</td><td>Reserved</td><td>COL</td></tr>
</table></td></tr>
</table>

Multi-function Pin Setting 1 (0x03)

Bit	7	6	5	4	3	2	1	0
Name	SPI_PSEL		TM_PSEL		UIT_PSEL		Reserved	
Reset Value	00		00		00		00	

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>1:0</td><td>Reserved</td><td>Please set to “00” for normal operation.</td></tr>
<tr><td>3:2</td><td>UIT_PSEL</td><td>UART1, INT1, Timer0 Pin Select. This selects the desired function (UART1/ INT1/Timer0 or PCA signals) of below multi-function pins, which users want to enable.
<table>
<tr><th>Pin #</th><th>UIT_PSEL = 00/01</th><th>UIT_PSEL = 10</th><th>UIT_PSEL = 11</th></tr>
<tr><td>107</td><td>RXD1</td><td>Reserved</td><td>ECI</td></tr>
<tr><td>110</td><td>TXD1</td><td>Reserved</td><td>CEX0</td></tr>
<tr><td>116</td><td>INT1</td><td>Reserved</td><td>INT1</td></tr>
<tr><td>8</td><td>TM0_CK</td><td>Reserved</td><td>TM0_CK</td></tr>
<tr><td>10</td><td>TM0_GT</td><td>Reserved</td><td>TM0_GT</td></tr>
</table></td></tr>
<tr><td>5:4</td><td>TM_PSEL</td><td>Timer1, Timer2 Pin Select. This selects the desired function (Timer1/2 or PCA signals) of below multi-function pins, which users want to enable.
<table>
<tr><th>Pin #</th><th>TM_PSEL = 00/01</th><th>TM_PSEL = 10</th><th>TM_PSEL = 11</th></tr>
<tr><td>11</td><td>TM1_CK</td><td>Reserved</td><td>CEX1</td></tr>
<tr><td>13</td><td>TM1_GT</td><td>Reserved</td><td>CEX2</td></tr>
<tr><td>15</td><td>TM2_CK</td><td>Reserved</td><td>CEX3</td></tr>
<tr><td>16</td><td>TM2_GT</td><td>Reserved</td><td>CEX4</td></tr>
</table></td></tr>
<tr><td>7:6</td><td>SPI_PSEL</td><td>SPI Pin Select. This selects the desired function (SPI, 1-Wire or CAN) of below multi-function pins, which users want to enable.
<table>
<tr><th>Pin #</th><th>SPI_PSEL = 00/01</th><th>SPI_PSEL = 10</th><th>SPI_PSEL = 11</th></tr>
<tr><td>4</td><td>SS1</td><td>DQ</td><td>CANRX</td></tr>
<tr><td>6</td><td>SS2</td><td>STPZ</td><td>CANTX</td></tr>
</table></td></tr>
</table>

3.1.4 Programmable Output Driving Strength (0x04)

Bit	7	6	5	4	3	2	1	0
Name	PCA_ODS	SPI_ODS	I2C_ODS	MI_ODS	P3_ODS	P2_ODS	P1_ODS	P0_ODS
Reset Value	0	0	1	1	0	0	0	0

Bit	**Name**	**Description**
0	P0_ODS	GPIO Port 0 Output Driving Strength setting. Note that this setting is independent of P0_PSEL in offset 0x02. 1: Set driving strength to 8mA on P0 [7:0] pins (pin # 18, 20, 22, 23, 25, 28, 34, 35). 0: Set driving strength to 4mA on P0 [7:0] pins.
1	P1_ODS	GPIO Port 1 Output Driving Strength setting. Note that this setting is independent of P1_PSEL in offset 0x02. 1: Set driving strength to 8mA on P1 [7:0] pins (pin # 77, 79, 81, 85, 87, 91, 94, 95). 0: Set driving strength to 4mA on P1 [7:0] pins.
2	P2_ODS	GPIO Port 2 Output Driving Strength setting. Note that this setting is independent of P2_PSEL in offset 0x02. 1: Set driving strength to 8mA on P2 [7:0] pins (pin # 5, 9, 12, 14, 17, 19, 21, 27). 0: Set driving strength to 4mA on P2 [7:0] pins.
3	P3_ODS	GPIO Port 3 Output Driving Strength setting. Note that this setting is independent of P3_PSEL in offset 0x02. 1: Set driving strength to 8mA on P3 [7:0] pins (pin # 78, 80, 82, 86, 88, 89, 92, 93). 0: Set driving strength to 4mA on P3 [7:0] pins.
4	MI_ODS	Memory Interface Output Driving Strength setting. 1: Set driving strength to 8mA on XDATA [7:0], XADDR [20:0], XPRGRD_N, XPRGWR_N, XDATRD_N, and XDATWR_N pins. 0: Set driving strength to 4mA on XDATA [7:0], XADDR [20:0], XPRGRD_N, XPRGWR_N, XDATRD_N, and XDATWR_N pins.
5	I2C_ODS	I2C Output Driving Strength setting. 1: Set driving strength to 8mA on SCL, SDA pins (pin # 113, 114). 0: Set driving strength to 4mA on SCL, SDA pins.
6	SPI_ODS	SPI Output Driving Strength setting. Note that this setting is independent of SPI_PSEL in offset 0x03. 1: Set driving strength to 8mA on SCLK, MISO, MOSI, SS1, and SS2 pins (pin # 1, 3, 2, 4, 6). 0: Set driving strength to 4mA on SCLK, MISO, MOSI, SS1, and SS2 pins.
7	PCA_ODS	PCA Output Driving Strength setting. Note that this setting is independent of UIT_PSEL and TM_PSEL in offset 0x03. 1: Set driving strength to 8mA on CEX [4:0], LINT, LALE, and LWR_N pins (pin # 16, 15, 13, 11, 110, 116, 8, 10). 0: Set driving strength to 4mA on CEX [4:0], LINT, LALE, and LWR_N pins.

3.1.5 Node ID (0x06~0x0B)

The Node ID 5 to Node ID 0 set the default MAC address of this chip. For example, if the MAC address is 01-23-45-67-89-AB, then put Node ID {5, 4, 3, 2, 1, 0} = {0x01, 0x23, 0x45, 0x67, 0x89, 0xAB}. The reset value of Node ID in this ASIC = 0x0000_0000_0000.

3.1.6 Maximum Packet Size (0x0C~0x0D)

The Maximum Packet Size 1 and Maximum Packet Size 0 set the maximum Ethernet packet size that can be received from network. If the received Ethernet packet exceeds this number, Ethernet MAC shall truncate it. Note that the Maximum Packet Size field must be even number in bytes and less than or equal to 2500 bytes. For example, if maximum packet size is 1522 bytes, then put Maximum Packet Size {1,0} = {0x05, 0xF2}. The reset value of Maximum Packet Size 1 and 0 in this ASIC = 0x05F2.

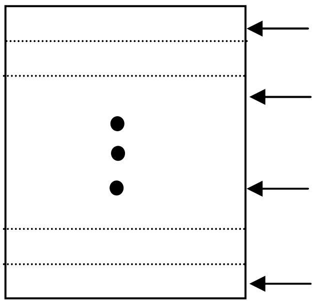
3.1.7 Primary/Secondary PHY Type and PHY ID (0x0E~0x0F)

Bit	7	6	5	4	3	2	1	0
Name	PHY Type			PHY ID				
Reset Value	000 (primary) 111 (secondary)			1_0000 (primary) 0_0000 (secondary)				

Bit	Name	Description
4:0	PHY ID	The PHY ID of PHY. Primary PHY ID: Set to 1_0000 for the embedded Ethernet PHY. Secondary PHY ID: Set to other value than 1_0000 or set to 0_0000 if not used.
7:5	PHY Type	PHY Type is defined as follows, 000: IEEE 802.3 10BASE-T/100BAS-TX Ethernet PHY. 001: HomePlug PHY. 100: Special case where the link status of PHY is always reported as link-up. 111: Non-supported PHY. For example, setting “0xE0” in Secondary PHY Type and PHY ID field means that the secondary PHY is not supported.

3.1.8 Pause Frame Low Water and High Water Mark (0x10~0x11)

When operating in full-duplex mode, correct setting of this field is very important and can affect the overall packet receive throughput performance in a great deal. The Low Water Mark is the threshold to trigger sending of Pause frame and the High Water Mark is threshold to stop sending of Pause frame. Note that each free buffer count here represents 256 bytes of packet storage space in RX packet buffer SRAM in Ethernet MAC. For now, set Pause Frame Low Water to 0x19 and Pause Frame High Water Mark to 0x1d.



Total free buffer count = 32

Stop sending Pause frame when free buffer > High Water Mark (reset value in this ASIC = 0x1d)

Start sending Pause frame when free buffer < Low Water Mark (reset value in this ASIC = 0x19)

0

3.1.9 TOE TX VLAN Tag (0x14~0x15)

This field sets the default value of TOE TX VLAN Tag Register. The reset value in this ASIC = 0x0000.

3.1.10 TOE RX VLAN Tag (0x16~0x17)

This field sets the default value of TOE RX VLAN Tag Register. The reset value in this ASIC = 0x0000.

3.1.11 TOE ARP Cache Timeout (0x18)

This field sets the default value of TOE ARP Cache Timeout Register. The reset value in this ASIC = 0x00.

3.1.12 TOE Source IP Address (0x19~0x1C)

This field sets the default value of TOE Source IP Address Register. The reset value in this ASIC = 0x0000_0000.

3.1.13 TOE Subnet Mask (0x1D~0x20)

This field sets the default value of TOE Subnet Mask Register. The reset value in this ASIC = 0x0000_0000.

3.1.14 TOE L4 DMA Transfer Gap (0x21)

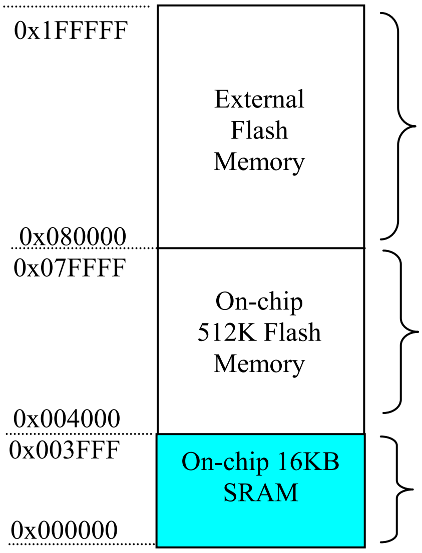
This field sets the default value of TOE L4 DMA Transfer Gap Register. The reset value in this ASIC = 0x00.

3.2 Program Memory Map

The 1T 80390 CPU core has separated address spaces for program and data memory. The Program Memory, Internal Data Memory, External Data Memory, SFRs areas each has its own address spaces. As shown in below two figures, the CPU core can address up to 2 MB of linear program space without bank select. The CPU core starts execution of program code at location 0x000000 in LARGE mode, after each reset. The CPU core can be then switched to FLAT mode to support 2 M bytes of linear program code space. The case I shows the program code shadow mode is not enabled while the case II shows the program code shadow mode is enabled.

Case I: EXT_PROG_SRAM_EN is pulled low during chip hardware reset, the Program non-Shadow mode

CPU Program Memory Address



The program code residing on external Flash memory will be fetched with wait states defined in SFR register, WTST [2:0].

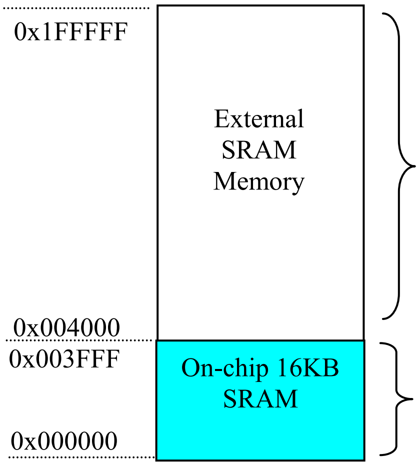
The program code residing on on-chip Flash memory space (0x004000~0x07FFFF) will be fetched with wait states defined in SFR register, WTST [2:0].

The program code residing on the on-chip 16K bytes SRAM is copied from the on-chip Flash memory space (0x000000~0x003FFF) by the Boot Loader hardware before CPU starts running, so-called “program code mirroring”. When CPU executing the program code in this range, it will always fetch from the on-chip SRAM and run at zero wait state (1T). This part of the code is used for BOOT code with system initialization

Figure 22: The Program Memory Map of 1T 80390 CPU Core (Program non-Shadow mode)

Case II: EXT_PROG_SRAM_EN is pulled high during chip hardware reset, the Program Shadow mode.

CPU Program Memory Address



The program code residing on on-chip Flash memory space (0x0004000~0x07FFFF) and external Flash memory space (0x080000~0x1FFFFF) will be copied to the external SRAM memory by the Boot Loader hardware before CPU starts running, the so-called “program code shadow”. When CPU executing the program code in this range, it will fetch from the external SRAM memory with wait states defined in SFR register, WTST [2:0].

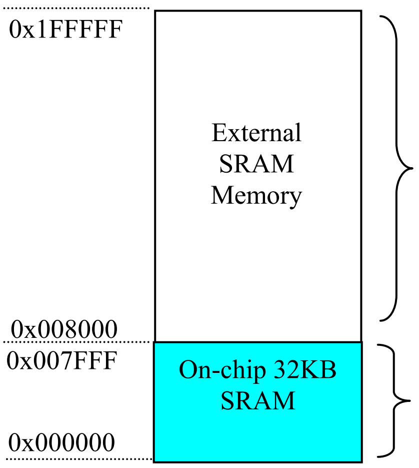
The program code residing on the on-chip 16K bytes SRAM is copied from the on-chip Flash memory space (0x000000~0x003FFF) by the Boot Loader hardware before CPU starts running, the so-called “program code mirroring”. When CPU executing the program code in this range, it will always fetch from the on-chip SRAM and run at zero wait state (1T). This part of the code is used for BOOT code with system initialization

Figure 23: The Program Memory Map of 1T 80390 CPU Core (Program Shadow mode)

3.3 External Data (xDATA) Memory Map

The data memory of 1T 80390 CPU core is divided onto 2 M bytes of External Data Memory (on-chip SRAM used 0~32K bytes memory space, off-chip SRAM used 32K~2M bytes memory space) and 256 bytes of Internal Data Memory, plus a 128-bytes of SFR memory area. As shown in below figure, the CPU core can address up to 2M bytes of External Data (xDATA) memory space without bank select. The xDATA memory is accessed by MOVX instructions only.

CPU External Data Memory Address



When CPU accessing xDATA memory space (0x0008000~0x1FFFFF), it will access with wait states defined in SFR register, CKCON [2:0].

The on-chip 32K bytes SRAM occupies the External Data Memory address space (0x000000~0x007FFF). When CPU accessing this range, it will access with wait states defined in SFR register, CKCON [2:0].

Figure 24: The External Data Memory Map of 1T 80390 CPU Core

3.4 Internal Data Memory and SFR Register Map

The Figure 25 below shows the Internal Data Memory (256 bytes) and Special Function Register (SFR) map of 1T 80390 CPU core. The lower internal memory consists of four register banks with eight registers each; a bit addressable segment with 128 bits (16 bytes) begins at 0x20, and a scratch pad area with 208 bytes.

With the indirect addressing mode, range 0x80 to 0xFF of the highest 128 bytes of the internal memory is addressed. With the direct addressing mode, range 0x80 to 0xFF, the SFR memory area is accessed.

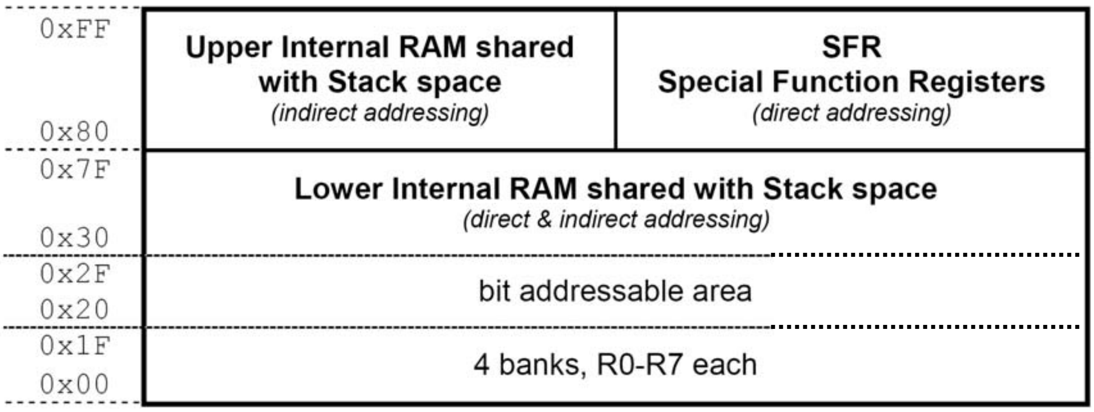

0xFF
0x80
0x7F
0x30
0x2F
0x20
0x1F
0x00
Upper Internal RAM shared with Stack space
(indirect addressing)
SFR
Special Function Registers
(direct addressing)
Lower Internal RAM shared with Stack space
(direct & indirect addressing)
bit addressable area
4 banks, R0-R7 each

Figure 25: The Internal Memory Map of 1T 80390 CPU Core

The Table 4 below shows the SFR Register Map, note that all registers in the column with Offset+0 are bit addressable.

SFR Offset	Offset+0	Offset+1	Offset+2	Offset+3	Offset+4	Offset+5	Offset+6	Offset+7
0xF8	**EIP**							
0xF0	**B**							
0xE8	**EIE**	**STATUS**	**MXAX**	**TA**				
0xE0	**ACC**	*HS_RTD*	*HS_ID*	*HS_IF*	*HS_LCR*	*HS_MCR*	*HS_LSR*	*HS_MSR*
0xD8	**WDCON**						*CANCIR*	*CANDR*
0xD0	**PSW**	*CCAPM0*	*CCAPM1*	*CCAPM2*	*CCAPM3*	*CCAPM4*	*OWCIR*	*OWDR*
0xC8	**T2CON**		**RLDL**	**RLDH**	**TL2**	**TH2**	*SPICIR*	*SPIDR*
0xC0	**SCON1**	**SBUF1**	*CMOD*	*CCON*	*CL*	*CH*		
0xB8	**IP**	*CCAP0H*	*CCAP1H*	*CCAP2H*	*CCAP3H*	*CCAP4H*	*EPCR*	*EPDR*
0xB0	**P3**	*CCAP0L*	*CCAP1L*	*CCAP2L*	*CCAP3L*	*CCAP4L*	*MCIR*	*MDR*
0xA8	**IE**	Reserved	Reserved	Reserved	Reserved	Reserved	*TCIR*	*TDR*
0xA0	**P2**	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
0x98	**SCON0**	**SBUF0**	*DBAR*	*DCIR*	*DDR*	**ACON**	*PISS1R*	*PISS2R*
0x90	**P1**	**EIF**	**WTST**	**DPX0**	*SDSTSR*	**DPX1**	*I2CCIR*	*I2CDR*
0x88	**TCON**	**TMOD**	**TL0**	**TL1**	**TH0**	**TH1**	**CKCON**	*CSREPR*
0x80	**P0**	**SP**	**DPL0**	**DPH0**	**DPL1**	**DPH1**	**DPS**	**PCON**

Bolded – are 1T-80390 CPU core related registers.
Italic – are peripheral functions, such as UART2, SPI, 1-Wire, PCA, Ethernet PHY, Ethernet MAC, TOE, I2C, CAN, and software DMA related.
Empty – are read-only.

Table 4: The SFR Register Map

4.0 Detailed Function Description

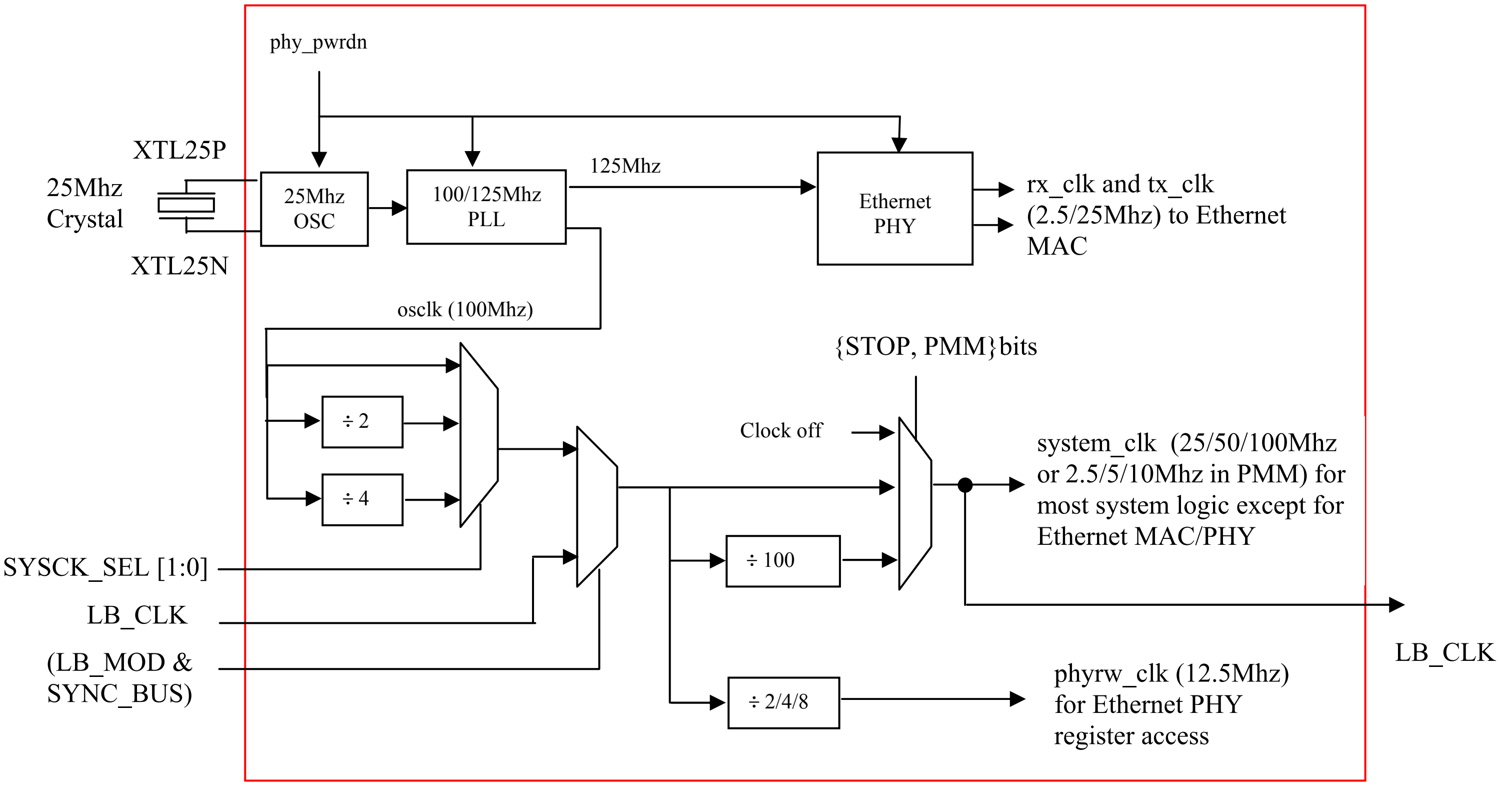
4.1 Clock Generation

The figure below shows the clock generation block of AX11025. The embedded PLL block generates the "osclk" (100Mhz) as the main clock source for system logic and 125Mhz for Ethernet PHY use. The internal "phy_pwrdn" signal is used to disable the oscillator, PLL, and Ethernet PHY for maximum power saving. After power-on reset, the "phy_pwrdn" signal is default to '0' to allow clock to oscillate initially. When entering the STOP mode by setting STOP bit (PCON.1), while the TOFFOP bit (Flag.1) in I2C EEPROM is 1, the "phy_pwrdn" signal is asserted to '1' to completely turn off oscillator, PLL, and Ethernet PHY.

During this deep power-down mode, to re-enable the oscillator/PLL/system clock, detecting a rising edge on EXT_WKUP pin or detecting a falling-edge transition (start bit) on RXD2 pin or falling-edge transition on RI pin of UART 2 will trigger the "phy_pwrdn" signal to change to "0" and then re-enable the oscillator/PLL back to free run mode. During this process, the internal "system_clk" will be gated until oscillator/PLL clock are stabled enough before it is fed to the system logic.

When internal PLL's "osclk" is selected as clock source, the SYSCK_SEL[1:0] decides the operating system clock frequency, where "00" = 25Mhz, "01" = 50Mhz, "11" = 100Mhz.

The LB_CLK input pin can be another clock source for the purpose of more accurate baud rate generation for UART0/1/2. In that case, input clock frequency of LB_CLK should be as close to 25/50/100Mhz as possible so that the internal timing function such as PCA and the dedicated Millisecond Timer can still function closely. For example, the possible LB_CLK can be 48Mhz, in that case, the SYSCK_SEL[1:0] should be set to "01" too.



Note: In above figure, lower case signal names represent chip internal signals.

Figure 26: AX11025 Clock Generation Block Diagram

4.2 Reset Generation

The Figure 27 below shows the reset generation block of AX11025. The output of power-on-reset generates a reset pulse to reset on-chip Flash memory during power-on. The internal "system_rst_n" signal is used to reset most system logic and its source can come from RST_N pin, power on reset condition, or software reset and reboot command via SFR register CSREPR. The SYS_RSTO_N is an optional reset output pin to be used by peripheral chips. Note: in below figure, the lower case signal names represent chip internal signals.

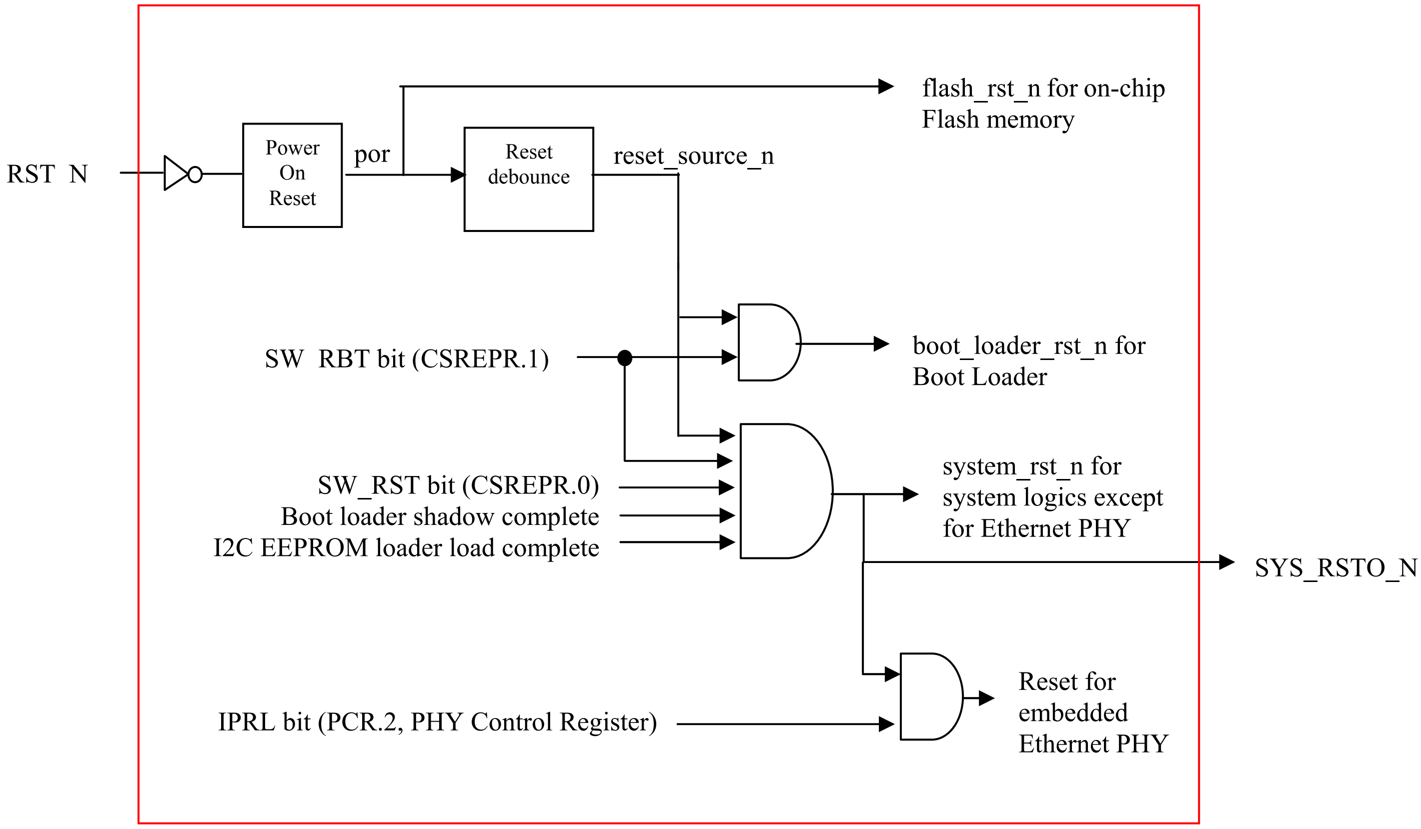

RST N
Power On Reset
por
Reset debounce
reset_source_n
flash_rst_n for on-chip Flash memory
SW RBT bit (CSREPR.1)
boot_loader_rst_n for Boot Loader
SW_RST bit (CSREPR.0)
Boot loader shadow complete
I2C EEPROM loader load complete
system_rst_n for system logics except for Ethernet PHY
SYS_RSTO_N
IPRL bit (PCR.2, PHY Control Register)
Reset for embedded Ethernet PHY

Figure 27: AX11025 Reset Generation Block Diagram

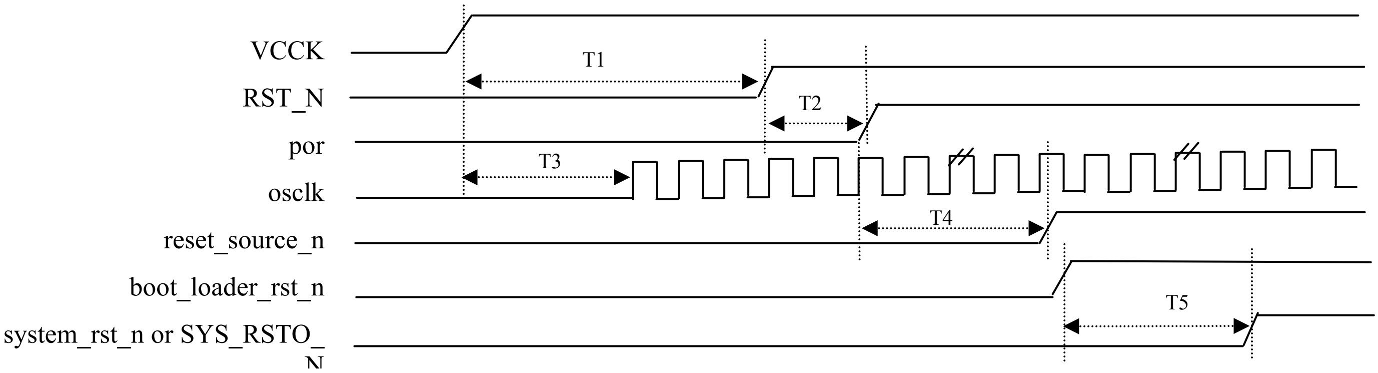


Figure 28: AX11025 Reset Timing Diagram

Symbol	Description	Min	Typ	Max	Unit
T1	RST_N asserting low interval after VCCK ramping up to 1.8V	4			ms
T2	The internal "por" signal asserting low interval after RST_N de-assertion	2.2	3.0	4.2	us
T3	From VCCK rise to 1.8V to first osclk transition		1.2		ms
T4	From internal "por" signal de-assertion to de-assertion of internal debounced "reset_source_n" signal.		100		clocks
T5	From internal "boot_loader_rst_n" signal de-assertion to internal "system_rst_n" signal or SYS_RSTO_N de-assertion: Internally, this time depends on the program code size which Boot Loader needs to copy to on-chip 16KB SRAM and/or off-chip SRAM, i.e., the bigger the code size, the longer the time it takes before de-asserting "system_rst_n" or SYS_RSTO_N signals.				

4.3 Voltage Regulator

Please refer to section 5.1.6.

4.4 CPU Core and Debugger

The 1T80390 CPU core block diagram is shown within the red line in Figure 29 below.

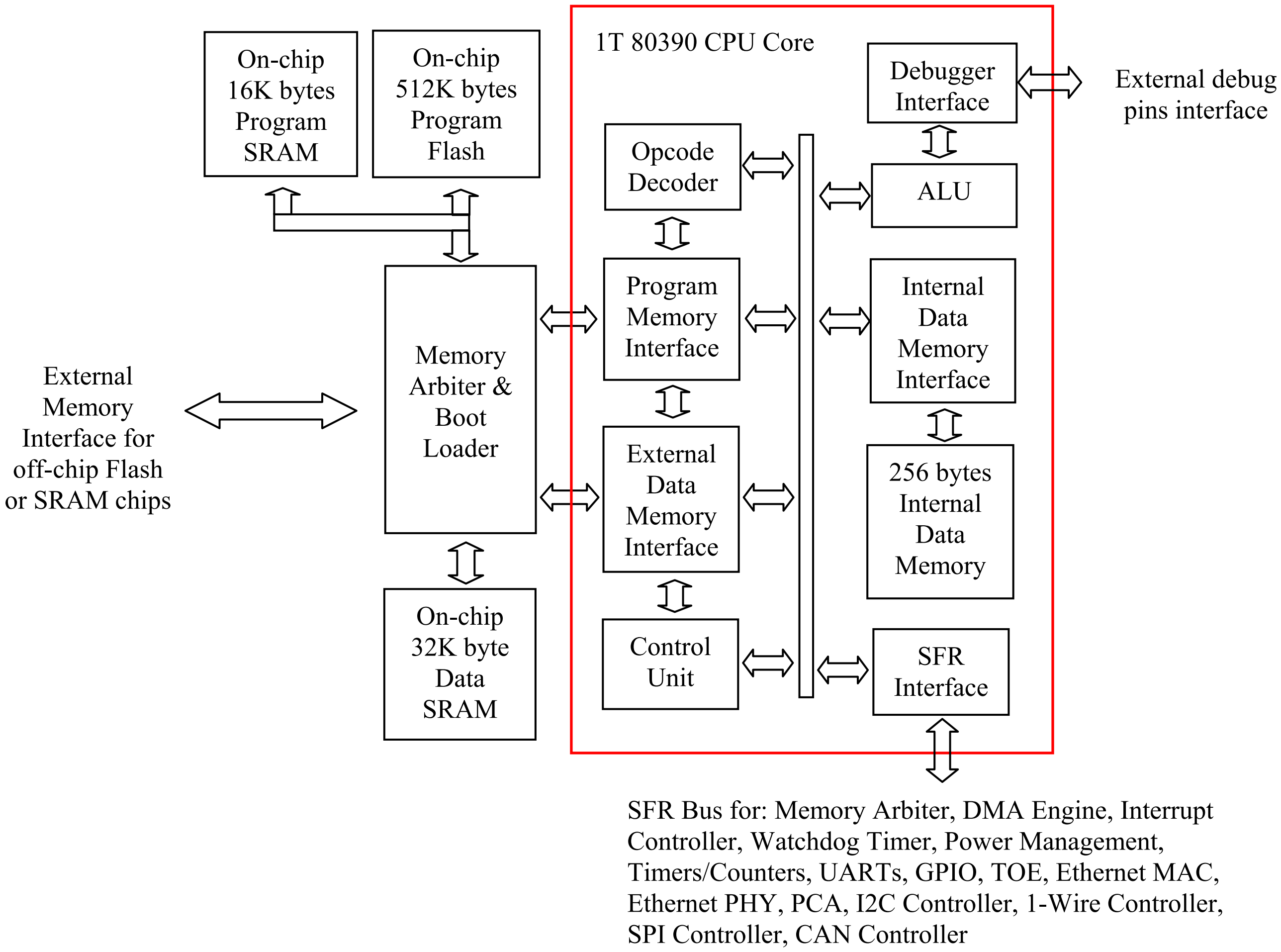

1T 80390 CPU Core
On-chip 16K bytes Program SRAM
On-chip 512K bytes Program Flash
Memory Arbiter & Boot Loader
On-chip 32K byte Data SRAM
External Memory Interface for off-chip Flash or SRAM chips
Opcode Decoder
Program Memory Interface
External Data Memory Interface
Control Unit
Debugger Interface
External debug pins interface
ALU
Internal Data Memory Interface
256 bytes Internal Data Memory
SFR Interface
SFR Bus for: Memory Arbiter, DMA Engine, Interrupt Controller, Watchdog Timer, Power Management, Timers/Counters, UARTs, GPIO, TOE, Ethernet MAC, Ethernet PHY, PCA, I2C Controller, 1-Wire Controller, SPI Controller, CAN Controller

Figure 29: CPU Core Block Diagram

ALU - Arithmetic Logic Unit performs the arithmetic and logic operations during execution of an instruction. It contains accumulator (ACC), Program Status Word (PSW), (B) registers and related logic such as arithmetic unit, logic unit, multiplier and divider.

Opcode Decoder - performs an instruction opcode decoding and the control functions for all other blocks.

Control Unit - performs the core synchronization and data flow control. This module is directly connected to Opcode Decoder and manages execution of all micro-controller tasks.

Program Memory Interface - contains Program Counter (PC) and related logic. It performs the instructions code fetching from on-chip 512K bytes Program Flash, on-chip 16K byte Program SRAM, or off-chip Program Flash/SRAM via External Memory Interface. The Program Memory can be also written.

External Data Memory Interface - contains memory access related registers such as Data Pointer High (DPH), Data Pointer Low (DPL) and Data Pointer eXtended (DPX) registers. It performs the external Program and Data Memory addressing and data transfers. The Program fetch cycle length is programmed by user.

Internal Data Memory Interface - Internal Data Memory interface controls access into the internal 256 bytes data memory. It contains 8-bit Stack Pointer (SP) register and related logic.

SFRs Interface - Special Function Registers interface controls access to the special registers. It contains standard 8051/80390 SFR registers and some additional SFR registers specific to this chip. SFR register access (read, written, modified) can use all direct addressing mode instructions.

Debugger Interface – provides an in-circuit emulator feature with 3 wires (clock out, data in, data out) interface and is used to connect to an external Hardware Assisted Debugger (HAD2) to communicate with the Debug Software running on PC.

4.4.1 CPU Core SFR Register Map

Address	Name	Description
0x81	SP	Stack Pointer register
0x82	DPL0	Data Pointer 0 register (DPTR0) low byte
0x83	DPH0	Data Pointer 0 register (DPTR0) high byte
0x84	DPL1	Data Pointer 1register (DPTR1) low byte
0x85	DPH1	Data Pointer 1register (DPTR1) high byte
0x86	DPS	Data Pointers Select register
0x87	PCON	Power Configuration register
0x8E	CKCON	Clock Control register
0x92	WTST	Program Memory Wait States register
0x93	DPX0	Data Pointer eXtended 0 register
0x95	DPX1	Data Pointer eXtended 1 register
0x9D	ACON	Address Control register
0xD0	PSW	Program Status Word register
0xE0	ACC	Accumulator A register
0xEA	MXAX	MOVX @Ri eXtended register
0xF0	B	B register

Table 5: CPU Core SFR Register Map

The following abbreviations are used in the “Access” column in all SFR register detailed description.

Access	Description
R/W	Software can read or write to the register bit.
RO	The register bit is read-only.
W1	Software can only write “1” to the register bit. Writing “0” to the register bit has no effect.
CR	The register bit will be clear after software reads it.
R/W1	Software can read or write “1” to the register bit. Writing “0” to the register bit has no effect.
WO	The register bit is write-only.
SC	Self-clearing.
PS	Value is permanently set.
LL	Latch to Low.
LH	Latch to High.

4.4.2 CPU Core SFR Register Description

The 1T 80390 CPU core is fully compatible to the standard 8051 micro-controller, maintains all instruction mnemonics and binary compatibility. The CPU core incorporates some great architectural enhancements, which allow the CPU execution of instructions with high performance.

The arithmetic section of the processor performs extensive data manipulation and is comprised of the 8-bit arithmetic logic unit (ALU), an ACC register, B register and PSW register as described below. The ALU performs typical arithmetic operations as: addition, subtraction, multiplication, division and additional operations such as: increment, decrement, BCD-decimal-add-adjust and compare. Within logic unit are performed: AND, OR, Exclusive OR, complement and rotation. The Boolean processor performs the bit operations as: set, clear, complement, jump-if-not-set, jump-if-set-and-clear and move to/from carry. The PSW contains several bits that reflect the current state of the CPU.

Accumulator A register (ACC, 0xE0)

Bit	7	6	5	4	3	2	1	0
Name	ACC							
Reset Value	0x00							

Bit	**Name**	**Access**	**Description**
7:0	ACC	R/W	The Accumulator A register.

B Register (B, 0xF0)

Bit	7	6	5	4	3	2	1	0
Name	B							
Reset Value	0x00							

Bit	**Name**	**Access**	**Description**
7:0	B	R/W	The B register is used during multiply and divide operations. In other cases may be used as normal SFR.

Program Status Word Register (PSW, 0xD0)

Bit	7	6	5	4	3	2	1	0
Name	CY	AC	F0	RS1	RS0	OV	F1	P
Reset Value	0x00							

Bit	**Name**	**Access**	**Description**
0	P	R/W	Parity flag
1	F1	R/W	General purpose flag 1
2	OV	R/W	Overflow flag
4:3	RS1, RS0	R/W	Register bank select bits **RS [1:0]** – **Function description** 00 – Bank 0, data address 0x00-0x07 01 – Bank 1, data address 0x08-0x0F 10 – Bank 2, data address 0x10-0x17 11 – Bank 3, data address 0x18-0x1F
5	F0	R/W	General purpose flag 0
6	AC	R/W	Auxiliary carry
7	CY	R/W	Carry flag

4.4.3 Memory Allocation

The 1T 80390 CPU core has separated address spaces for program and data memory. The Internal Data Memory, External Data Memory, SFRs and Program Memory areas have their own address spaces. The data memory is divided onto 2 M bytes of External Data Memory (on-chip SRAM used 0~32K bytes memory space, off-chip SRAM used 32K~2M bytes memory space) and 256 bytes of Internal Data Memory, plus a 128-bytes of SFR memory area. Please refer to section 3.2, section 3.3, and section 3.4 for memory map description.

Program Memory Allocation

The Program Memory is typically used for main code and constants. The 1T 80390 CPU core can support program memory operation in LARGE and FLAT mode. In LARGE mode, the addressable program memory space is located in 0x0000~0xFFFF (64K bytes), while in FLAT mode, the addressable program memory space is located in 0x000000~0x1FFFFF (2M bytes). After each reset, the 1T 80390 CPU core starts execution of program code at location 0x000000 in LARGE mode. The CPU core then can be switched to FLAT mode to support 2M bytes of linear program code space. The user is recommended to operate the 1T 80390 CPU core of AX11025 in FLAT mode to save the troubles of handling code banking. For program memory map description, please refer to section 3.2.

The on-chip 16K bytes Program SRAM is located in program memory space 0x000000~0x003FFF. This part of the code is usually for BOOT code with system initialization functions, TFTP or UART, and Flash programming functions. After hardware reset or software reboot via setting SW_RBT bit (CSREPR.1), the Boot Loader will always copy this part of code from the lower 16K bytes space of on-chip 512K bytes Program Flash, before CPU starts running. When the CPU core runs and accesses program memory space between 0x000000~0x003FFF, it will fetch from the on-chip 16K bytes Program SRAM. When accessing beyond 0x003FFF program memory space, it will fetch from the on-chip 512K bytes Program Flash or the external memory chips. Having a separate Program SRAM allows updating firmware on the on-chip Flash memory while the CPU core continues running, to support the so-called In Application Programming (IAP) function.

Program Memory Wait-state

The program code residing on the on-chip 16K bytes Program SRAM is always fetched and executed by the CPU core without wait state (i.e., 1T). So besides BOOT code, user can consider using program memory space 0x000000~0x003FFF for any timing-critical routines or firmware to yield better CPU performance.

The program code residing beyond 0x003FFF address space (on on-chip Flash memory and/or off-chip Flash or SRAM) may require some wait-state cycles depending on operating system clock frequency and whether or not the "program code shadow" mode is enabled. If "program code shadow" mode is not enabled, the program code is running on Flash memory, which need more wait states. If "program code shadow" mode is enabled, the program code is running on SRAM, which needs less wait states. The Program Memory Wait States (WTST) register is used to set user programmable wait state during program memory read and write access cycles.

Program Memory Wait States Register (WTST, 0x92)

Bit	7	6	5	4	3	2	1	0
Name	Reserved					WTST		
Reset Value	0x07							

<table>
<tr><th>Bit</th><th>Name</th><th>Access</th><th>Description</th></tr>
<tr><td>2:0</td><td>WTST</td><td>R/W</td><td>Wait States register holds the information about Program Memory access time. The minimal read cycle takes 1 clock period (WTST = 000) and maximal 8 clock periods (WTST = 111).
Based on operating system clock frequency, the recommended setting is as below,
<table>
<tr><th rowspan="2">System Clock</th><th colspan="2">Program Memory Wait State Setting, WTST [2:0]</th></tr>
<tr><th>Program Code Shadow Mode</th><th>Non-Shadow Mode</th></tr>
<tr><td>25Mhz</td><td>000</td><td>001</td></tr>
<tr><td>50Mhz</td><td>000</td><td>011</td></tr>
<tr><td>100Mhz</td><td>001</td><td>111</td></tr>
</table></td></tr>
<tr><td>7:3</td><td>Reserved</td><td></td><td></td></tr>
</table>

FLAT/LARGE Mode Switching

Switching between LARGE and FLAT modes is performed by appropriate writes into ACON (0x9D) register. ACON is Timed Access protected register and has built in mechanism preventing its accidental writes. To switch between modes the following instructions should be performed:

```
MOV  TA, #0xAA;
MOV  TA, #0x55;       Enable write to ACON register
MOV  ACON, #0x02;     Switch to FLAT mode
```

or

```
MOV  TA, #0xAA
MOV  TA, #0x55;       Enable write to ACON register
MOV  ACON, #0x00;      Switch to LARGE mode
```

It can be done at any time while software is running. The time elapsed between first, second, and third operation does not matter (any number of Program Wait Sates is allowed). The only correct sequence is required. Any third instruction causes protection mechanism to be turned on. This means that time protected register is opened for write only for single instruction. Reading from such register is never protected.

Address Control Register (ACON, 0x9D)

Bit	7	6	5	4	3	2	1	0
Name	Reserved						AM	Reserved
Reset Value	0x00							

Bit	Name	Access	Description
0	Reserved	R/W	
1	AM	R/W	Address Mode Control bit. This bit establishes the addressing mode for the 1T80390 CPU core. 0: 16-bit Addressing Mode – LARGE Mode. 1: 24-bit Contiguous Addressing Mode – FLAT Mode.
7:2	Reserved		

Please note that some instructions are different for FLAT and LARGE mode. There are:

LCALL, **ACALL**,
JMP, **LJMP**, **AJMP**,
MOVC, **MOVX** - DPTR related only
POP, **PUSH**, **RET**

Please refer to “AX110xx CPU Core Instruction Set User Guide” for more details.

Program Write Enable Bit

The Program Write Enable (PWE) bit (PCON.4) is used to enable/disable program memory write signal activity during MOVX instructions. When PWE bit is set to logic 1, the **MOVX @DPTR, A** instruction writes data located in accumulator register into program memory addressed by DPTR register (active DPX: DPH: DPL). The **MOVX @Rx, A** instruction writes data located in accumulator register into program memory addressed by MXAX (bits 23:16), P2 register (bits 15:8) and Rx register (bits 7:0). The bits 23:16 are always equal to 0x00 for LARGE mode (64 KB of CODE). For detailed description of program memory write access to Flash memory, please refer to section 4.6.4 and 4.6.5.

Power Configuration Register (PCON, 0x87)

Bit	7	6	5	4	3	2	1	0
Name	SMOD0	SMOD1	Reserved	PWE	RSM	SWB	STOP	PMM
Reset Value	0x00							

Bit	Name	Access	Description
0	PMM	R/W	Power Management Mode Enable bit. 1: PMM entered. 0: PMM disabled.
1	STOP	R/W	STOP mode bit. 1: STOP mode entered. 0: Disabled.
2	SWB	R/W	Switchback enable. 1: Enabled interrupts and serial ports cause switchback. PMM bit is cleared. 0: Interrupts and serial ports don't affect PMM bit.
3	RSM	R/W	Regulator Standby Mode. 1: Set the internal 3.3V to 1.8V regulator to operate at standby mode (when the 1.8V current drawn is less than 30mA) for better conversion efficiency. 0: Set the internal 3.3V to 1.8V regulator to full operating mode (when the 1.8V current drawn is more than 30mA) for better conversion efficiency.
4	PWE	R/W	Program memory Write Enable bit. 1: Enable Program Memory write access signal activity during MOVX instructions. 0: Disabled.
5	Reserved	R/W	
6	SMOD1	R/W	UART1 double baud rate bit.
7	SMOD0	R/W	UART0 double baud rate bit.

Data Memory Allocation

The 1T 80390 CPU core can address up to 2M bytes of External Data (xDATA) Memory space without bank select. The xDATA memory is accessed by MOVX instructions only. The on-chip 32K bytes SRAM is located at address space 0x000000~0x007FFF. For address space above 0x007FFF, user will need to add some external SRAM chips for that. Please refer to section 2.6 for external SRAM chip connection and section 3.3 for external data memory map description.

Data Memory Wait-state

The External Data Memory (applied to both the on-chip 32K bytes SRAM and external SRAM chips) access cycles may need some wait states, depending on operating system clock frequency and the cycle time of external SRAM chips being used. The MD bits (CKCON.2~0) register is used to set user programmable wait state during data memory read and write access cycles.

Clock Control Register (CKCON, 0x8E)

Bit	7	6	5	4	3	2	1	0
Name	WD		T2M	T1M	T0M	MD		
Reset Value	0x07							

<table>
<tr><th>Bit</th><th>Name</th><th>Access</th><th>Description</th></tr>
<tr><td>2:0</td><td>MD</td><td>R/W</td><td>This adjusts the stretch cycles of XDATWR_N and XDATRD_N signals during MOVX instruction for External Data Memory write and read access cycles. The Minimal read/write pulse length is equal to 1 clock period (MD = 000) and maximal 8 clock periods (MD = 111). The MD bits can be changed any time during program execution.
Based on operating system clock frequency and typical SRAM chip with 8ns access time, the recommended setting is as below,
<table>
<tr><th rowspan="2">System Clock</th><th colspan="2">Data Memory Wait State Setting, MD</th></tr>
<tr><th>Program Code Shadow Mode</th><th>Non-Shadow Mode</th></tr>
<tr><td>25Mhz</td><td>001</td><td>001</td></tr>
<tr><td>50Mhz</td><td>001</td><td>001</td></tr>
<tr><td>100Mhz</td><td>001</td><td>001</td></tr>
</table></td></tr>
<tr><td>3</td><td>T0M</td><td>R/W</td><td>This bit controls the division of the system clock that drives Timer 0.
1: Timer 0 uses a divide-by-4 of the system clock frequency.
0: Timer 0 uses a divide-by-12 of the system clock frequency.</td></tr>
<tr><td>4</td><td>T1M</td><td>R/W</td><td>This bit controls the division of the system clock that drives Timer 1.
1: Timer 1 uses a divide-by-4 of the system clock frequency.
0: Timer 1 uses a divide-by-12 of the system clock frequency.</td></tr>
<tr><td>5</td><td>T2M</td><td>R/W</td><td>This bit controls the division of the system clock that drives Timer 2. This bit has no effect when the timer is in baud rate generator mode.
1: Timer 2 uses a divide-by-4 of the system clock frequency.
0: Timer 2 uses a divide-by-12 of the system clock frequency.</td></tr>
<tr><td>7:6</td><td>WD</td><td>R/W</td><td>WD bits select Watchdog timer timeout period.</td></tr>
</table>

Memory Related SFR Registers

The following paragraph describes Program Memory, External Data Memory, and Internal Data Memory related SFRs of 1T 80390 CPU core and their functionality.

Data Pointer Registers

Dual data pointer registers are implemented to speed up data block copying. DPTR0 and DPTR1 are located at four SFR addresses. Active DPTR register is selected by SEL bit of Data Pointer Select (DPS) register. If SEL bit is equal to 0 then DPTR0 (0x83:0x82) is selected otherwise DPTR1 (0x85:0x84).

DPH0 (0x83)								DPL0 (0x82)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Data Pointer register 0 (DPTR0)

DPH1 (0x85)								DPL1 (0x84)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Data Pointer register 1 (DPTR1)

Selected data pointer register is used in the following instructions:

```
MOVX  @DPTR, A
MOVX  A, @DPTR
MOVC  A, @A+DPTR
JMP  @A+DPTR
INC  DPTR
MOV  DPTR, #data16/#data24
```

Data Pointer 0 Register (DPTR0) High Byte (DPH0, 0x83)

Bit	7	6	5	4	3	2	1	0
Name	DPH0							
Reset Value	0x00							

Bit	**Name**	**Access**	**Description**
7:0	DPH0	R/W	The high byte of Data Pointer 0 register.

Data Pointer 0 Register (DPTR0) Low Byte (DPL0, 0x82)

Bit	7	6	5	4	3	2	1	0
Name	DPL0							
Reset Value	0x00							

Bit	**Name**	**Access**	**Description**
7:0	DPL0	R/W	The low byte of Data Pointer 0 register.

Data Pointer 1 Register (DPTR1) High Byte (DPH1, 0x85)

Bit	7	6	5	4	3	2	1	0
Name	DPH1							
Reset Value	0x00							

Bit	**Name**	**Access**	**Description**
7:0	DPH1	R/W	The high byte of Data Pointer 1 register.

Data Pointer 1 Register (DPTR1) Low Byte (DPL1, 0x84)

Bit	7	6	5	4	3	2	1	0
Name	DPL1							
Reset Value	0x00							

Bit	**Name**	**Access**	**Description**
7:0	DPL1	R/W	The low byte of Data Pointer 1 register.

Data Pointers Select Register (DPS, 0x86)

Bit	7	6	5	4	3	2	1	0
Name	ID1	ID0	TSL	Reserved				SEL
Reset Value	0x00							

<table>
<tr><th>Bit</th><th>Name</th><th>Access</th><th>Description</th></tr>
<tr><td>0</td><td>SEL</td><td>R/W</td><td>Active DPTR register is selected by SEL bit. If SEL bit is equal to 0 then DPTR0 (0x83:0x82) is selected, otherwise DPTR1 (0x85:0x84).</td></tr>
<tr><td>4:1</td><td>Reserved</td><td>-</td><td></td></tr>
<tr><td>5</td><td>TSL</td><td>R/W</td><td>Toggle select enable. When set, this bit allows the following DPTR related instructions to toggle the SEL bit following execution of the instruction:
INC DPTR
MOV DPTR, #data16/#data24
MOVC A, @A+DPTR
MOVX @DPTR, A
MOVX A, @DPTR

When TSL=0, DPTR related instructions will not affect the state of the SEL bit.</td></tr>
<tr><td>7:6</td><td>ID1, ID0</td><td>R/W</td><td>Increment/decrement function select. See table below.
<table>
<tr><th>ID1</th><th>ID0</th><th>SEL=0</th><th>SEL=1</th></tr>
<tr><td>0</td><td>0</td><td>INC DPTR0</td><td>INC DPTR1</td></tr>
<tr><td>0</td><td>1</td><td>DEC DPTR0</td><td>INC DPTR1</td></tr>
<tr><td>1</td><td>0</td><td>INC DPTR0</td><td>DEC DPTR1</td></tr>
<tr><td>1</td><td>1</td><td>DEC DPTR0</td><td>DEC DPTR1</td></tr>
</table></td></tr>
</table>

Data Pointer Extended Registers

Data Pointer Extended registers DPX0, DPX1, MXAX hold the most significant part of memory address during access to data located above 64 K bytes. Note that DPX1 register is available only with DPTR1 register (DPH1, DPL1). During MOVX instruction using DPTR0/DPTR1 register, the most significant part of address bit [23:16] is always equal to DPX0 (0x93)/DPX1 (0x95) contents. During MOVX instruction using R0 or R1 register, the most significant part of address bit [23:16] is always equal to MXAX (0xEA) contents and address bit [15:8] is always equal to P2 (0xA0) contents.

Data Pointer EXtended 0 Register (DPX0, 0x93)

Bit	7	6	5	4	3	2	1	0
Name	DPX0							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	DPX0	R/W	Data Pointer Extended register DPX0 holds the most significant part of memory address during access to data located above 64 K bytes. During MOVX instruction using DPTR0 register, the most significant part of address bit [23:16] is always equal to DPX0 contents.

Data Pointer EXtended 1 Register (DPX1, 0x95)

Bit	7	6	5	4	3	2	1	0
Name	DPX1							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	DPX1	R/W	Data Pointer Extended register DPX1 holds the most significant part of memory address during access to data located above 64 K bytes. During MOVX instruction using DPTR1 register, the most significant part of address bit [23:16] is always equal to DPX1 contents.

MOVX @Ri EXtended Register (MXAX, 0xEA)

Bit	7	6	5	4	3	2	1	0
Name	MXAX							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	MXAX	R/W	Data Pointer Extended register MXAX holds the most significant part of memory address during access to data located above 64 K bytes. During MOVX instruction using R0 or R1 register, the most significant part of address bit [23:16] is always equal to MXAX contents and address bit [15:8] is always equal to P2 (0xA0) contents.

Stack Pointer

The 1T 80390 CPU core in both modes LARGE & FLAT has 8-bit stack pointer called SP (0x81) located in the internal RAM space. It is incremented before data is stored during PUSH and CALL execution and decremented after data is popped during POP, RET and RETI execution. In other words it always points to the last valid stack byte. The SP is accessed as any other SFRs. An example stack bytes order after some CALL instruction is shown in figure below.

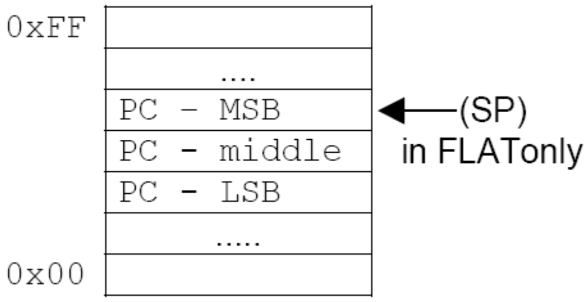


Figure 30: Stack Bytes Order

Stack Pointer Register (SP, 0x81)

Bit	7	6	5	4	3	2	1	0
Name	SP							
Reset Value	0x07							

Bit	Name	Access	Description
7:0	SP	R/W	The Stack Pointer register.

Internal Data Memory & SFRs Allocation

Please refer to section 3.4 Internal Data Memory and SFR Register Map for details.

4.4.4 Performance Improvement

This section presents performance benefits from using 1T 80390 CPU core over standard 8051 families.

8-Bit Arithmetic Functions

Addition

(a) Immediate data: The following code performs immediate data (constant) addition to an 8-bit register.

Rx = Rx + #n

Mnemonic		Opcode	Bytes	80C51 cycles	1T 80390 cycles
MOV	A, Rx	E8h - EFh	1	12	1
ADD	A, #n	24h	2	12	2
MOV	Rx, A	F8h - FFh	1	12	1
Sum:				36	4
80390 Performance Improvement:					9.0

(b) Direct addressing: The following code performs direct addressing addition to an 8-bit register.

Rx = Rx + (dir)

Mnemonic		Opcode	Bytes	80C51 cycles	1T 80390 cycles
MOV	A, Rx	E8h - EFh	1	12	1
ADD	A, dir	25h	2	12	2
MOV	Rx, A	F8h - FFh	1	12	1
Sum:				36	4
80390 Performance Improvement:					9.0

(c) Indirect addressing: The following code performs indirect addressing addition to an 8-bit register.

Rx = Rx + (@Rx)

Mnemonic		Opcode	Bytes	80C51 cycles	1T 80390 cycles
MOV	A, Rx	E8h - EFh	1	12	1
ADD	A, @Rx	26h - 27h	1	12	2
MOV	Rx, A	F8h - FFh	1	12	1
Sum:				36	4
80390 Performance Improvement:					9.0

(d) Register addressing: The following code performs an 8-bit register-to-register addition.

Rx = Rx + Ry

Mnemonic		Opcode	Bytes	80C51 cycles	1T 80390 cycles
MOV	A, Rx	E8h - EFh	1	12	1
ADD	A, @Ry	28h - 2Fh	1	12	1
MOV	Rx, A	F8h - FFh	1	12	1
Sum:				36	3
80390 Performance Improvement:					12.0

Subtraction

(a) Immediate data: The following code performs immediate data (constant) subtraction from an 8-bit register.

Rx = Rx - #n

Mnemonic		Opcode	Bytes	80C51 cycles	1T 80390 cycles
MOV	A, Rx	E8h - EFh	1	12	1
SUBB	A, #n	24h	2	12	2
MOV	Rx, A	F8h - FFh	1	12	1
Sum:				36	4
80390 Performance Improvement:					9.0

(b) Direct addressing: The following code performs direct addressing subtraction from an 8-bit register.

Rx = Rx - (dir)

Mnemonic		Opcode	Bytes	80C51 cycles	1T 80390 cycles
MOV	A, Rx	E8h - EFh	1	12	1
SUBB	A, dir	25h	2	12	2
MOV	Rx, A	F8h - FFh	1	12	1
Sum:				36	4
80390 Performance Improvement:					9.0

(c) Indirect addressing subtraction: The following code performs indirect addressing subtraction from an 8-bit register.

Rx = Rx - (@Ry)

Mnemonic		**Opcode**	**Bytes**	**80C51 cycles**	1T **80390 cycles**
MOV	A, @Ry				
MOV	A, Rx	E8h - EFh	1	12	1
SUBB	A, @Ry	26h - 27h	1	12	2
MOV	Rx, A	F8h - FFh	1	12	1
Sum:				36	4
80390 Performance Improvement:					9.0

(d) Register addressing subtraction: The following code performs an 8-bit register from register subtraction.

Rx = Rx - Ry

Mnemonic		**Opcode**	**Bytes**	**80C51 cycles**	**80390 cycles**
MOV	A, Rx	E8h - EFh	1	12	1
SUBB	A, Ry	28h - 2Fh	1	12	1
MOV	Rx, A	F8h - FFh	1	12	1
Sum:				36	3
80390 Performance Improvement:					12.0

Multiplication

The following code performs the 8-bit registers multiplication.

Rx = Rx * Ry

Mnemonic		Opcode	Bytes	80C51 cycles	80390 cycles
MOV	A, Rx	E8h - EFh	1	12	1
MOV	B, Ry	88h - 8Fh	2	24	2
MUL	AB	A4h	1	48	2
MOV	Rx, A	F8h - FFh	1	12	1
Sum:				96	6
80390 Performance Improvement:				16.0	

Division

The following code performs the 8-bit registers division.

Rx = Rx / Ry

Mnemonic		Opcode	Bytes	80C51 cycles	80390 cycles
MOV	A, Rx	E8h - EFh	1	12	1
MOV	B, Ry	88h - 8Fh	2	24	2
DIV	AB	84h	1	48	6
MOV	Rx, A	F8h - FFh	1	12	1
Sum:				96	10
80390 Performance Improvement:				9.6	

16-Bit Arithmetic Functions

Addition

The following code performs 16-bit addition. The first operand and result are located in registers pair RaRb. Second operand is located in registers pair RxRy.

RaRb = RaRb + RxRy

Mnemonic		Opcode	Bytes	80C51 cycles	80390 cycles
MOV	A, Rb	E8h - EFh	1	12	1
ADD	A, Ry	28h - 2Fh	1	12	1
MOV	Rb, A	F8h - FFh	1	12	1
MOV	A, Ra	E8h - EFh	1	12	1
ADDC	A, Rx	38h - 3Fh	1	12	1
MOV	Ra, A	F8h - FFh	1	12	1
Sum:				72	6
80390 Performance Improvement:				12.0	

Subtraction

The following code performs 16-bit subtraction. The first operand and result are located in registers pair RaRb. Second operand is located in registers pair RxRy.

RaRb = RaRb - RxRy

Mnemonic		Opcode	Bytes	80C51 cycles	80390 cycles
CLR	C	C3h	1	12	1
MOV	A, Rb	E8h - EFh	1	12	1
SUBB	A, Ry	28h - 2Fh	1	12	1
MOV	Rb, A	F8h - FFh	1	12	1
MOV	A, RA	E8h - EFh	1	12	1
SUBB	A, Rx	98h - 9Fh	1	12	1

MOV Ra, A	F8h - FFh	1	12	1

Sum: 84 7

80390 Performance Improvement: 12.0

Multiplication

The following code performs 16-bit multiplication. The first operand and result are located in registers pair RaRb. Second operand is located in registers pair RxRy.

RaRb = RaRb * RxRy

Mnemonic	Opcode	Bytes	80C51 cycles	80390 cycles
MOV A, Rb	E8h - EFh	1	12	1
MOV B, Ry	88h - 8Fh	2	24	2
MUL AB	A4 h	1	48	2
MOV Rz, B	A8h - AFh	2	24	3
XCH A, Rb	C8h - CFh	1	12	2
MOV B, Rx	88h - 8Fh	2	24	2
MUL AB	A4h	1	48	2
ADD A, Rz	28h - 2Fh	1	12	1
XCH A, Ra	C8h - CFh	1	12	2
MOV B, Ry	88h - 8Fh	2	24	2
MUL AB	A4h	1	48	2
ADD A, Ra	28h - 2Fh	1	12	1
MOV Ra, A	F8h - FFh	1	12	1

Sum: 312 23

80390 Performance Improvement: 13.6

32-Bit Arithmetic Function

Addition

The following code performs 32-bit addition. The first operand and result are located in four registers RaRbRcRd. Second operand is located in four registers RvRxRyRz.

RaRbRcRd = RaRbRcRd + RvRxRyRz

Mnemonic	Opcode	Bytes	80C51 cycles	80390 cycles
MOV A,Rd	E8h - EFh	1	12	1
ADD A, Rz	28h - 2Fh	1	12	1
MOV Rd, A	F8h - FFh	1	12	1
MOV A, Rc	E8h - EFh	1	12	1
ADDC A, Ry	38h - 3Fh	1	12	1
MOV Rc, A	F8h - FFh	1	12	1
MOV A, Rb	E8h - EFh	1	12	1
ADDC A, Rx	38h - 3Fh	1	12	1
MOV Rb, A	F8h - FFh	1	12	1
MOV A, Ra	E8h - EFh	1	12	1
ADDC A, Rv	38h - 3Fh	1	12	1
MOV Ra, A	F8h - FFh	1	12	1

Sum: 144 12

80390 Performance Improvement: 12.0

Subtraction

The following code performs 32-bit subtraction. The first operand and result are located in four registers RaRbRcRd. Second operand is located in four registers RvRxRyRz.

RaRbRcRd = RaRbRcRd - RvRxRyRz

Mnemonic		Opcode	Bytes	80C51 cycles	80390 cycles
CLR	C	C3h	1	12	1
MOV	A, Rd	E8h - EFh	1	12	1
SUBB	A, Rz	98h - 9Fh	1	12	1
MOV	Rd, A	F8h - FFh	1	12	1
MOV	A, Rc	E8h - EFh	1	12	1
SUBB	A, Ry	98h - 9Fh	1	12	1
MOV	Rc, A	F8h - FFh	1	12	1
MOV	A, Rb	E8h - EFh	1	12	1
SUBB	A, Rx	98h - 9Fh	1	12	1
MOV	Rb, A	F8h - FFh	1	12	1
MOV	A, Ra	E8h - EFh	1	12	1
SUBB	A, Rv	98h - 9Fh	1	12	1
MOV	Ra, A	F8h - FFh	1	12	1
Sum:				156	13

80390 Performance Improvement: 12.0

Multiplication

The following code performs 32-bit multiplication. The first operand and result are located in four registers RaRbRcRd. Second operand is located in four registers RvRxRyRz.

RaRbRcRd = RaRbRcRd * RvRxRyRz

Mnemonic		Opcode	Bytes	80C51 cycles	80390 cycles
MOV	A, R0	E8h - EFh	1	12	1
MOV	B, R7	88h - 8Fh	2	24	2
MUL	AB	A4 h	1	48	2
XCH	A, R4	C8h - CFh	1	12	2
MOV	B, R3	88h - 8Fh	2	24	2
MUL	AB	A4h	1	48	2
ADD	A, R4	28h - 2Fh	1	12	1
MOV	R4, A	F8h - FFh	1	12	1
MOV	A, R1	E8h - EFh	1	12	1
MOV	B, R6	88h - 8Fh	2	24	2
MUL	AB	A4h	1	48	2
ADD	A, R4	28h - 2Fh	1	12	1
MOV	R4, A	F8h - FFh	1	12	1
MOV	B, R2	88h - 8Fh	2	24	2
MOV	A, R5	E8h - EFh	1	12	2
MUL	AB	A4h	1	48	2
ADD	A, R4	28h - 2Fh	1	12	1
MOV	R4, A	F8h - FFh	1	12	1
MOV	A, R2	E8h - EFh	1	12	1
MOV	B, R6	88h - 8Fh	2	24	2
MUL	AB	A4h	1	48	2
XCH	A, R5	C8h - CFh	1	12	2
MOV	R0, B	A8h - AFh	2	24	3
MOV	B, R3	88h - 8Fh	2	24	2
MUL	AB	A4h	1	48	2
ADD	A, R5	28h - 2Fh	1	12	1
XCH	A, R4	C8h - CFh	1	12	2
ADDC	A, R0	38h - 3Fh	1	12	1

ADD	A, B	25h	2	12	2
MOV	R5, A	F8h - FFh	1	12	1
MOV	A, R1	E8h - EFh	1	12	1
MOV	B, R7	88h - 8Fh	2	24	2
MUL	AB	A4h	1	48	2
ADD	A, R4	28h - 2Fh	1	12	1
XCH	A, R5	C8h - CFh	1	12	2
ADDC	A, B	35h	2	12	2
MOV	R4, A	F8h - FFh	1	12	1
MOV	A, R3	E8h - EFh	1	12	1
MOV	B, R6	88h - 8Fh	2	24	2
MUL	AB	A4h	1	48	2
MOV	R6, A	F8h - FFh	1	12	1
MOV	R1, B	A8h - AFh	2	24	3
MOV	A, R3	E8h - EFh	1	12	1
MOV	B, R7	88h - 8Fh	2	24	2
MUL	AB	A4h	1	48	2
XCH	A, R7	C8h - CFh	1	12	2
XCH	A, B	C5h	2	12	3
ADD	A, R6	28h - 2Fh	1	12	1
XCH	A, R5	C8h - CFh	1	12	2
ADDC	A, R1	38h - 3Fh	1	12	1
MOV	R6, A	F8h - FFh	1	12	1
CLR	A	E4h	1	12	1
ADDC	A, R4	38h - 3Fh	1	12	1
MOV	R4, A	F8h - FFh	1	12	1
MOV	A, R2	E8h - EFh	1	12	1
MUL	AB	A4h	1	48	2
ADD	A, R5	28h - 2Fh	1	12	1
XCH	A, R6	C8h - CFh	1	12	2
ADDC	A, B	38h – 3Fh	2	12	2
MOV	R5, A	F8h - FFh	1	12	1
CLR	A	E4h	1	12	1
ADDC	A, R4	38h - 3Fh	1	12	1
MOV	R4, A	F8h - FFh	1	12	1
Sum:				1248	99
80390 Performance Improvement:				12.6	

Performance Improvement Summary

Total performance improvement has been summarized in the table below. It shows the most common used multi-precision arithmetic operation.

Function	80C51 cycle	80390 cycle	Improvement
8-bit addition (immediate data)	36	4	9.0
8-bit addition (direct addressing)	36	4	9.0
8-bit addition (indirect addressing)	36	4	9.0
8-bit addition (register addressing)	36	3	12.0
8-bit subtraction (immediate data)	36	4	9.0
8-bit subtraction (direct addressing)	36	4	9.0
8-bit subtraction (indirect addressing)	36	4	9.0
8-bit subtraction (register addressing)	36	3	12.0
8-bit multiplication	96	6	16.0
8-bit division	96	10	9.6
16-bit addition	72	6	12.0
16-bit subtraction	84	7	12.0
16-bit multiplication	312	23	13.6
32-bit addition	144	12	12.0

32-bit subtraction	156	13	12.0
32-bit multiplication	1248	99	12.6
Average speed improvement:			11.12

4.4.5 Debugger

Flash Programming

The debugger fully supports programming of all Flash memory devices. Such support is assured by configurability of Flash programming algorithm, and supported devices database. New Flash device can be easily added to existing base using build-in editor. The debugger allows user to simply perform in-system programming of its Flash memory without using any external equipment. Flash programming task is performed directly within Debug software, and after uploading of code, it is ready for debugging. Programming time is very short, because of HAD2 support. This feature saves time, and makes usage of debugger very comfortable and flexible.

Non-Intrusive System

In typical intrusive systems a debugging tool consumes for its own needs some system resources e.g.: part of program space, several cells of RAM memory, ports' pins sometimes system is loosing interrupts or the program code is manipulated to support software breakpoints, and so on. Even simple debugging system consumes the UART and timer resources to support own tasks. These simple 'emulators' cannot provide trace and other advanced debugging functions, while also being very intrusive in the debugging cycle. Imagine trying to debug an interrupt problem while the 'emulator' is manipulating interrupts itself!

Developing firmware is all about producing code that is 100% reliable in operation and fully understood in how it will perform in adverse conditions. A real non-intrusive on-chip debugger that assists user in this task is the most important tool user can have. That is the reason why using of non-intrusive systems is so important. The debugger and debug software tools has been designed as a non-intrusive system.

Real-time Hardware Debugger

Real-time hardware debugger we call for a tool that is able to detect processor internal properties that are not visible outside the processor without any violation of real-time operations. The debugger gives you the chance to track down hidden bugs within the application running with micro-controller. Internal events such as the reading of the SBUF-control register are not mirrored on the external address-data bus. However, by using special logic to detect operations that affect internal resources, debugger gives user ability to track such internal events without any violation of real-time operation. There is no need to use a special external logic for the emulation.

4.5 On-Chip Flash Memory

Block Diagram

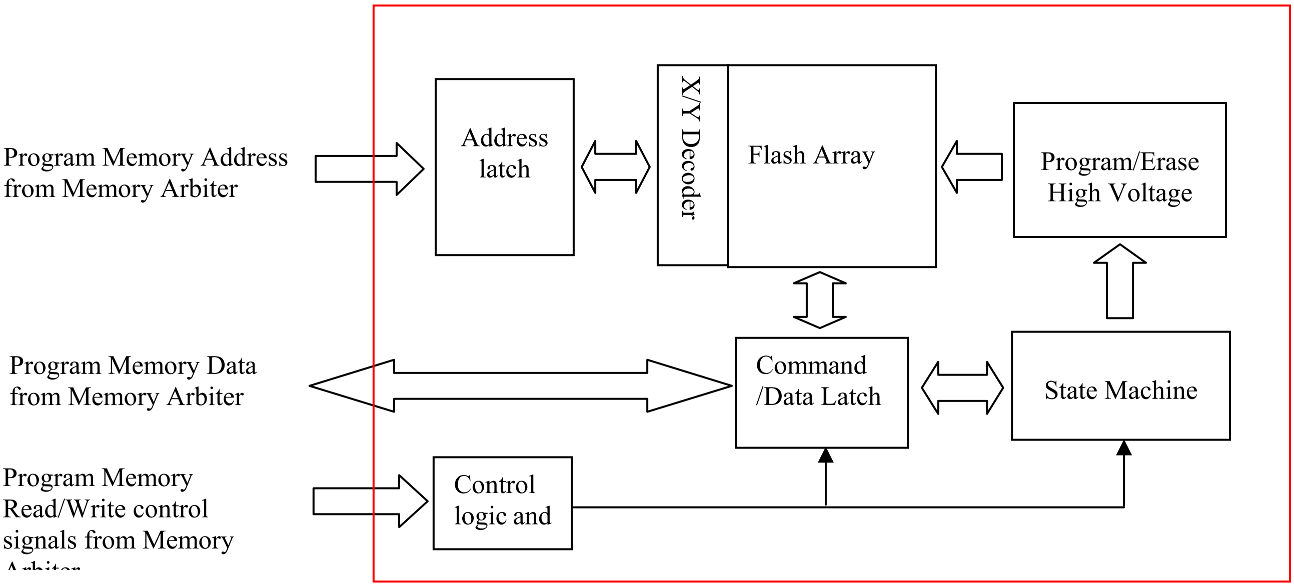

Program Memory Address from Memory Arbiter
Program Memory Data from Memory Arbiter
Program Memory Read/Write control signals from Memory Arbiter
Address latch
X/Y Decoder
Flash Array
Program/Erase High Voltage
Command /Data Latch
State Machine
Control logic and

Figure 31: On-Chip Flash Memory Block Diagram

Sector Structure

Sector	Sector Size	Address Range	Sector Address					
			A18	A17	A16	A15	A14	A13
SA0	16Kbytes	00000-03FFF	0	0	0	0	0	X
SA1	8Kbytes	04000-05FFF	0	0	0	0	1	0
SA2	8Kbytes	06000-07FFF	0	0	0	0	1	1
SA3	32Kbytes	08000-0FFFF	0	0	0	1	X	X
SA4	64Kbytes	10000-1FFFF	0	0	1	X	X	X
SA5	64Kbytes	20000-2FFFF	0	1	0	X	X	X
SA6	64Kbytes	30000-3FFFF	0	1	1	X	X	X
SA7	64Kbytes	40000-4FFFF	1	0	0	X	X	X
SA8	64Kbytes	50000-5FFFF	1	0	1	X	X	X
SA9	64Kbytes	60000-6FFFF	1	1	0	X	X	X
SA10	64Kbytes	70000-7FFFF	1	1	1	X	X	X

Table 6: On-Chip Flash Memory Sector Structure

Automatic Programming

The on-chip Flash memory is byte programmable using the Automatic Programming algorithm. The Automatic Programming algorithm makes system do not need to have time-out sequence nor to verify the data programmed.

Automatic Chip Erase

The entire on-chip Flash memory is bulk erased using 10ms erase pulses according to Automatic Chip Erase algorithm. Typical erasure at room temperature is accomplished in less than 4 second. The Automatic Erase algorithm automatically programs the entire array prior to electrical erase. The timing and verification of electrical erase are controlled internally within the Flash memory.

Automatic Sector Erase

The on-chip Flash memory is sector(s) erasable using Automatic Sector Erase algorithm. The Automatic Sector Erase algorithm automatically programs the specified sector(s) prior to electrical erase. The timing and verification of electrical erase are controlled internally within the Flash memory. An erase operation can erase one sector, multiple sectors, or the entire Flash memory.

Automatic Programming Algorithm

The Automatic Programming algorithm requires the user to only write program set-up commands (including 2 unlock write cycle and A0H) and a program command (program data and address). The Flash memory automatically times the programming pulse width, provides the program verification, and counts the number of sequences. During a program cycle, the state-machine will control the program sequences and command register will not respond to any command set. A status bit similar to Data# Polling and a status bit toggling between consecutive read cycles, provide feedback to the user as to the status of the programming operation. Refer to write operation status, Table 8, for more information on these status bits.

Automatic Erase Algorithm

The Automatic Erase algorithm requires the user to write commands to the command register. The Flash memory will automatically pre-program and verify the entire array. Then the Flash memory automatically times the erase pulse width, provides the erase verification, and counts the number of sequences. A status bit toggling between consecutive read cycles provides feedback to the user as to the status of the erasing operation.

Register contents serve as inputs to an internal state-machine, which controls the erase and programming circuitry. During write cycles, the command register internally latches address and data needed for the programming and erase operations.

During a Sector Erase cycle, the command register will only respond to Erase Suspend command. After Erase Suspend is completed, the Flash memory stays in read mode. After the state machine has completed its task, it will allow the command register to respond to its full command set.

Command Definitions

Flash memory operations are selected by writing specific address and data sequences into the command register. Writing incorrect address and data values or writing them in the improper sequence will reset the Flash memory to the Read mode. Table 7 defines the valid register command sequences. Note that the Erase Suspend (B0H) and Erase Resume (30H) commands are valid only while the Sector Erase operation is in progress.

An erase operation can erase one sector, multiple sectors, or the entire Flash memory. Table 6 indicates the address space that each sector occupies. A "sector address" consists of the address bits required to uniquely select a sector. The writing specific address and data commands or sequences into the command register initiates the Flash memory operations.

Command	Bus Cycle Required	1st Bus Cycle		2nd Bus Cycle		3rd Bus Cycle		4th Bus Cycle		5th Bus Cycle		6th Bus Cycle	
		Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data
Reset	1	XXXH	F0H										
Read	1	RA	RD										
Program	4	555H	AAH	2AAH	55H	555H	A0H	PA	PD				
Chip Erase	6	555H	AAH	2AAH	55H	555H	80H	555H	AAH	2AAH	55H	555H	10H
Sector Erase	6	555H	AAH	2AAH	55H	555H	80H	555H	AAH	2AAH	55H	SA	30H
Sector Erase Suspend	1	XXXH	B0H										
Sector Erase Resume	1	XXXH	30H										

Note:

1. RA=Address of memory location to be read. RD=Data to be read at location RA.
2. PA = Address of memory location to be programmed.
   PD = Data to be programmed at location PA.
   SA = Address of the sector to be erased.
3. The software should generate the following address patterns: 555H or 2AAH to Address A11~A0.
   Address bit A12~A18 = X = Don't care for all address commands except for Program Address (PA) and Sector Address (SA). Write Sequence may be initiated with A12~A18 in either state.

Table 7: On-Chip Flash Memory Command Definitions

Read Flash Array Data

The internal state machine is set for reading array data upon Flash memory power-up, or after a hardware reset. This ensures that no spurious alteration of the memory content occurs during the power transition. No command is necessary in this mode to obtain array data. The Flash memory remains enabled for read access until the command register contents are altered. The Flash memory is also ready to read array data after completing an Automatic Program or Automatic Erase algorithm.

After the Flash memory accepts an Erase Suspend command, the Flash memory enters the Erase Suspend mode. The CPU can read array data using the standard read timings, except that if it reads at an address within erase-suspended sectors, the Flash memory outputs status data. After completing a programming operation in the Erase Suspend mode, the CPU may once again read array data with the same exception. See "Erase Suspend/Erase Resume Commands" for more information on this mode. The system must issue the reset command to re-enable the device for reading array data if Q5 (data bit 5) goes high, or while in the auto-select mode. See the "Reset Command" section, next.

Reset Command

The reset operation is initiated by writing the reset command sequence into the command register. The Flash memory remains enabled for reads until the command register contents are altered. If program-fail or erase-fail happen, the write of F0H will reset the Flash memory to abort the operation. Address bits are don't-care for this command. A valid command must then be written to place the Flash memory in the desired state. Writing the reset command to the Flash memory resets the device to reading array data.

The reset command may be written between the sequence cycles in an erase command sequence before erasing begins. This resets the device to reading array data. Once erasure begins, however, the Flash memory ignores reset commands until the operation is complete.

The reset command may be written between the sequence cycles in a program command sequence before programming begins. This resets the Flash memory to reading array data (also applies to programming in Erase Suspend mode). Once programming begins, however, the Flash memory ignores reset commands until the operation is complete.

If Q5 (data bit 5) goes high during a program or erase operation, writing the reset command returns the Flash memory to reading array data (also applies during Erase Suspend).

Automatic Chip Erase Commands

Chip Erase is a six-bus cycle operation. There are two "unlock" write cycles. These followed by writing the set-up command 80H. The second "unlock" write cycles are then followed by the chip erase command 10H.

The Automatic Chip Erase does not require the Flash memory to be entirely pre-programmed prior to executing the Automatic Chip Erase. Upon executing the Automatic Chip Erase, the state machine in Flash memory will automatically program and verify the entire Flash memory for an all-zero data pattern. When the Flash memory is automatically verified to contain an all-zero pattern, a self-timed chip erase and verify begin. The erase and verify operations are completed when the data on Q7 (data bit 7) is "1" at which time the Flash memory returns to the Read mode. The software is not required to provide any control or timing during these operations.

When using the Automatic Chip Erase algorithm, note that the erase automatically terminates when adequate erase margin has been achieved for the memory array (no erase verification command is required).

If the Erase operation was unsuccessful, the data on Q5 (data bit 5) is "1"(see Table 8), indicating the erase operation exceed internal timing limit.

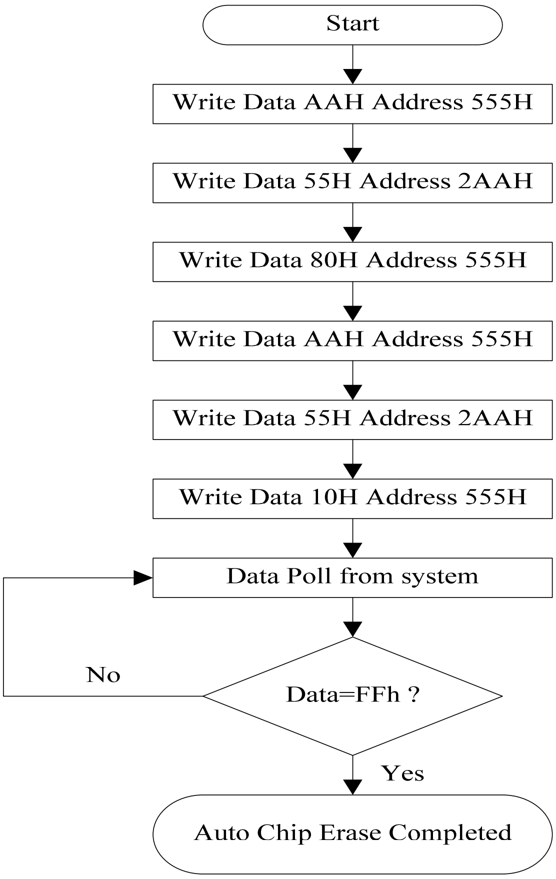

Start
Write Data AAH Address 555H
Write Data 55H Address 2AAH
Write Data 80H Address 555H
Write Data AAH Address 555H
Write Data 55H Address 2AAH
Write Data 10H Address 555H
Data Poll from system
No
Data=FFh ?
Yes
Auto Chip Erase Completed

Figure 32: Automatic Chip Erase Algorithm Flowchart

Automatic Sector Erase Commands

Sector Erase is a six-bus cycle operations. There are two "unlock" write cycles. These followed by writing the set-up command 80H. Two more "unlock" write cycles are then followed by the sector erase command 30H. Sector addresses selected are loaded into internal register on the sixth command.

The Automatic Sector Erase does not require the Flash memory to be entirely pre-programmed prior to executing the Automatic Sector Erase Set-up commands and Automatic Sector Erase command. Upon executing the Automatic Sector Erase command, the Flash memory will automatically program and verify the sector(s) memory for an all-zero data pattern. The software is not required to provide any control or timing during these operations.

When the sector(s) is automatically verified to contain an all-zero pattern, a self-timed sector erase and verify begin. The erase and verify operations are completed when the data on Q7 (data bit 7) is "1" and the data on Q6 (data bit 6) stops toggling for two consecutive read cycles, at which time the Flash memory returns to the Read mode. The software is not required to provide any control or timing during these operations.

When using the Automatic Sector Erase algorithm, note that the erase automatically terminates when adequate erase margin has been achieved for the memory array (no erase verification command is required).

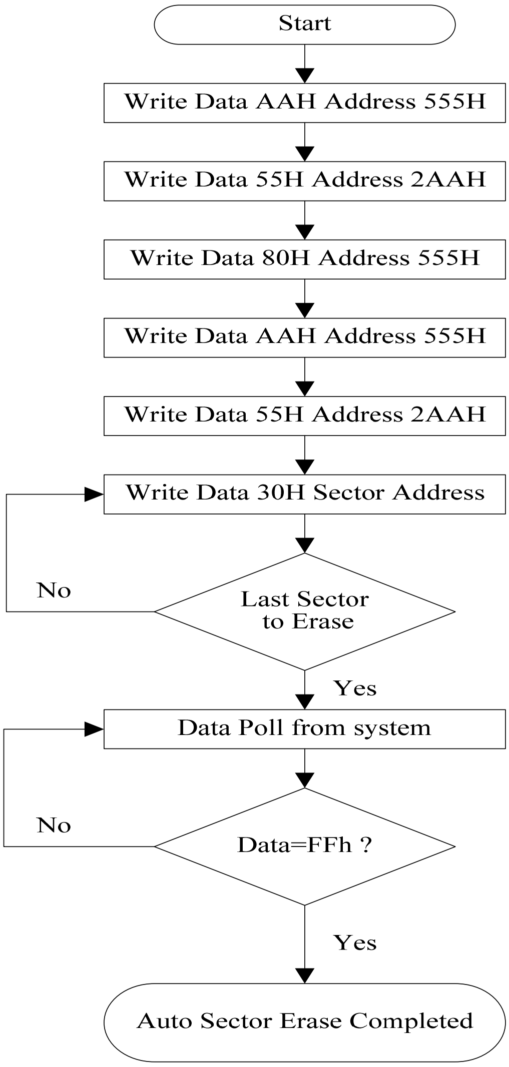

Start
Write Data AAH Address 555H
Write Data 55H Address 2AAH
Write Data 80H Address 555H
Write Data AAH Address 555H
Write Data 55H Address 2AAH
Write Data 30H Sector Address
No
Last Sector to Erase
Yes
Data Poll from system
No
Data=FFh ?
Yes
Auto Sector Erase Completed

Figure 33: Automatic Sector Erase Algorithm Flowchart

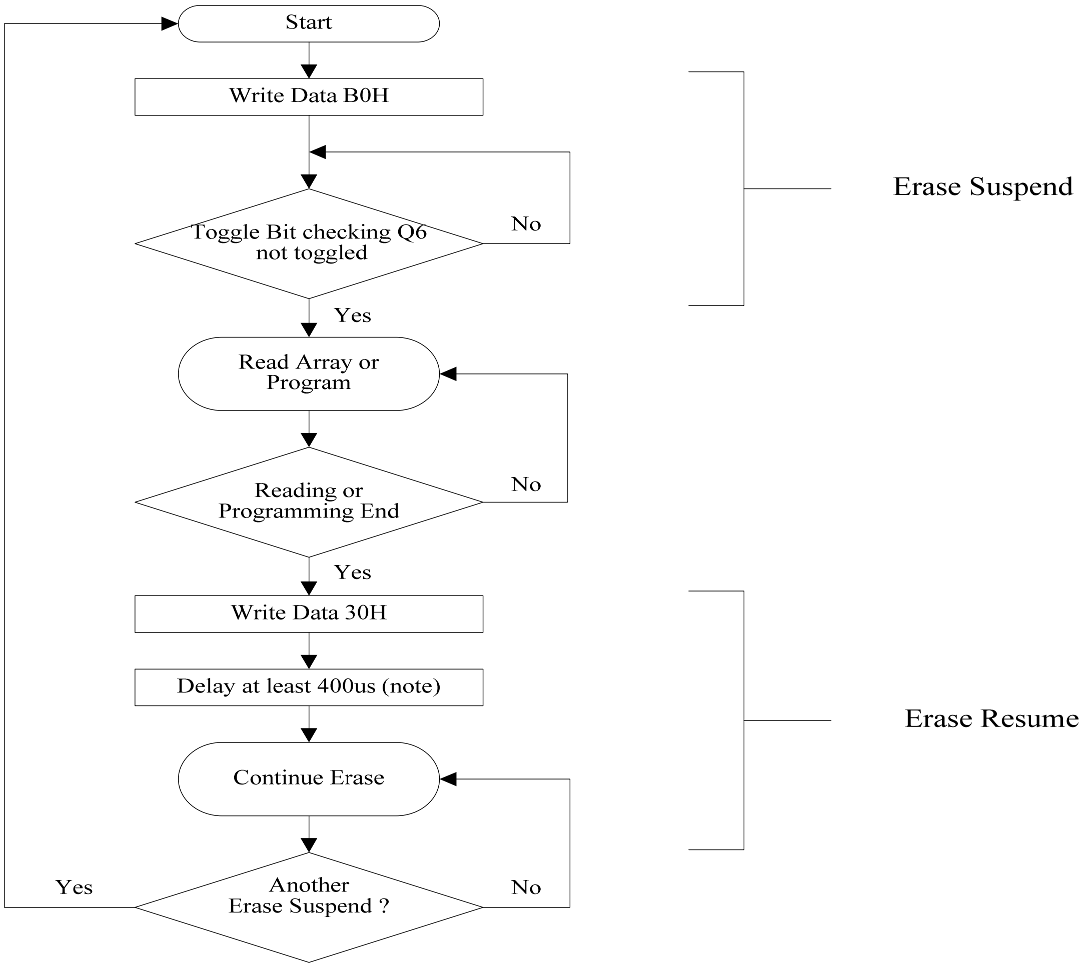
Erase Suspend

This command only has meaning while the state machine is executing Automatic Sector Erase operation, and therefore will only be responded during Automatic Sector Erase operation. When the Erase Suspend command is written during a sector erase operation, the Flash memory requires a maximum of 20us to suspend the erase operations. However, When the Erase Suspend command is written during the sector erase time-out, the device immediately terminates the time-out period and suspends the erase operation. After this command has been executed, the command register will initiate erase suspend mode. The state machine will return to read mode automatically after suspend is ready. At this time, state machine only allows the command register to respond to the Read Memory Array, Erase Resume and Program commands.

The system can determine the status of the program operation using the Q7 (data bit 7) or Q6 (data bit 6) status bits, just as in the standard program operation. After an erase-suspend operation is complete, the software can once again read array data within non-suspended sectors.

Erase Resume

This command will cause the command register to clear the suspend state and return back to Sector Erase mode but only if an Erase Suspend command was previously issued. Erase Resume will not have any effect in all other conditions. Another Erase Suspend command can be written after the chip has resumed erasing. The minimum time from Erase Resume to next Erase Suspend is 400us. Repeatedly suspending the device more often may have undetermined effects.



Note: Repeatedly suspending the device more often may have undetermined effects.

Figure 34: Erase Suspend/Erase Resume Flowchart

Automatic Program Commands

To initiate Automatic Program mode, a three-cycle command sequence is required. There are two "unlock" write cycles, followed by writing the Automatic Program command A0H. The program address and data are written next, which in turn initiate the embedded Program Algorithm. Once the Automatic Program command is initiated, the next write causes a transition to an active programming operation. The software is not required to provide further controls or timings. The Flash memory will automatically provide an adequate internally generated program pulse and verify the programmed cell margin.

When the embedded Program algorithm is complete, the Flash memory then returns to reading array data and addresses are no longer latched. The Flash memory provides Q2, Q3, Q5, Q6 and Q7 (i.e., data bit 2, 3, 5, 6, and 7) to determine the status of a write operation. If the program operation was unsuccessful, the data on Q5 is "1"(see Table 8), indicating the program operation exceed internal timing limit. The automatic programming operation is completed when the data read on Q6 stops toggling for two consecutive read cycles and the data on Q7 and Q6 are equivalent to data written to these two bits, at which time the Flash memory returns to the Read mode (no program verify command is required).

Any commands written to the Flash memory during the embedded Program Algorithm are ignored. Note that a hardware reset on RST_N pin immediately terminates the programming operation. The Byte Program command sequence should be reinitiated once the Flash memory has reset to reading array data, to ensure data integrity. Programming is allowed in any sequence and across sector boundaries. Please note that a bit cannot be programmed from a "0" back to a "1". Attempting to do so may halt the operation and set the Q5 to "1", or cause the Data# Polling algorithm to indicate the operation was successful. However, a succeeding read will show that the data is still "0". Only erase operations can convert a "0" to a "1".

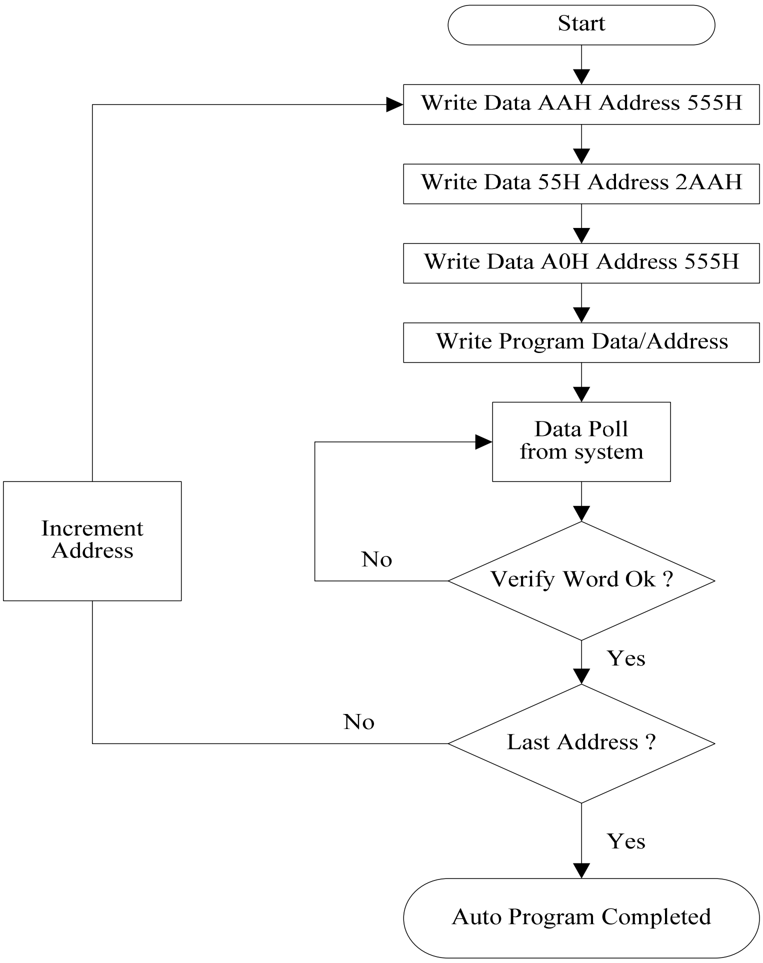

Start
Write Data AAH Address 555H
Write Data 55H Address 2AAH
Write Data A0H Address 555H
Write Program Data/Address
Data Poll from system
Increment Address
No
Verify Word Ok ?
Yes
No
Last Address ?
Yes
Auto Program Completed

Figure 35: Automatic Programming Algorithm Flowchart

Write Operation Status

The Flash memory provides several bits on data bus to determine the status of a write operation: Q2, Q3, Q5, Q6 and Q7. Table 8 and the following subsections describe the functions of these bits. Q7 and DQ6 each offer a method for determining whether a program or erase operation is complete or in progress. These three bits are discussed first.

	Status		Q7 (Note 1)	Q6	Q5 (Note 2)	Q3	Q2
In Progress	Byte Program in Auto Program Algorithm		Q7#	Toggle	0	N/A	No Toggle
	Auto Erase Algorithm		0	Toggle	0	1	Toggle
	Erase Suspended Mode	Erase Suspend Read (Erase Suspended Sector)	1	No Toggle	0	N/A	Toggle
		Erase Suspend Read Non-Erase Suspended Sector)	Data	Data	Data	Data	Data
		Erase Suspend Program	Q7#	Toggle	0	N/A	N/A
Exceeded Time Limits	Byte Program in Auto Program Algorithm		Q7#	Toggle	1	N/A	No Toggle
	Auto Erase Algorithm		0	Toggle	1	1	Toggle
	Erase Suspend Program		Q7#	Toggle	1	N/A	N/A

Note:

1. Q7 and Q2 require a valid address when reading status information. Refer to the appropriate subsection for further details.
2. Q5 switches to '1' when an Auto Program or Auto Erase operation has exceeded the maximum timing limits. See "Q5: Exceeded Timing Limits " for more information.

Table 8: Write Operation Status

Q7: Data# Polling

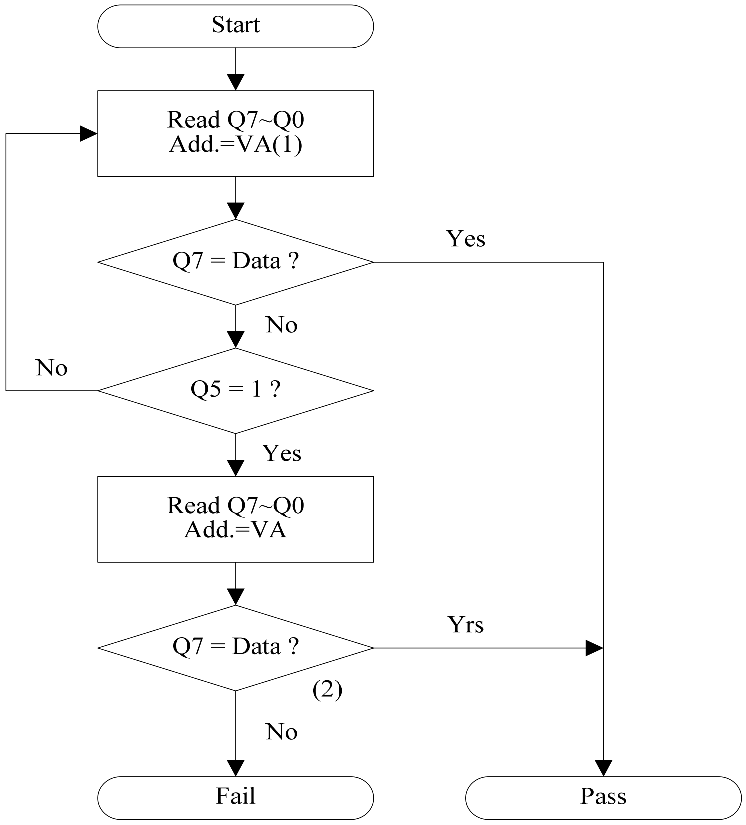
The Data# Polling bit, Q7, indicates to the software whether an Automatic Algorithm is in progress or completed, or whether the Flash memory is in Erase Suspend.

During the Automatic Program algorithm, the Flash memory outputs on Q7 the complement of the datum programmed to Q7. This Q7 status also applies to programming during Erase Suspend. When the Automatic Program algorithm is complete, the device outputs the datum programmed to Q7. The software must provide the program address to read valid status information on Q7. If a program address falls within a protected sector, Data# Polling on Q7 is active for approximately 1 us, then the device returns to reading array data.

During the Automatic Erase algorithm, Data# Polling produces a "0" on Q7. When the Automatic Erase algorithm is complete, or if the Flash memory enters the Erase Suspend mode, Data# Polling produces a "1" on Q7. This is analogous to the complement/true datum output described for the Automatic Program algorithm: the erase function changes all the bits in a sector to "1" prior to this, the Flash memory outputs the "complement," or "0". The software must provide an address within any of the sectors selected for erasure to read valid status information on Q7.

After an erase command sequence is written, if all sectors selected for erasing are protected, Data# Polling on Q7 is active for approximately 100 us, then the Flash memory returns to reading array data. If not all selected sectors are protected, the Automatic Erase algorithm erases the unprotected sectors, and ignores the selected sectors that are protected.

When the software detects Q7 has changed from the complement to true data, it can read valid data at Q7-Q0 on the following read cycles.



Note:

1. VA=Valid address for programming
2. Q7 should be re-checked even Q5="1" because Q7 may change simultaneously with Q5.

Figure 36: Data# Polling Algorithm

Q6: Toggle BIT I

Toggle Bit I on Q6 indicates whether an Automatic Program or Erase algorithm is in progress or complete, or whether the Flash memory has entered the Erase Suspend mode. Toggle Bit I may be read at any address, in the command sequence (prior to the program or erase operation), and during the sector timeout.

During an Automatic Program or Erase algorithm operation, successive read cycles to any address cause Q6 to toggle. When the operation is complete, Q6 stops toggling.

After an erase command sequence is written, if all sectors selected for erasing are protected, Q6 toggles and returns to reading array data. If not all selected sectors are protected, the Automatic Erase algorithm erases the unprotected sectors, and ignores the selected sectors that are protected.

The software can use Q6 and Q2 together to determine whether a sector is actively erasing or is erase be suspended. When the Flash memory is actively erasing (that is, the Automatic Erase algorithm is in progress), Q6 start toggling. When the Flash memory enters the Erase Suspend mode, Q6 stops toggling. However, the software must also use Q2 to determine which sectors are erasing or erase-suspended. Alternatively, the system can use Q7.

If a program address falls within a protected sector, Q6 toggles for approximately 2 us after the program command sequence is written, then returns to reading array data.

Q6 also toggles during the erase-suspend-program mode and stops toggling once the Automatic Program algorithm is complete.

Q2: Toggle Bit II

The "Toggle Bit II" on Q2, when used with Q6, indicates whether a particular sector is actively erasing (that is, the Automatic Erase algorithm is in process), or whether that sector is erase-suspended.

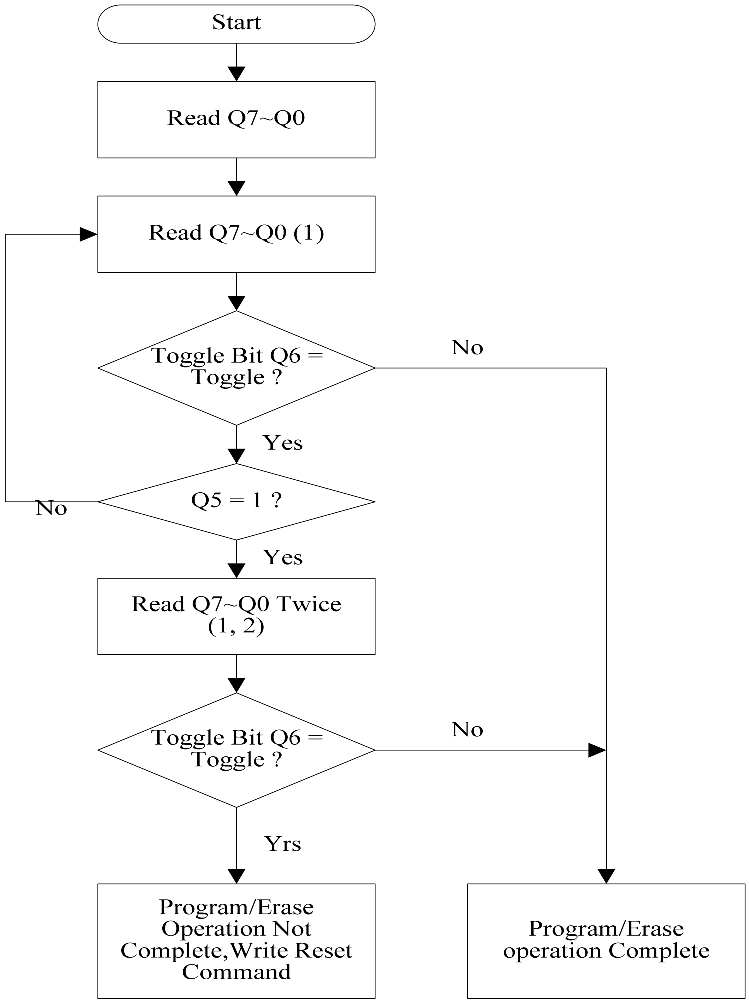
Q2 toggles when the software reads at addresses within those sectors that have been selected for erasure. But Q2 cannot distinguish whether the sector is actively erasing or is erase-suspended. Q6, by comparison, indicates whether the Flash memory is actively erasing, or is in Erase Suspend, but cannot distinguish which sectors are selected for erasure. Thus, both status bits are required for sectors and mode information. Refer to Table 8 to compare outputs for Q2 and Q6.

Reading Toggle Bits Q6/ Q2

Whenever the software initially begins reading toggle bit status, it must read Q7-Q0 at least twice in a row to determine whether a toggle bit is toggling. Typically, the software would note and store the value of the toggle bit after the first read. After the second read, the software would compare the new value of the toggle bit with the first. If the toggle bit is not toggling, the Flash memory has completed the program or erase operation. The software can read array data on Q7-Q0 on the following read cycle.

However, if after the initial two read cycles, the software determines that the toggle bit is still toggling, the software also should note whether the value of Q5 is high (see the section on Q5). If it is, the software should then determine again whether the toggle bit is toggling, since the toggle bit may have stopped toggling just as Q5 went high. If the toggle bit is no longer toggling, the Flash memory has successfully completed the program or erase operation. If it is still toggling, the Flash memory did not complete the operation successfully, and the software must write the reset command to return to reading array data.

The remaining scenario is that software initially determines that the toggle bit is toggling and Q5 has not gone high. The software may continue to monitor the toggle bit and Q5 through successive read cycles, determining the status as described in the previous paragraph. Alternatively, it may choose to perform other system tasks. In this case, the software must start at the beginning of the algorithm when it returns to determine the status of the operation.



Note: 1.Read toggle bit twice to determine whether or not it is toggling.
2. Recheck toggle bit because it may stop toggling as Q5 change to "1".

Figure 37: Toggle Bit Algorithm

Q5: Exceeded Timing Limits

Q5 will indicate if the program or erase time has exceeded the specified limits (internal pulse count). Under these conditions Q5 will produce a "1". This time-out condition indicates that the program or erase cycle was not successfully completed. Data# Polling and Toggle Bit are the only operating functions of the device under this condition.

If this time-out condition occurs during sector erase operation, it specifies that a particular sector is bad and it may not be reused. However, other sectors are still functional and may be used for the program or erase operation. The Flash memory must be reset to use other sectors. Write the Reset command sequence to the Flash memory, and then execute program or erase command sequence. This allows the software to continue to use the other active sectors in the Flash memory.

If this time-out condition occurs during the chip erase operation, it specifies that the entire chip is bad or combination of sectors is bad.

If this time-out condition occurs during the byte programming operation, it specifies that the entire sector containing that byte is bad and this sector may not be reused, (other sectors are still functional and can be reused).

The time-out condition will not appear if a user tries to program a non-blank location without erasing. Please note that this is not a Flash memory failure condition since the Flash memory was incorrectly used.

Q3: Sector Erase Timer

After the completion of the initial sector erase command sequence, the sector erase time-out will begin. Q3 will remain low until the time-out is complete. Data# Polling and Toggle Bit are valid after the initial sector erase command sequence.

If Data# Polling or the Toggle Bit indicates the Flash memory has been written with a valid erase command, Q3 may be used to determine if the sector erase timer window is still open. If Q3 is high ("1") the internally controlled erase cycle has begun; attempts to write subsequent commands to the Flash memory will be ignored until the erase operation is completed as indicated by Data# Polling or Toggle Bit. If Q3 is low ("0"), the Flash memory will accept additional sector erase commands. To insure the command has been accepted, the software should check the status of Q3 prior to and following each subsequent sector erase command. If Q3 were high on the second status check, the command may not have been accepted.

Erase and Programming Performance

Parameter	Limits			Unit
	Min.	Typ.	Max.	
Sector Erase Time		0.7	15	sec
Chip Erase Time		4	32	sec
Byte Programming Time		9	300	us
Chip Programming Time		4.5	13.5	sec
Erase/Program Cycles	100,000			cycles

Program Code Read Protection in On-chip Flash Memory

When the program code in on-chip Flash memory needs to be protected from unauthorized downloading for copyright protection purpose, the on-chip Flash memory offers a hardware mechanism to support this. The on-chip Flash memory location 0x03FFF in bit 7 is used to enable/disable the on-chip Flash memory read protection. Setting or clearing this bit will not affect normal program code execution by the CPU core. See below Table 9 for detailed description.

On-chip Flash Memory Location 0x003FFF		CPU Debugger Access
Bit 7 = Read Protection Disable bit	Bit [6:0]	
1	Reserved and put 0x00	CPU Debugger access is enabled. The program code can be downloaded from Flash memory to CPU Debugger software. This setting is usually used during software development in progress.
0	Reserved and put 0x00	CPU Debugger access is disabled. The program code cannot be downloaded from Flash memory to CPU Debugger software. This setting is usually used after the software development is complete and ready for production.

Table 9: On-chip Flash Memory Read Protection

4.6 Memory Arbiter & Boot Loader

Figure 38 below shows Memory Arbiter, Boot Loader, and Flash Programming Controller block diagram.

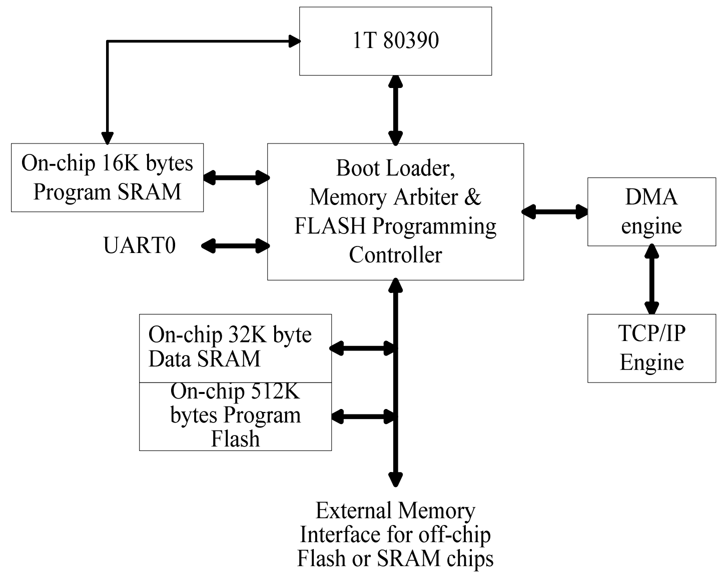

1T 80390
On-chip 16K bytes Program SRAM
Boot Loader, Memory Arbiter & FLASH Programming Controller
DMA engine
UART0
TCP/IP Engine
On-chip 32K byte Data SRAM
On-chip 512K bytes Program Flash
External Memory Interface for off-chip Flash or SRAM chips

Figure 38: Boot Loader, Memory Arbiter & Flash Programming Controller Block Diagram

4.6.1 Boot Loader

After power-on reset or software reboot command via setting SW_RBT bit (CSREPR.1), the boot loader will read the Flash memory to load the program code to the internal 16K byte Program SRAM and to the optional external Program SRAM first before allowing CPU core to start running. The setting of EXT_PROG_SRAM_EN pin determines whether the external Program SRAM is present or not and is used to determine whether to load to the external Program SRAM or not.

When EXT_PROG_SRAM_EN = 1, indicating the external program SRAM is present, the on-chip Flash memory address location, 0x00_3FFF, in bit [6:0] called "**BLOCK_NUMBER**" is used to tell the boot loader how many bytes of program code need to be loaded from the Flash memory to the external Program SRAM. Each block number represents 32K bytes of program code data. For example, if program code size is less than 32K Bytes, user shall write 0x01 in "BLOCK_NUMBER" byte. If program code size is between 32-64KB, then put 0x02 in "BLOCK_NUMBER" byte, etc. Note that the internal 16K bytes Program SRAM (boot code) is always loaded by boot loader regardless of the setting of EXT_PROG_SRAM_EN.

Note that in on-chip Flash memory address location 0x00_3FFF, the bit 7 is used for program code read protection bit. Please see Table 9 for details.

The "BLOCK_NUMBER" byte is also being passed to the memory arbiter to control the address bus conversion of External Memory Interface when the external SRAM chip(s) is used for both program code shadow mode and xDATA memory expansion purposes.

Clock Speed, Software Reset and Ext. Program Memory Select Register (CSREPR, 0x8Fh)

Bit	7	6	5	4	3	2	1	0
Name	SCS [1:0]		ICD	PMS	FAES	FARM	SW_RBT	SW_RST
Reset Value	00		0	0	0	0	0	0

Bit	Name	Access	Description
0	SW_RST	R/W1	Software Reset. Setting to "1" to reset all peripheral logics and CPU itself. Upon activated, this bit will be cleared by chip hardware automatically.
1	SW_RBT	R/W1	Software Reboot. Setting to "1" to reset and reboot the whole chip including CPU and all peripherals. This step will cause the Bootloader to redo the program mirroring/shadow step and reload the content of I2C Configuration EEPROM to related registers. Upon activated, this bit will be cleared by chip hardware automatically.
2	FARM	R/W	Flash Address Re-Mapping. See section 4.6.4 for detailed description. 1: To enable software to gain access to the first 16KB (0x00_0000~0x00_3FFF) of the on-chip Flash memory. After enabled, software can access to the first 16KB of the Flash memory by accessing the program memory space 16K~32K (x00_4000~0x007FFF) which then will be remapped to 0~16K (0x00_0000~0x00_3FFF) address space of on-chip Flash memory by the "Memory Arbiter" hardware. Note that when enabled, the software accessing to the lower 16KB of program memory space is still accessing to the internal 16KB Program SRAM as usual, and the accessing of the 16KB~32KB address space of on-chip Flash memory is temporarily disabled. 0: To disable software accessing the first 16KB of on-chip Flash memory. When disabled, software accessing to the program memory space 16K~32K (0x00_4000 ~ 0x007FFF) would be accessing to the same address space of the on-chip Flash memory without re-mapping (default). Note that when disabled, the software accessing to the first 16KB of program memory space is still accessing to the internal 16KB Program SRAM as usual.
3	FAES	R/W	Flash Access Enable in Shadow mode. See section 4.6.4 for detailed description. 1: To enable Flash memory access during enabling program code shadow mode. Normally, when an external Program SRAM is being used for program code shadow purpose, in that case, the CPU program read or write accesses to 16KB above (0x00_4000 and above) address space would be directed to the external Program SRAM. When there comes the time that the software needs to read or write the 16KB above address space of the on-chip Flash memory, it needs to set this bit to 1 to be able to gain access to the on-chip Flash memory instead of the external Program SRAM. When enabled, the access of external Program SRAM is temporarily disabled. 0: To disable Flash memory access during enabling program code shadow mode. When program shadow mode is enabled, software accessing of 16KB above (0x00_4000 and above) address space would be the accessing of the external Program SRAM (default).
4	PMS	RO	Program Memory Shadow. This bit reflects the current setting of input pin EXT_PROG_SRAM_EN. 1: Program shadow mode is enabled, i.e., the program code is run on Program SRAM. 0: Program non-shadow mode, the program code is run on Flash memory.
5	ICD	RO	I2C Configuration EEPROM is Disabled during boot-up. This bit reflects the current setting of the input pin I2C_BOOT_DISABLE. 1: I2C Configuration EEPROM is disabled during boot, meaning that the I2C controller has not loaded configuration data from I2C EEPROM during reset. 0: I2C Configuration EEPROM is enabled during boot, meaning that the I2C controller has loaded the configuration data from I2C EEPROM during reset.
7:6	SCS [1:0]	RO	System Clock Select. These bits reflect the current setting of input pin SYSCK_SEL[1:0], which configures the operating system clock frequency. 11: 100 MHz 10: Reserved for test mode 01: 50 MHz 00: 25 MHz

<table>
<tr><th>BLOCK_NUMBER, On-chip Flash Memory Location 0x003FFF, Bit [6:0]</th><th>EXT_PROG_SRAM_EN pin setting</th><th>Program SRAM</th><th>Bootloader Loading Time (typical)</th></tr>
<tr><td>Any value</td><td>0 = Shadow mode disabled</td><td>The on-chip Flash memory’s lower 16K bytes program code is mirrored to on-chip 16K bytes Program SRAM.

Any external SRAM chips will be used for xDATA memory expansion purpose only.</td><td><table><tr><th>System Clock</th><th>Time</th></tr><tr><td>100Mhz</td><td>4.1 ms</td></tr><tr><td>50Mhz</td><td>4.3 ms</td></tr><tr><td>25Mhz</td><td>4.9 ms</td></tr></table></td></tr>
<tr><td>000_0001 (0x01)</td><td>1 = shadow mode enabled and program code is less than 32K bytes</td><td>The on-chip Flash memory’s lower 16K byte program code is mirrored to on-chip 16K bytes Program SRAM.

The lower 32K bytes range of the external SRAM chip is used for program code shadow and will be loaded with on-chip Flash memory’s lower 32K bytes program code by Bootloader. The remaining portion of the external SRAM chip is used for xDATA memory expansion purpose.</td><td><table><tr><th>System Clock</th><th>Time</th></tr><tr><td>100Mhz</td><td>8.2 ms</td></tr><tr><td>50Mhz</td><td>8.5 ms</td></tr><tr><td>25Mhz</td><td>9.2 ms</td></tr></table></td></tr>
<tr><td>000_0010 (0x02)</td><td>1 = shadow mode enabled and program code is less than 64K bytes</td><td>The on-chip Flash memory’s lower 16K byte program code is mirrored to on-chip 16K bytes Program SRAM.

The lower 64K bytes range of the external SRAM chip is used for program code shadow and will be loaded with on-chip Flash memory’s lower 64K bytes program code by Bootloader. The remaining portion of the external SRAM chip is used for xDATA memory expansion purpose.</td><td>(8.2ms * 2) at 100Mhz

or (8.5ms * 2) at 50Mhz

or (9.2ms * 2) at 25Mhz</td></tr>
<tr><td>000_0011 (0x03)
000_0100 (0x04)
000_1111 (0x0F)</td><td>1 = shadow mode enabled and program code is less than 96K/128K/…/480K bytes</td><td>The on-chip Flash memory’s lower 16K byte program code is mirrored to on-chip 16K bytes Program SRAM.

The lower 96K/128K/…/480K bytes range of the external SRAM chip is used for program code shadow and will be loaded with on-chip Flash memory’s lower 96K/128K/…/480K bytes program code by Bootloader. The remaining portion of the external SRAM chip is used for xDATA memory expansion purpose.</td><td>(8.2ms * BLOCK_NUMBER) at 100Mhz

or (8.5ms * BLOCK_NUMBER) at 50Mhz

or (9.2ms * BLOCK_NUMBER) at 25Mhz</td></tr>
<tr><td>001_0000 (0x10)</td><td>1 = shadow mode enabled and program code is less than 512K bytes</td><td>The on-chip Flash memory’s lower 16K byte program code is mirrored to on-chip 16K bytes Program SRAM.

The lower 512K bytes range of the external SRAM chip is used for program code shadow and will be loaded with on-chip Flash memory’s lower 512K bytes program code by Bootloader. The remaining portion of the external SRAM chip is used for xDATA memory expansion purpose.</td><td><table><tr><th>System Clock</th><th>Time</th></tr><tr><td>100Mhz</td><td>131.2 ms</td></tr><tr><td>50Mhz</td><td>136 ms</td></tr><tr><td>25Mhz</td><td>147.2 ms</td></tr></table></td></tr>
</table>

Table 10: Block Number Setting for Program Shadow Mode

4.6.2 Memory Arbiter

During normal CPU core access operations, the Memory Arbiter manages the Program and xDATA memory bus access to the embedded program/data memory as well as External Memory Interface and their address conversion. During DMA access, the Memory Arbiter arbitrates the xDATA memory bus access between the CPU core and the DMA engine.

When DMA Engine receives DMA requests from TOE initiated DMA or software initiated DMA, it will cause Memory Arbiter to generate an interrupt request to INT2, notifying CPU and software that it needs the ownership of xDATA memory bus in order to perform DMA access on xDATA memory. Within the interrupt service routine (ISR) of INT2, the CPU and software can then grant the DMA request to DMA engine through SFR register DBAR as shown below. **Note that the interrupt service routine for INT2 for DMA request should always be stored within the internal Program SRAM region (0x00_0000 ~ 0x00_3FFF) to allow DMA access properly.**

DMA Bus Arbitration Register (DBAR, 0x9Ah)

Bit	7	6	5	4	3	2	1	0
Name	BUS_GR	Reserved						BUS_REQ
Reset Value	0	00						0

Bit	Name	Access	Description
0	BUS_REQ	RO	Bus Request. The Memory Arbiter will set this bit to “1” to request to switch the ownership of xDATA memory bus to DMA engine in order to perform DMA transfer. The types of event to trigger this bit being set include software DMA transfer, and Ethernet packet transmit/receive events. Upon DMA transfer completed, the Memory Arbiter will clear this bit automatically. Note: The interrupt service routine (ISR) of INT2 for DMA transfer in software should keep polling this bit and only after seeing this bit being cleared, then the ISR is allowed to exit.
6:1	Reserved	RO	
7	BUS_GR	W1/R	Bus Grant. The CPU or software sets this bit to 1 to grant the ownership of xDATA memory bus to DMA engine allowing the DMA transfer to start. Upon DMA transfer completed, the Memory Arbiter will clear this bit automatically.

Note:

1. The interrupt for initiating DMA transfer is being assigned to INT2. Software should set INT2 to high priority to avoid other interrupt sources to intercept the DMA transfer in progress. When the CPU is servicing a high priority interrupt within an ISR while another high priority interrupt is occurring, the CPU will continue servicing the pending ISR until finished.

4.6.3 Program and Data Memory Address Mapping

Figure 39 shows the program memory map when no external program SRAM is used (EXT_PROG_SRAM_EN = 0 in Program Non-Shadow Mode). On left-hand side, it represents the CPU's external program memory address space, where 0x00_0000 ~ 0x07_FFFF address space is reserved for on-chip Flash memory and 0x08_0000 ~ 0x1F_FFFF address space is used by the optional external Flash memory. On right-hand side, it represents the Flash memory's physical address space, where 0x00_0000 ~ 0x07_FFFF is seen on the on-chip Flash and 0x00_0000 ~ 0xF7_FFFF is seen on the optional external Flash memory.

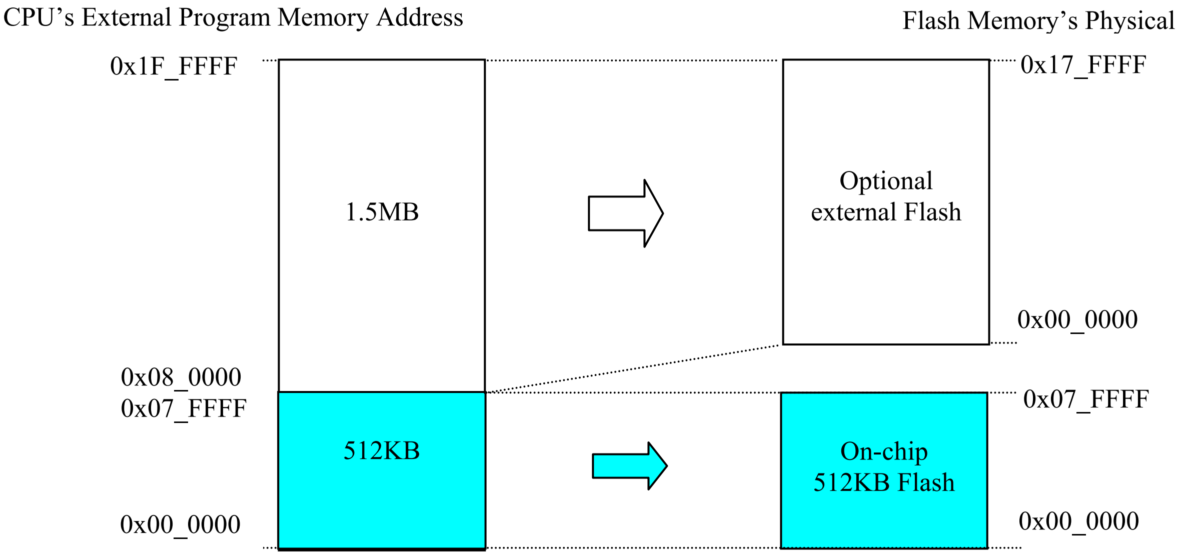


Figure 39: CPU Program Memory Map (EXT_PROG_SRAM_EN = 0)

Figure 40 shows when external Program SRAM chips are being used for program code shadow (EXT_PROG_SRAM_EN = 1 in Program Shadow Mode), the program code size is more than 512KB (BLOCK_NUMBER > 0x10), and there are two external SRAM being used.

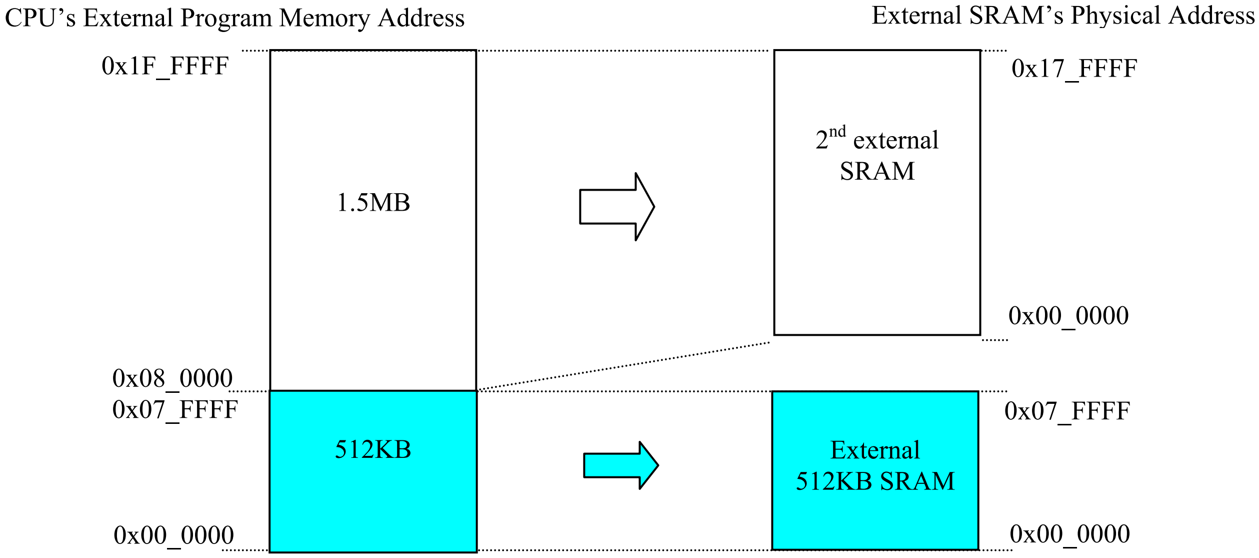


Figure 40: CPU Program Memory Map (EXT_PROG_SRAM_EN = 1)

Figure 41 shows when an external SRAM is being used for both program code shadow (EXT_PROG_SRAM_EN = 1 in Program Shadow Mode) and xDATA memory expansion. The program code size is 256KB as an example (BLOCK_NUMBER = 0x08) and the xDATA memory expansion is also 256KB.

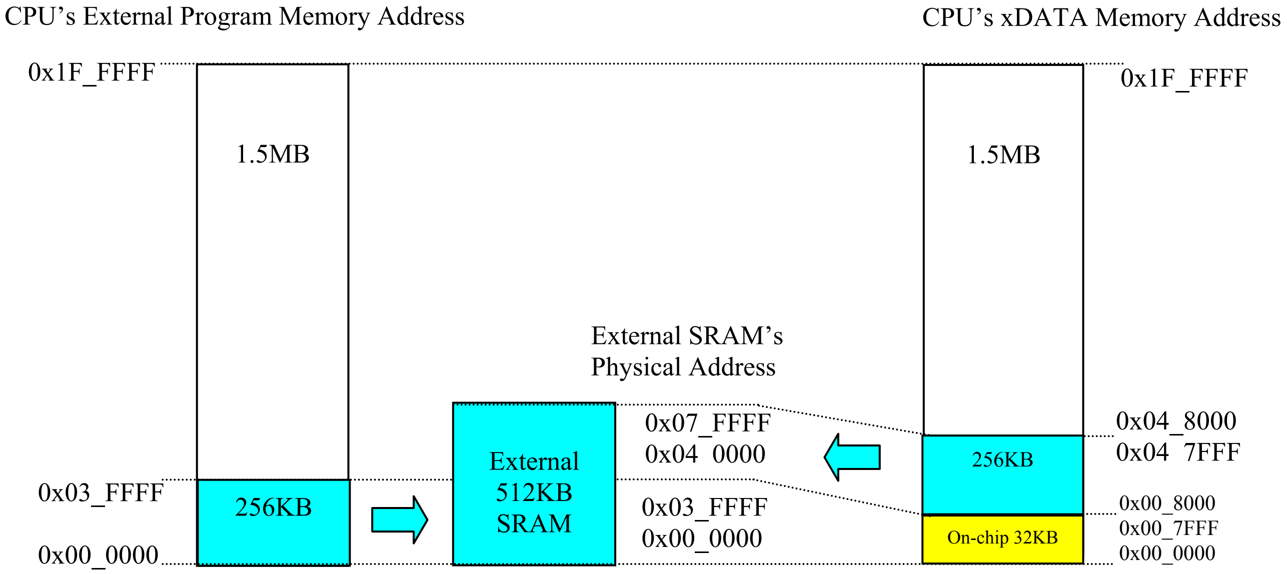


Figure 41: CPU Program Memory and xDATA Memory Map (EXT_PROG_SRAM_EN = 1)

Figure 42 shows when external SRAM chips are being used for both program code shadow (EXT_PROG_SRAM_EN = 1 in Program Shadow Mode) and xDATA memory expansion. The program code size is 512KB as an example (BLOCK_NUMBER = 0x10) and the xDATA memory expansion is also 512KB.

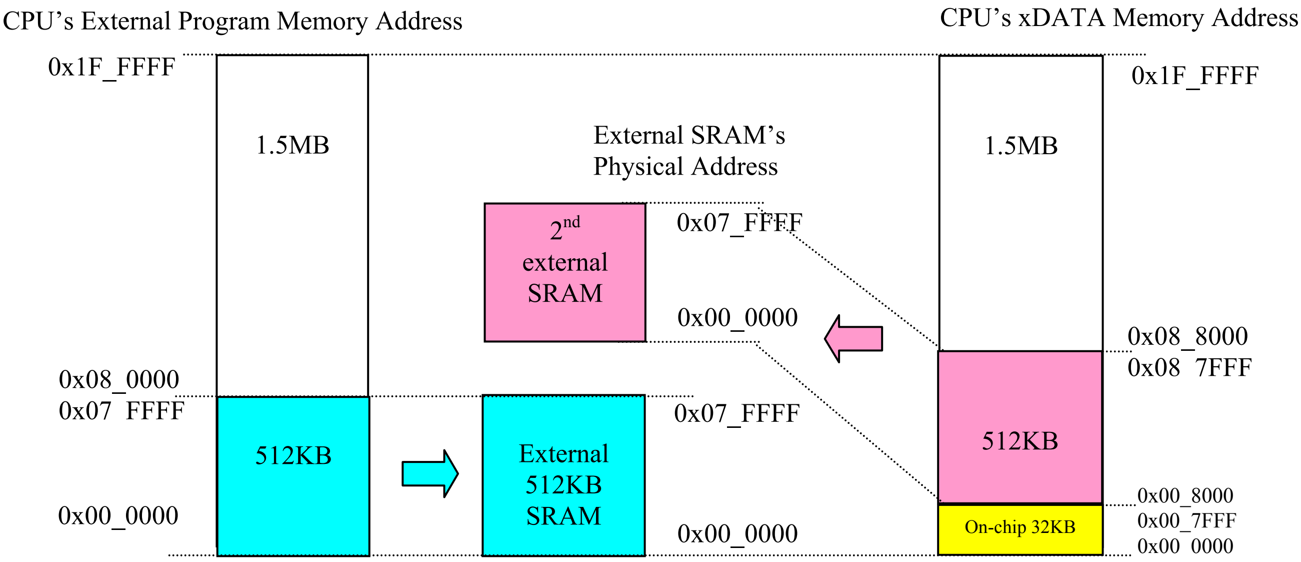


Figure 42: CPU Program Memory and xDATA Memory Map (EXT_PROG_SRAM_EN = 1)

Figure 43 shows when an external SRAM is used for xDATA memory expansion (EXT_PROG_SRAM_EN = 0 in Program Non-Shadow Mode). The program code size is 512KB or less and the xDATA memory expansion is also 512KB.

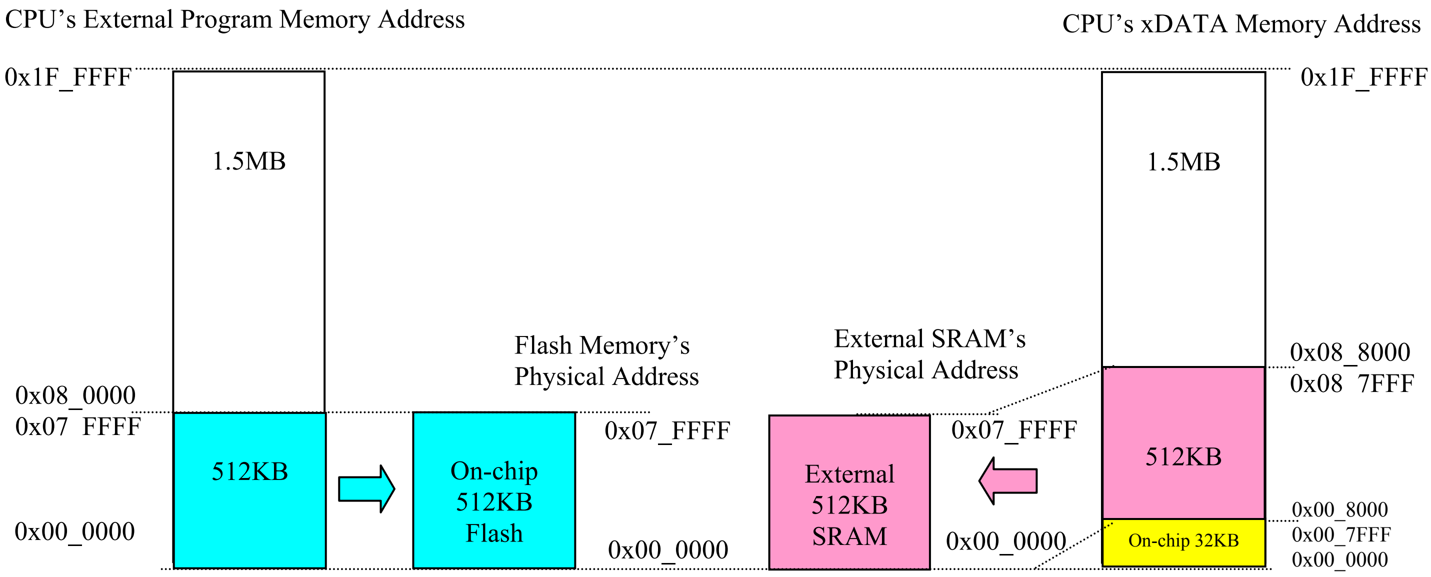


Figure 43: CPU Program Memory and xDATA Memory Map (EXT_PROG_SRAM_EN = 0)

4.6.4 Flash Memory Address Re-mapping for the Lower 16KB Boot Sector

The lower 16KB (0x00_0000 to 0x00_3FFF) of CPU's program memory space is being defined as ROM space to the CPU and the AX11025 embeds a 16KB internal Program SRAM for storing program code of that address space. By default, the CPU program read or program write access to that 16KB space would be accessing the 16KB internal Program SRAM. When there comes the time that the software needs to read or write the lower 16KB (0x00_0000 to 0x00_3FFF) of the on-chip Flash memory, it needs a **Flash memory Address Re-mapping mechanism** to be able to gain access to the on-chip Flash memory instead of the on-chip 16KB Program SRAM.

Following section describes proper software procedures to perform read/write access to the lower 16KB of on-chip Flash memory.

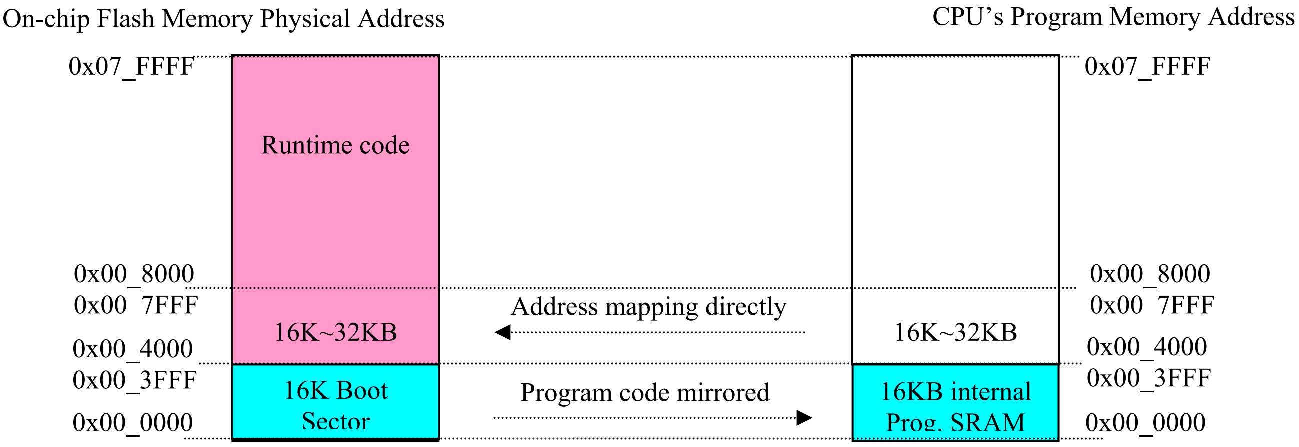


Figure 44: Flash Memory Address Without Re-Mapping, SFR register CSREPR [FARM] = 0 (default)

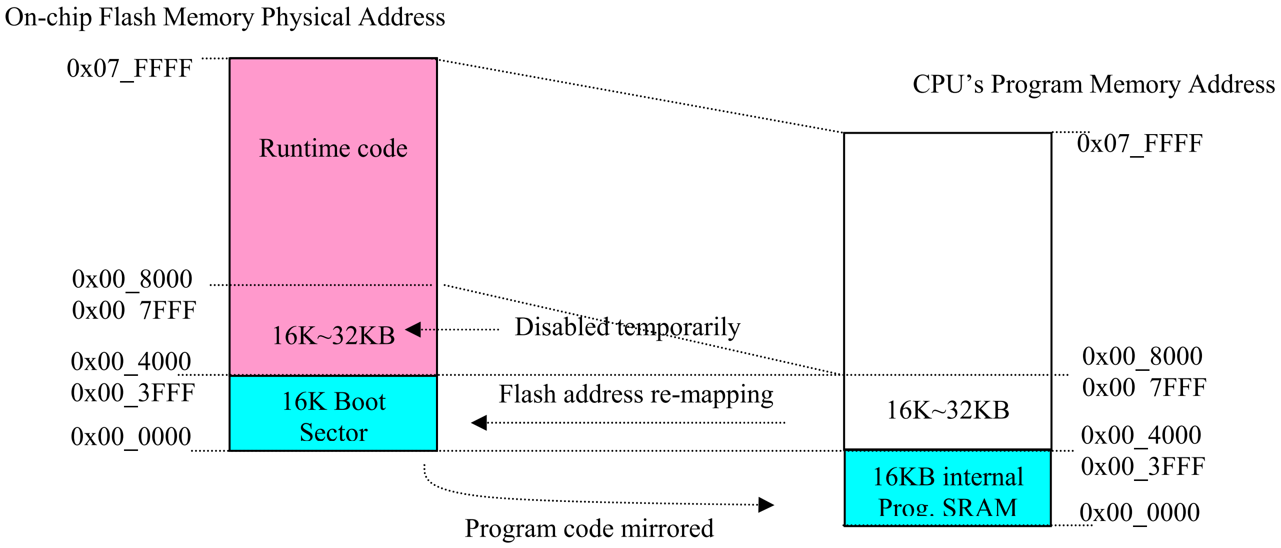


Figure 45: Flash Memory Address Re-Mapping Enabled, SFR register CSREPR [FARM] = 1

Case 1: Programming procedure to read the lower 16KB of on-chip Flash memory (only applies to Program Non-Shadow mode, EXT_PROG_SRAM_EN = 0)

Step 1: Software first writes FARM bit = 1 (CSREPR.2, 0x8F), to enable Flash memory address re-mapping mechanism.

Step 2: Reading any program address space 16K~32K (0x00_4000~0x007FFF) will be remapped to the 0K~16K (0x00_0000~0x00_3FFF) address space of the Flash memory. Therefore, the memory content of lower 16KB of the on-chip Flash memory can now be accessed. Note that the accessing of the 16KB~32KB address space of on-chip Flash memory is temporarily disabled after step 1.

Step 3: After done with accessing the lower 16KB of on-chip Flash memory, the software then writes FARM bit = 0 (CSREPR.2), to disable Flash memory address re-mapping mechanism. Now the access of program address space 16KB~32KB would revert back to the same address space of the on-chip Flash memory as usual, and the accessing of 0K~16K (0x00_0000~0x00_3FFF) address space of the Flash memory is disabled again.

Case 2: Programming procedure to write the lower 16KB of on-chip Flash memory (only applies to Program Non-Shadow mode, EXT_PROG_SRAM_EN = 0)

Step 1: Software first writes PWE bit = 1 (PCON.4, 0x87), to enable program write in CPU.

Step 2: Software writes FARM bit = 1 (CSREPR.2, 0x8F), to enable Flash memory address re-mapping mechanism. Now accessing the 0K~16K (0x00_0000~0x00_3FFF) address space of the Flash memory would come from software accessing the program address space 16K~32K (0x00_4000~0x007FFF).

Step 3: Programming sequence for performing "Sector Erase" commands for on-Flash memory:

1. Write (0x4000 + 0x555) = 0xAA
2. Write (0x4000 + 0x2AA) = 0x55
3. Write (0x4000 + 0x555) = 0x80
4. Write (0x4000 + 0x555) = 0xAA
5. Write (0x4000 + 0x2AA)= 0x55
6. Write (0x4000 + 0x000) = 0x30
7. Repeatedly read (0x4000 + 0x3FFF). If equal to 0xFF, the Sector Erase command is completed.

Step 4: Programming sequence for performing "Byte Program" commands for on-Flash memory:

1. Write (0x4000 + 0x555) = 0xAA
2. Write (0x4000 + 0x2AA) = 0x55
3. Write (0x4000 + 0x555) = 0xA0
4. Write (0x4000 + PA) = PD (where PA: any 0~16KB of on-chip Flash memory address to be programmed, PD: write data)
5. Repeatedly read (0x4000 + PA). If equal to PD, then the "Byte Program" command is completed.

Step 5: Software writes FARM bit = 0 (CSREPR.2), to disable Flash memory address re-mapping.

Step 6: Software writes PWE bit = 0 (PCON.4), to disable program write in CPU.

Case 3: Programming procedure to write 16KB above address space of on-chip Flash memory (only applies to Program Non-Shadow mode, EXT_PROG_SRAM_EN = 0)

Step 1: Software first writes PWE bit = 1 (PCON.4, 0x87), to enable program write in CPU.

Step 2: Software writes FARM bit = 1 (CSREPR.2, 0x8F), to enable Flash memory address re-mapping mechanism.

Step 3: Programming sequence for enabling "Byte Program" command for on-Flash memory:

1. Write (0x4000 + 0x555) = 0xAA
2. Write (0x4000 + 0x2AA) = 0x55
3. Write (0x4000 + 0x555) = 0xA0

Step 4: Software writes FARM bit = 0 (CSREPR.2), to disable Flash memory address re-mapping mechanism.

Step 5: Now perform the actual byte write command.

1. Write PA = PD (where PA: any 16KB above Flash memory address to be programmed, PD: write data)
2. Repeatedly read PA. If equal to PD, then the Program command is completed.

Step 6: Software writes PWE bit = 0 (PCON.4), to disable program write in CPU.

4.6.5 Flash Memory Access in Program Code Shadow Mode

When an external program SRAM is being used for program code shadow purpose, in that case, the CPU program read or program write access to 16KB above (0x00_4000 and above) address space would be the accessing of the external SRAM. When there comes the time that the software needs to read or write the 16KB above address space of the on-chip Flash memory, it needs a **Flash memory Address Un-shadowed mechanism** to be able to gain access to the on-chip Flash memory instead of the external SRAM.

Case 1: Programming procedure to read the lower 16KB of on-chip Flash memory (only applies to shadow mode, EXT_PROG_SRAM_EN = 1)

Same as the case 1 in Program Non-Shadow mode, except that in step 1 the software first changes WTST to Non-Shadow Mode value, then writes FARM bit = 1 (CSREPR.2) and FAES bit = 1 (CSREPR.3), to temporarily enable Flash memory access and disable the external Program SRAM access. When done, in step 3 the software should clear both FARM and FAES bits in CSREPR register, and then revert the WTST back to Program Shadow Mode value.

Case 2: Programming procedure to write the lower 16KB of on-chip Flash memory (only applies to shadow mode, EXT_PROG_SRAM_EN = 1)

Same as the case 2 in Program Non-Shadow mode, except that in step 2, the software first changes WTST to Non-Shadow Mode value, then writes FARM bit = 1 (CSREPR.2) and FAES bit = 1 (CSREPR.3), to temporarily enable Flash memory access and disable the external Program SRAM access. When done, in step 5 the software should clear both FARM and FAES bits in CSREPR register, and then revert the WTST back to Program Shadow Mode value.

Case 3: Programming procedure to write 16KB above address space of on-chip Flash memory (only applies to shadow mode, EXT_PROG_SRAM_EN = 1)

Step 1: Software first writes PWE bit = 1 (PCON.4), to enable program write in CPU.

Step 2: Software changes WTST to Non-Shadow Mode value, writes FARM bit = 1 (CSREPR.2) and FAES bit = 1 (CSREPR.3), to enable Flash memory address re-mapping mechanism, and temporarily enable Flash memory access and disable the external Program SRAM access

Step 3: Programming sequence for enabling "Byte Program" command for on-Flash memory:

1. Write (0x4000 + 0x555) = 0xAA
2. Write (0x4000 + 0x2AA) = 0x55
3. Write (0x4000 + 0x555) = 0xA0

Step 4: Software writes FARM bit = 0 (CSREPR.2), to disable Flash memory address re-mapping mechanism.

Step 5: Now perform the actual byte write command.

1. Write PA = PD (where PA: any 16KB above of Flash memory address to be programmed, PD: write data)
2. Repeatedly read PA. If equal to PD, the Program command is completed.

Step 6: Software writes FAES bit = 0 (CSREPR.3), to disable Flash memory access and re-enable the external Program SRAM access in program code shadow mode, and then revert the WTST back to Program Shadow Mode value.

Step 7: Software writes PWE bit = 0 (PCON.4), to disable program write in CPU.

4.6.6 Flash Programming Controller

When asserting chip reset (via RST_N pin) to AX11025 and also asserting "BURN_FLASH_EN" pin to high, the AX11025 will enter into Flash memory programming enabled mode. Upon enabled, the Flash programming controller can start receiving command packets from Flash Programming utilities through the RXD0 pin of UART0, decoding the packets, passing the decoded command to perform on-chip and off-chip Flash memory erase/programming tasks, and then returning the acknowledgement packets with the result back to Flash Programming utilities running on a PC. The Flash programming controller is responsible for generating the waveform for Flash memory access. The Flash programming speed via UART0 can support two speeds, 115.2 Kbps (BURN_FLASH_912K pin set to low) and 921.6Kbps (BURN_FLASH_912K pin set to high).

Note that during Flash programming enabled mode, the internal CPU core is not running. Therefore, after Flash programming is completed, another chip reset (via RST_N pin) should be applied to AX11025 without asserting "BURN_FLASH_EN" pin to allow CPU to start running normally.

4.7 DMA Engine

As shown in figure below, the DMA engine supports direct External Data (xDATA) Memory read and write access without CPU intervention for the TCP/IP Offload Engine (TOE) as well as bulk data copy for software DMA.

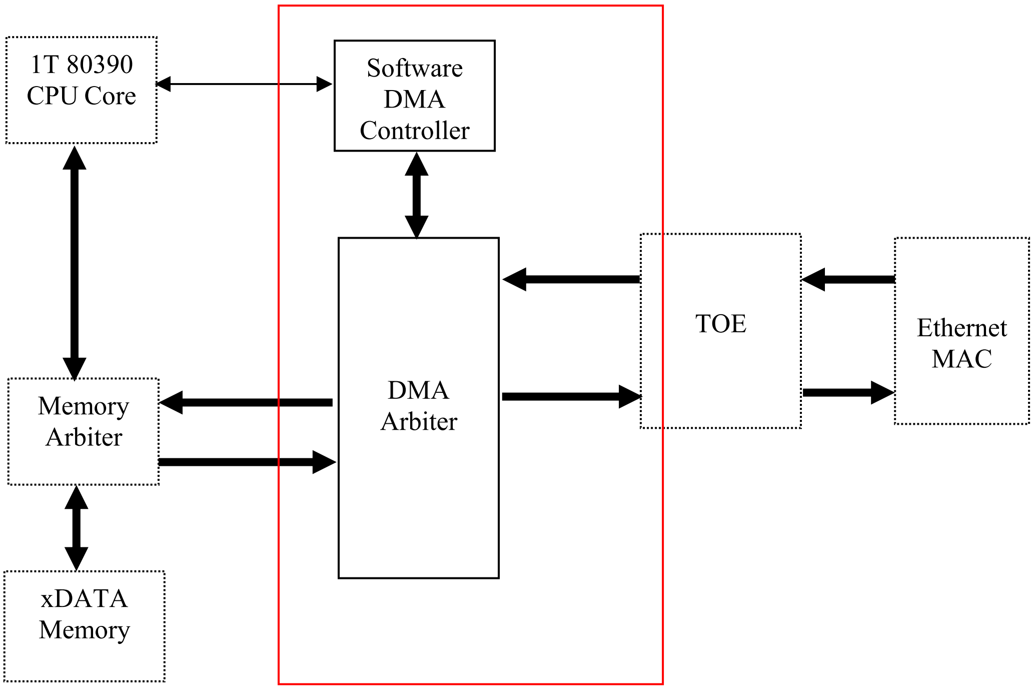


Figure 46: DMA Engine Block Diagram

4.7.1 DMA Transfers for Ethernet Packet Receive and Transmit

Packet Receive

During normal Ethernet packet receive process, the Ethernet MAC shall forward the received packets from its receive buffer to TOE receive block which then moves and stores the received packets into xDATA memory via DMA write access. The received packets will be stored in “Receive Packet Buffer Ring” region of xDATA memory being defined by software during initialization. See section 4.14 for more detailed description. During this process, the DMA arbiter will receive DMA request from TOE receive block and then it will trigger the Memory Arbiter to generate an interrupt to CPU on INT2 to notify CPU the pending DMA request from TOE receive block and waiting for CPU to grant it. After CPU grants it, if the received packet size is more than 256 bytes, it will be executed in several transfers, with maximum of 256 bytes per transfer. For example, as shown in Figure 48, for a 1500 bytes Ethernet packet, it will take up to 6 DMA transfers to finish moving it into “Receive Packet Buffer Ring” region of xDATA memory. The gap between each transfer is programmable by TL4DGR register, see section 4.14 for more details. In DMA write access case, each byte will take (CKCON[2:0] +2) operating system clocks to write into xDATA memory, where CKCON is SFR register offset 0x8E.

Packet Transmit

In normal Ethernet packet transmit process, the software first prepares the to-be-transmitted packets and stores them in the "Transmit Packet Buffer Ring" region of xDATA memory being defined by software during initialization. The software then configures the TL4CMR [SP] bit to initiate the packet transmit process in TOE transmit block for moving packets to Ethernet MAC transmit buffer. Please Refer to section 4.14 for more details. Now the TOE transmit block will send DMA request to DMA arbiter which again will trigger the Memory Arbiter to generate an interrupt to CPU on INT2 to notify CPU the pending DMA request from TOE transmit block and waiting for CPU to grant it. After CPU grants it, if the to-be-transmitted packet size is more than 256 bytes, it will be executed in several transfers, with maximum of 256 bytes per transfer. For example, for a 1518 bytes Ethernet packet, it will take up to 6 DMA transfers to finish moving it out of "Transmit Packet Buffer Ring" region of xDATA memory. The gap between each transfer is programmable by TL4DGR register. During this DMA read access case, each byte will take (CKCON[2:0] +2) operating system clocks to read from xDATA memory, where CKCON is SFR register offset 0x8E.

4.7.2 Software DMA

The software DMA can perform bulk data copy from one region of xDATA memory to another region in a timely manner, based on software configuration. This hardware based software DMA controller can greatly reduce the time spending in bulk data movement very often needed in network protocol stack processing, and, hence, help achieve better performance on micro-controller computing power.

If software DMA transfer size is more 128 bytes, it will be executed in several transfers, with maximum of 128 bytes per transfer. For example, to copy 512 bytes data from one region to another region of xDATA memory, it will take 4 DMA transfers to complete. The gap between each transfer is programmable by TL4DGR register.

When software DMA transfer involves copying one Ethernet packet from the "Receive Packet Buffer Ring (RPBR)" to the software's application buffer area or from the software's application buffer area to the "Transmit Packet Buffer Ring (TPBR)", in that case, the software DMA controller has been designed with "ring-aware" architecture and can deal with the ring structure of RPBR and TPBR automatically, particularly, the buffer ring wrap-around issue.

For example, when copying a packet from RPBR to software's application buffer area, if the packet data happens to across the ring boundary, (i.e., the packet data starts at the last few pages of the ring and wraps around to the first few pages of the ring), the software DMA controller will automatically make sure the packet data is retrieved from the ring structure of RPBR without having the software to worry about the packet crossing RPBR ring boundary issue and having to perform this software DMA in two times to take care of the boundary issue.

The same applies to the case when the software needs to move an Ethernet packet from software's application buffer area into TBPR, and software doesn't have to worry about packet data crossing TPBR ring boundary.

However, please note that the software DMA controller do not have the knowledge of data structure of software's application buffer area, therefore, it cannot deal with the ring boundary crossing issue for software's application buffer area. It's software's responsibility to cover its buffer area wrap-around scenario when software initiates such DMA transfer. Because for non-RPBR and non-TPBR types of DMA transfers, the software DMA controller can only increase DMA source address or target address linearly after each byte transferred, therefore, it can not adjust these address for the ring type of data structure in software's application buffer area in xDATA memory.

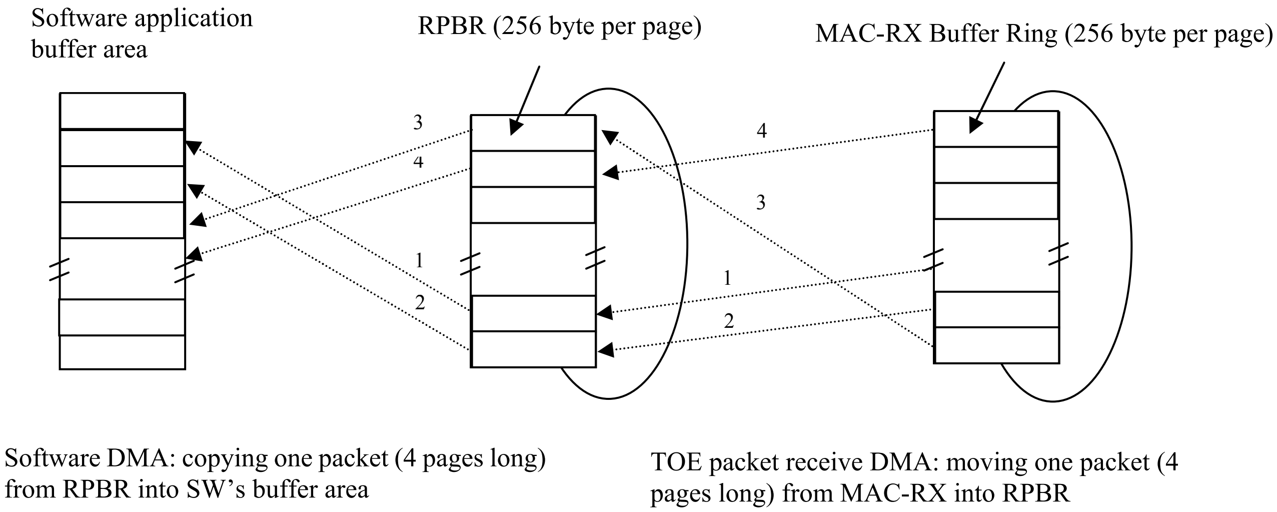

Software application
buffer area
RPBR (256 byte per page)
MAC-RX Buffer Ring (256 byte per page)
3
4
1
2
4
3
1
2
Software DMA: copying one packet (4 pages long) from RPBR into SW's buffer area
TOE packet receive DMA: moving one packet (4 pages long) from MAC-RX into RPBR

Figure 47: Ring-aware Software DMA Example

Software DMA and Millisecond Timer Related SFR Register Map

Address	Name	Description
0x9B	DCIR	DMA Command Index Register is used to indicate the address of to-be-accessed register listed in Table 12.
0x9C	DDR	DMA Data Register is used to read data from or write data to the specific register provided by DCIR.
0x94	SDSTSR	Software DMA and Millisecond Timer Status Register

Table 11: Software DMA and Millisecond Timer Related SFR Register Map

DMA Command Index Register (DCIR, 0x9B)

Bit	7	6	5	4	3	2	1	0
Name	RI							
Reset Value	0x00							

<table>
<tr><th>Bit</th><th>Name</th><th>Access</th><th>Description</th></tr>
<tr><td>[7:0]</td><td>RI</td><td>WO</td><td>Register Index.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x00~0x0f</td><td>Indicate to access which of the Software DMA and Millisecond Timer registers listed in Table 12.</td></tr>
<tr><td>0xff</td><td>Command Abort</td></tr>
</table></td></tr>
</table>

DMA Data Register (DDR, 0x9C)

Bit	7	6	5	4	3	2	1	0
Name	DR							
Reset Value	0x00							

Bit	Name	Access	Description
[7:0]	DR	R/W	Data Register is used to write data to or read data from Software DMA and Millisecond

			Timer registers.

Software DMA and Millisecond Timer Status Register (SDSTSR, 0x94)

Bit	7	6	5	4	3	2	1	0
Name	Reserved						STT	SDC
Reset Value	0x00							

Bit	Name	Access	Description
0	SDC	CR	The Software DMA transfer is Completed. When reading “1”, this bit indicates that the software DMA transfer requested via SDCSR has been completed.
1	STT	CR	The Millisecond Timer has Timed out. When reading “1”, this bit indicates that the Millisecond Timer has reached the timeout value being set in MSTR register.
7:2	Reserved		

Software DMA and Millisecond Timer Register Indirect Access Method

Software shall use indirect access method through DCIR and DDR registers to read or write the Software DMA and Millisecond Timer register listed in Table 12 below.

Read a register from Software DMA and Millisecond Timer:

Step 1. Write DCIR: Software indicates the Software DMA or Millisecond Timer register address to be accessed as the data and write it to the SFR register DCIR.

Step 2. Read DDR: Software then read SFR register DDR. The data read from DDR is the Software DMA or Millisecond Timer register data indicated in step 1. Keep reading from DDR if the Software DMA or Millisecond Timer registers have more than one byte, in that case, the first byte being read back is LSB byte.

Write a register to Software DMA and Millisecond Timer:

Step 1. Write DDR: Software writes the data you want to write into Software DMA or Millisecond Timer registers to the SFR register DDR. Keep writing to DDR if the Software DMA or Millisecond Timer registers have more than one byte, in that case, the first byte being written should be LSB byte.

Step 2. Write DCIR: After writing Software DMA or Millisecond Timer register data to DDR, software then indicates the target Software DMA or Millisecond Timer register address as data and write it to DCIR.

Software DMA and Millisecond Timer Register Map

Address	Register Name	Description
0x00	SDCSR	Software DMA Command Status Register
0x02	SDSSAR	Software DMA Source Starting Address Register (24 bits)
0x06	SDTSAR	Software DMA Target Starting Address Register (24 bits)
0x0A	SDBCR	Software DMA Byte Count Register (16 bits)
0x0C	MSTR	Millisecond Timer Register (10 bits)

Table 12: Software DMA and Millisecond Timer Register Map

Software DMA Command Status Register (SDCSR, 0x00)

Bit	7	6	5	4	3	2	1	0
Name	EI_SDC	Reserved	TAIT	SAIR	Reserved	DMAERR	FS	GO
Reset Value	00		0	0	0	0	0	0

Bit	Name	Access	Description
0	GO	W1/R	Software sets GO bit to "1" to initiates the "software DMA transfer" which facilitates copying a block of data from the specified range of xDATA Memory to another specified range of xDATA Memory. This bit will remain "1" while the DMA transfer is still in progress and will be cleared automatically after the requested DMA transfer is completed or stopped by software via FS bit. Note: Software can only write "1" to this bit and can't write "0".
1	FS	W1/R	Force to Stop the software DMA transfer in progress. This bit will remain "1" while the software DMA controller is trying to stop the DMA transfer and will be cleared automatically after software DMA controller is completely stopped.
2	DMAERR	CR	DMA Error indication. When reading back a "1", it indicates that the requested software DMA has encountered error and can't be finished. The condition that causes DMA error could be that, for example: - The value of SDBCR register = 0 byte - Or the value of target memory address, SDTSAR register is equal to the value of source memory address, SDSSAR register - Or the memory address range of target (SDTSAR + SDBCR) is overlapping with the memory address range of the source (SDSSAR + SDBCR). - If SAIR bit is set and SDSSAR is not in the range of RSPP and REPP. - If TAIT bit is set and SDTSAR is not in the range of TSPP and TEPP.
3	Reserved		
4	SAIR	W1/R	Source Address Is in RPBR. Software sets SAIR to "1" at the same time as setting GO bit to "1" to indicate that the requested DMA involves copying packet data from RPBR to the target location. This way the software DMA controller will base on the ring structure of RPBR to locate the source data and will wrap around the ring of RPBR when the last page of the ring is hit. This bit will remain "1" while the DMA transfer is still in progress and will be cleared automatically after the requested DMA transfer is completed or stopped by software via FS bit.
5	TAIT	W1/R	Target Address Is in TPBR. Software sets TAIT to "1" at the same time as setting GO bit to "1" to indicate that the requested DMA involves copying packet data from source location to TPBR. This way the software DMA controller will base on the ring structure of TPBR to write the data and will wrap around the ring of TPBR when the last page of the ring is hit. This bit will remain "1" while the DMA transfer is still in progress and will be cleared automatically after the requested DMA transfer is completed or stopped by software via FS bit.
6	Reserved		
7	EI_SDC	R/W	Enable Interrupt whenever the requested Software DMA is Completed. The interrupt is asserted on INT 5.

Software DMA Source Starting Address Register (SDSSAR, 0x02)

Bit	7	6	5	4	3	2	1	0
Name	S_ADDR 0							
	S_ADDR 1							
	S_ADDR 2							
Reset Value	0x00_0000							

Bit	Name	Access	Description
7:0 15:8	S_ADDR 0 S_ADDR 1	R/W	The Source starting Address for software DMA transfer. This is the 24-bit starting address of the source memory block to be copied from in CPU's xDATA Memory.

23:16	S_ADDR 2		

Software DMA Target Starting Address Register (SDTSAR, 0x06)

Bit	7	6	5	4	3	2	1	0
Name	T_ADDR 0							
	T_ADDR 1							
	T_ADDR 2							
Reset Value	0x00_0000							

Bit	Name	Access	Description
7:0 15:8 23:16	T_ADDR 0 T_ADDR 1 T_ADDR 2	R/W	The Target starting Address for software DMA transfer. This is the 24-bit starting address of target memory block to be copied to in CPU’s xDATA Memory.

Software DMA Byte Count Register (SDBCR, 0x0A)

Bit	7	6	5	4	3	2	1	0
Name	B_CNT 0							
	B_CNT 1							
Reset Value	0x0000							

Bit	Name	Access	Description
7:0 15:8	B_CNT 0 B_CNT 1	R/W	The block of data in terms of bytes the software DMA transfer is to be copied from source memory address to target memory address. Note that if the byte count is greater than 128 bytes, then the software DMA transfer will be executed in separate transfer with 128 bytes per transfer until all the requested bytes are copied to the target memory block.

Millisecond Timer Register (MSTR, 0x0C)

Bit	7	6	5	4	3	2	1	0
Name	MS_TMR 0							
	EI_STT	Reserved	RS_TMR	ST_TMR	Reserved		MS_TMR 1	
Reset Value	0x0001							

Bit	Name	Access	Description
7:0 9:8	MS_TMR 0 MS_TMR 1	R/W	Millisecond Timer Timeout value. Each count is about 1 msec in time. For example, 0x001 = 1 msec. 0x002 = 2 msec. The maximum timeout is 1024 msec. Whenever the Million-Second Timer reaches the timeout value being set here, the timer will reset to 0 to restart the timer all over again. And if the EI_STT register is enabled, it will also generate an interrupt on INT5 to CPU.
11:10	Reserved		
12	ST_TMR	R/W	Setting the ST_TMR bit to “1” to enable the Millisecond Timer to start counting.
13	RS_TMR	R/W1	Setting the RS_TMR bit to “1”to reset the Millisecond Timer to 0 and this bit will then be cleared to “0” by hardware automatically.
14	Reserved	R/W	
15	EI_STT	R/W	Enable Interrupt whenever the Millisecond Timer reaches the timeout value being set in MS_TMR 1,0 register. The interrupt is asserted on INT 5.

Software DMA Programming Procedure

Software needs to use indirect access procedure to read or write the specific Software DMA or Millisecond Timer registers through SFR registers, DCIR (0x9B) and DDR (0x9C). Following describes how to initiate a Software DMA transfer.

1. Software first writes to DDR register with data of S_ADDR 0 for SDSSAR.
2. Software writes to DDR register with data of S_ADDR 1 for SDSSAR.
3. Software writes to DDR register with data of S_ADDR 2 for SDSSAR.
4. Software writes to DCIR register with data of 0x02 (the address of SDSSAR)

----

5. Software writes to DDR register with data of T_ADDR 0 for SDTSAR.
6. Software writes to DDR register with data of T_ADDR 1 for SDTSAR.
7. Software writes to DDR register with data of T_ADDR 2 for SDTSAR.
8. Software writes to DCIR register with data of 0x06 (the address of SDTSAR)

----

9. Software writes to DDR register with data of B_CNT 0 for SDBCR.
10. Software writes to DDR register with data of B_CNT 1 for SDBCR.
11. Software writes to DCIR register with data of 0x0A (the address of SDBCR)

----

12. Software then writes to DDR register with data of SDCSR (e.g. set GO = 1, FS = 0).
13. Software then writes to DCIR register with data of 0x00 (the address of SDCSR)

----

14. Now software can wait for a while or wait for the interrupt to verify if the requested software DMA transfer is completed by software DMA controller or not.
15. Software first writes to DCIR register with data of 0x00 (the address of SDCSR).
16. Software then reads from DDR register with data of SDCSR. If the GO bit is still “1”, then the DMA operation is still in progress. Until software reads “0” on GO bit, it indicates that the DMA operation is completed.

----

17. If software reads back a “1” on DMAERR bit, that means that an error has occurred during the software DMA transfer. At this point, the software has to force to stop the unfinished DMA by setting the FS bit to “1” in order to clear the GO bit.
18. Software then writes to DDR register with data of SDCSR (e.g. set FS = 1).
19. Software then writes to DCIR register with data of 0x00 (the address of SDCSR)

4.7.3 DMA Arbitration

The DMA arbiter arbitrates the simultaneous DMA requests that come from Ethernet packet receive, Ethernet packet transmit, and software DMA.

If TOE and software DMA both request DMA transfers at the same time, the software’s DMA request will be granted first. The software DMA will always take higher priority over DMA transfers for Ethernet packet transmit/receive, which means it can intercept the later two cases when all three DMA requests are pending at the same. The arbitration rules are as follows:

- Software DMA transfer has priority over DMA transfers for Ethernet packet receive and packet transmit.

- Between DMA transfer for packet receive and packet transmit, it is round-robin fashion.

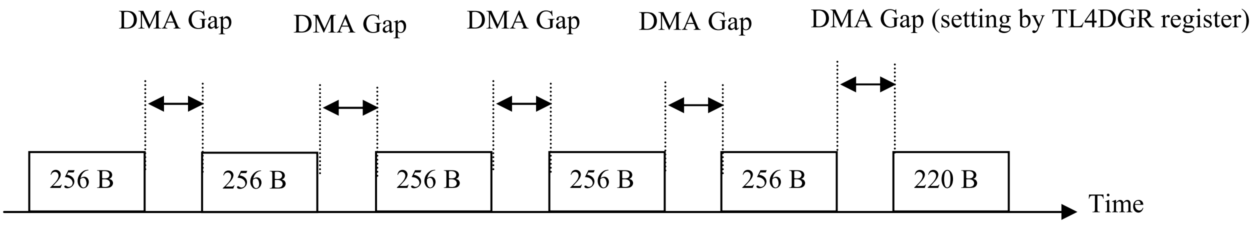


Figure 48: Example: Ethernet Packet Receive DMA Transfer Only (receiving a 1500-byte packet)

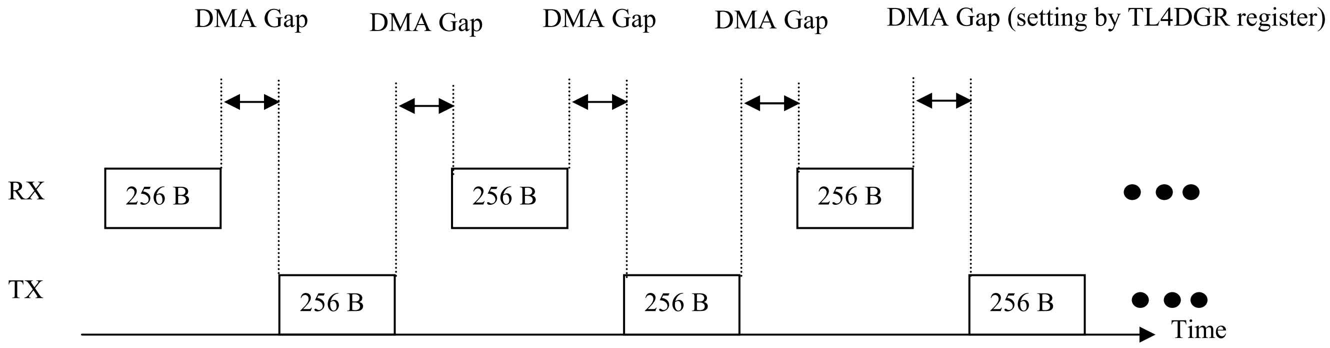


Figure 49: Example: Ethernet Packet Receive and Transmit DMA Transfers Simultaneously (receiving and transmitting a 1500-byte packet)

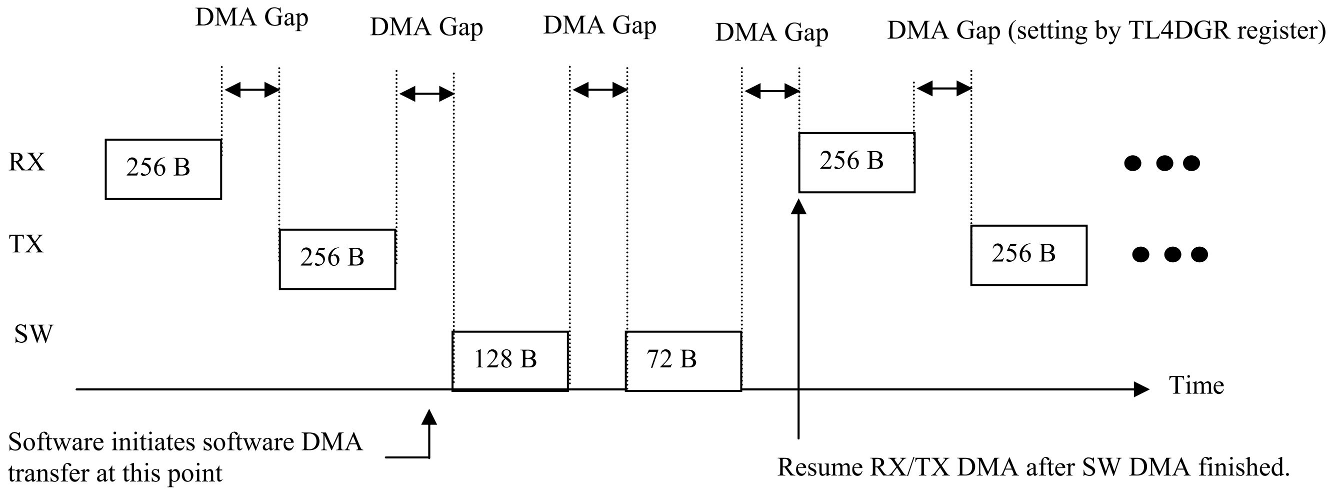

DMA Gap
DMA Gap
DMA Gap
DMA Gap
DMA Gap (setting by TL4DGR register)
RX
TX
SW
256 B
256 B
128 B
72 B
256 B
256 B
Time
Software initiates software DMA transfer at this point
Resume RX/TX DMA after SW DMA finished.

Figure 50: Example: Ethernet Packet Receive and Transmit and software DMA Transfers Simultaneously (receiving and transmitting a 1500-byte packet, and software copying a 200-bytes data block)

4.8 Interrupt Controller

The interrupt controller of AX11025 supports 2 external interrupt pins, INT0 and INT1, each having two levels of interrupt priority control. They can be in high or low-level priority group (set via SFR register IP and EIP). The INT0 and INT1 external interrupt pins can be either low-level trigger or falling-edge trigger. Also, the interrupt controller supports various interrupt requests internal to the AX11025, again each having two levels of interrupt priority control.

The interrupts flag summary is as shown in Table 13 below. Each interrupt vector can be individually enabled or disabled by setting or clearing a corresponding bit in the SFR register IE (0xA8) and EIE (0xE8). The IE contains global interrupt system disable/enable bit called EA bit (IE.7), which has to be set in order to enable individual interrupt requests listed in Table 13.

Interrupt Flag	Function	Active (level/edge)	Flag resets	Vector	Natural priority
IE0	The external interrupt input pin, INT0	Low/Falling	Hardware	0x03	1
TF0	The internal Timer 0 interrupt request	-	Hardware	0x0B	2
IE1	The external interrupt input pin, INT1	Low/Falling	Hardware	0x13	3
TF1	The internal Timer 1 interrupt request	-	Hardware	0x1B	4
TI0 & RI0	The internal UART 0 interrupt request	-	Software	0x23	5
TF2	The internal Timer 2 interrupt request	-	Software	0x2B	6
TI1 & RI1	The internal UART 1 interrupt request	-	Software	0x33	7
INT2F	The internal DMA transfer interrupt request for TOE/software DMA mode, please set to high priority	-	Hardware	0x3B	8
INT3F	The internal programmable counter array interrupt request	-	Hardware	0x43	9
INT4F	The internal peripheral interrupt request for TOE, MAC/PHY, I2C, SPI, 1-Wire, UART2, CAN, etc.	-	Hardware	0x4B	10
INT5F	The internal software DMA complete and millisecond timer timeout interrupt	-	Software	0x53	11
INT6F	The wake-up interrupt request (resume from CPU STOP mode)	-	Software	0x5B	12
WDIF	Internal watchdog interrupt	-	Software	0x63	13

Table 13: Interrupts Flag Summary

4.8.1 Interrupt Controller SFR Register Map

Address	Name	Description
0xA8	IE	Interrupt Enable Register
0xB8	IP	Interrupt Priority Register
0x88	TCON	Timer 0,1 Configuration Register
0x98	SCON0	UART 0 Configuration Register
0xC0	SCON1	UART 1 Configuration Register
0xE8	EIE	Extended Interrupt Enable Register
0xF8	EIP	Extended Interrupt Priority Register
0x91	EIF	Extended interrupt Flag Register
0x9E	PISS1R	Peripheral Interrupt Status Summary 1 Register
0x9F	PISS2R	Peripheral Interrupt Status Summary 2 Register

Table 14: Interrupt Controller SFR Register Map

Interrupt Enable Register (IE, 0xA8)

Bit	7	6	5	4	3	2	1	0
Name	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0
Reset Value	0x00							

Bit	Name	Access	Description
0	EX0	R/W	Enable INT0 interrupt. 1: Enabled. 0: Disabled.
1	ET0	R/W	Enable Timer 0 interrupt. 1: Enabled. 0: Disabled.
2	EX1	R/W	Enable INT1 interrupt. 1: Enabled. 0: Disabled.
3	ET1	R/W	Enable Timer 1 interrupt. 1: Enabled. 0: Disabled.
4	ES0	R/W	Enable UART0 interrupt. 1: Enabled. 0: Disabled.
5	ET2	R/W	Enable Timer 2 interrupt. 1: Enabled. 0: Disabled.
6	ES1	R/W	Enable UART1 interrupt. 1: Enabled. 0: Disabled.
7	EA	R/W	Enable global interrupt. 1: Enabled. 0:Disabled.

Interrupt Priority Register (IP, 0xB8)

Bit	7	6	5	4	3	2	1	0
Name	Reserved	PS1	PT2	PS0	PT1	PX1	PT0	PX0
Reset Value	0x00							

Bit	Name	Access	Description
0	PX0	R/W	INT0 priority level control. 1: High level. 0: Low level.
1	PT0	R/W	Timer 0 priority level control. 1: High level. 0: Low level.
2	PX1	R/W	INT1 priority level control. 1: High level. 0: Low level.
3	PT1	R/W	Timer 1 priority level control. 1: High level. 0: Low level.
4	PS0	R/W	UART0 priority level control. 1: High level. 0: Low level.
5	PT2	R/W	Timer 2 priority level control.

			1: High level. 0: Low level.
6	PS1	R/W	UART1 priority level control. 1: High level. 0: Low level.
7	Reserved		

In TCON register, all of bits that generate interrupts can be set or cleared by software, with the same result as if they had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be cancelled by software. The only exceptions are the request flags IE0 bit (TCON.1) and IE1 bit (TCON.3). If the external interrupt pin INT0 or INT1 is programmed to be level activated, IE0 and lE1 are controlled by the external source via pin INT0 and INT1, respectively. Thus, writing a one to these bits will not set the request flag IE0 and/or lE1. The same exception also applies to INT5F and INT6F.

Timer 0,1 Register (TCON, 0x88)

Bit	7	6	5	4	3	2	1	0
Name	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Reset Value	0x00							

Bit	Name	Access	Description
0	IT0	R/W	INT0 level or edge sensitivity. 1: Edge triggered. 0: Level triggered.
1	IE0	RO	INT0 interrupt flag. This bit is cleared by hardware automatically when CPU branches to interrupt routine.
2	IT1	R/W	INT1 level or edge sensitivity. 1: Edge triggered. 0: Level triggered.
3	IE1	RO	INT1 interrupt flag. This bit is cleared by hardware automatically when CPU branches to interrupt routine.
4	TR0	R/W	Timer 0 run control bit. 1: Enabled. 0: Disabled.
5	TF0	R/W	Timer 0 interrupt (overflow) flag. This bit is cleared by hardware when CPU branches to interrupt routine.
6	TR1	R/W	Timer 1 run control bit. 1: Enabled. 0: Disabled.
7	TF1	R/W	Timer 1 interrupt (overflow) flag. This bit is cleared by hardware when CPU branches to interrupt routine.

UART0 Configuration Register (SCON0, 0x98)

Bit	7	6	5	4	3	2	1	0
Name	SM00	SM01	SM02	REN0	TB08	RB08	TI0	RI0
Reset Value	0x00							

Bit	Name	Access	Description
0	RI0	R/W	UART0 receive interrupt flag, set by hardware after completion of a serial reception. Must be cleared by software.
1	TI0	R/W	UART0 transmit interrupt flag, set by hardware after completion of a serial transfer. Must be cleared by software.

2	RB08	R/W	In Modes 2 and 3 it is the 9th data bit received. In Mode 1, if SM02 is 0, RB08 is the stop bit. In Mode 0 this bit is not used.
3	TB08	R/W	The 9th transmitted data bit in Modes 2 and 3. Set or cleared by the CPU, depending on the function it performs (parity check, multiprocessor communication etc.).
4	REN0	R/W	If set, enables serial reception on UART0. Cleared by software to disable reception.
5	SM02	R/W	Enables a multiprocessor communication feature.
6	SM01	R/W	Sets baud rate.
7	SM00	R/W	Sets baud rate.

UART1 Configuration Register (SCON1, 0xC0)

Bit	7	6	5	4	3	2	1	0
Name	SM10	SM11	SM12	REN1	TB18	RB18	TI1	RI1
Reset Value	0x00							

Bit	Name	Access	Description
0	RI1	R/W	UART1 receive interrupt flag, set by hardware after completion of a serial reception. Must be cleared by software.
1	TI1	R/W	UART1 transmit interrupt flag, set by hardware after completion of a serial transfer. Must be cleared by software.
2	RB18	R/W	In Modes 2 and 3 it is the 9th data bit received. In Mode 1, if SM12 is 0, RB18 is the stop bit. In Mode 0 this bit is not used.
3	TB18	R/W	The 9th transmitted data bit in Modes 2 and 3. Set or cleared by the CPU, depending on the function it performs (parity check, multiprocessor communication etc.).
4	REN1	R/W	If set, enables serial reception on UART1. Cleared by software to disable reception.
5	SM12	R/W	Enables a multiprocessor communication feature.
6	SM11	R/W	Sets baud rate.
7	SM10	R/W	Sets baud rate.

Extended Interrupt Enable Register (EIE, 0xE8)

Bit	7	6	5	4	3	2	1	0
Name	Reserved		EWDI	EINT6	EINT5	EINT4	EINT3	EINT2
Reset Value	0x00							

Bit	Name	Access	Description
0	ENIT2	R/W	Enable INT2 interrupt for the DMA transfer interrupt request, which comes from the Memory Arbiter for the TOE/software DMA mode. 1: Enabled. 0: Disabled.
1	ENIT3	R/W	Enable INT3 interrupt for the programmable counter array interrupt request. 1: Enabled. 0: Disabled.
2	EINT4	R/W	Enable INT4 interrupt for the peripheral interrupt requests, which may be generated by TOE, MAC/PHY, I2C, SPI, 1-Wire, UART2, and CAN modules. 1: Enabled. When enabled, the summary of these peripheral interrupts are given in SFR register, PISS1R (0x9E) and PISS2R (0x9F). 0: Disabled.
3	EINT5	R/W	Enable INT5 interrupt for the internal software DMA complete and millisecond timer timeout interrupt. 1: Enabled. 0: Disabled.
4	EINT6	R/W	Enable INT6 interrupt for the wake-up interrupt request (when resuming from CPU STOP mode). 1: Enabled.

			0: Disabled.
5	EWDI	R/W	Enable Watchdog interrupt. 1: Enabled. 0: Disabled.
7:6	Reserved		

Extended Interrupt Priority Register (EIP, 0xF8)

Bit	7	6	5	4	3	2	1	0
Name	Reserved		PWDI	PINT6	PINT5	PINT4	PINT3	PINT2
Reset Value	0x00							

Bit	Name	Access	Description
0	PINT2	R/W	INT2 priority level control for DMA transfer interrupt request for TOE/software DMA mode. Please set to high priority. 1: High level. 0: Low level.
1	PINT3	R/W	INT3 priority level control for the internal programmable counter array interrupt request. 1: High level. 0: Low level.
2	PINT4	R/W	INT4 priority level control for the peripheral interrupt requests, which may be generated by TOE, MAC/PHY, I2C, SPI, 1-Wire, UART2, and CAN modules. 1: High level. 0: Low level.
3	PINT5	R/W	INT5 priority level control for the internal software DMA complete and millisecond timer timeout interrupt request. 1: High level. 0: Low level.
4	PINT6	R/W	INT6 priority level control for the wake-up interrupt request. 1: High level. 0: Low level.
5	PWDI	R/W	Watchdog priority level control. 1: High level. 0: Low level.
7:6	Reserved		

Extended Interrupt Flag Register (EIF, 0x91)

Bit	7	6	5	4	3	2	1	0
Name	Reserved			INT6F	INT5F	INT4F	INT3F	INT2F
Reset Value	0x00							

Bit	Name	Access	Description
0	INT2F	RO	INT2 interrupt flag for the DMA transfer interrupt request, which comes from the Memory Arbiter for the TOE/software DMA mode. This bit is cleared by hardware automatically when CPU branches to interrupt routine.
1	INT3F	RO	INT3 interrupt flag for the programmable counter array interrupt request. This bit is cleared by hardware automatically when CPU branches to interrupt routine.
2	INT4F	RO	INT4 interrupt flag for the peripheral interrupt requests, which may be generated by TOE, MAC/PHY, I2C, SPI, 1-Wire, UART2, and CAN modules. This bit is cleared by hardware automatically when CPU branches to interrupt routine.
3	INT5F	R/W	INT5 interrupt flag for the internal software DMA complete and millisecond timer timeout interrupt. This bit must be cleared by software within interrupt routine.
4	INT6F	R/W	INT6 interrupt flag for the wake-up interrupt request (when resuming from CPU STOP mode). This bit must be cleared by software within interrupt routine.

7:5	Reserved		

The on-chip peripheral modules such as TOE, Ethernet MAC, Ethernet PHY, I2C, SPI, 1-Wire, CAN and UART2 share the same interrupt request, INT4. The interrupt requests generated by these modules are first being merged (OR'ed) together to generate one single interrupt signal on INT4 to CPU. When CPU receives interrupt(s) on INT4, the software within interrupt routine can read SFR register, PISS1R and PISS2R to identify the source of pending interrupt(s) is coming from which module(s) and then base on the status provided here to further read the interrupt status register of each modules, corresponding to the bit being flagged.

Peripheral Interrupt Status Summary 1 Register (PISS1R, 0x9Eh)

Bit	7	6	5	4	3	2	1	0
Name	CAN_INT	I2C_INT	SPI_INT	OW_INT	TOE_INT	ETH_INT	Reserved	
Reset Value	0	0	0	0	0	0	00	

Bit	**Name**	**Access**	**Description**
1:0	Reserved	RO	
2	ETH_INT	RO	Reading “1” indicates that the Ethernet MAC/PHY has pending interrupt.
3	TOE_INT	RO	Reading “1” indicates that the TOE has pending interrupt.
4	OW_INT	RO	Reading “1” indicates that the 1-Wire controller has pending interrupt.
5	SPI_INT	RO	Reading “1” indicates that the SPI controller has pending interrupt.
6	I2C_INT	RO	Reading “1” indicates that the I2C controller has pending interrupt.
7	CAN_INT	RO	Reading “1” indicates that the CAN controller has pending interrupt.

Peripheral Interrupt Status Summary 2 Register (PISS2R, 0x9Fh)

Bit	7	6	5	4	3	2	1	0
Name	Reserved							UART2_INT
Reset Value	00							0

Bit	**Name**	**Access**	**Description**
0	UART2_INT	RO	Reading “1” indicates that the UART 2 has pending interrupt.
7:1	Reserved		

4.9 Watchdog Timer

The watchdog timer of AX11025 is a user programmable clock counter that can serve as:

- A time-base generator
- An event timer
- System supervisor

As shown in Figure 51, the watchdog timer is driven by the operating system clock, which is supplied to a series of dividers. The divider output is selectable, and determines interval between timeouts. When the timeout is reached, an interrupt flag will be set, and if enabled, a reset will occur (to reset CPU core). The interrupt flag will cause an interrupt to occur if enabled. The reset and interrupt are discrete functions that may be acknowledged or ignored, together or separately for various applications.

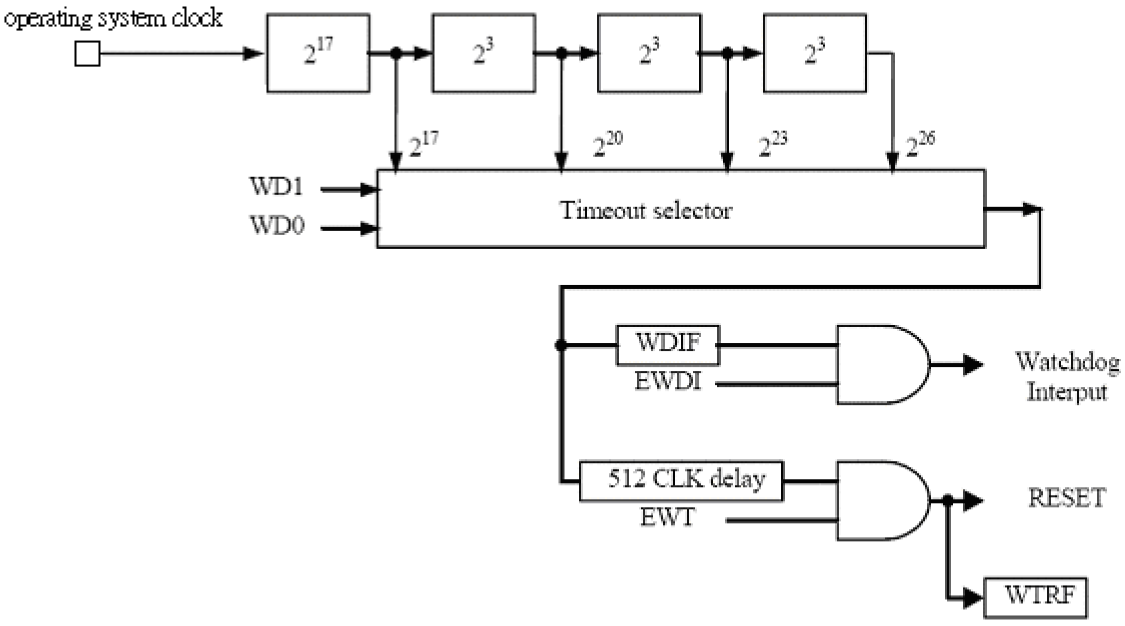


Figure 51: Watchdog Timer Block Diagram

4.9.1 Watchdog SFR Register Map

The watchdog timer has several SFR bits that contribute to its operation. It can be enabled to function as either a reset source, interrupt source, software polled timer or any combination of the three. Both the reset and interrupt have status flags. The watchdog also has a bit that restarts the timer.

Address	Name	Description
0xE8	EIE	Extended Interrupt Enable Register
0xF8	EIP	Extended Interrupt Priority Register
0xA8	IE	Interrupt Enable register.
0xD8	WDCON	Watchdog Control register.
0x8E	CKCON	Clock Control register.

Table 15: Watchdog Timer SFR Register Map

A summary table showing the bit locations of SFR register used for watchdog function is below.

Register	Bit name	Bit position	Description
EIE	EWDI	EIE.5	Enable Watchdog Timer Interrupt.
EIP	PWDI	EIP.5	Priority of Watchdog Timer Interrupt.
CKCON	WD[1:0]	CKCON.7-6	Watchdog Interval
WDCON	RWT	WDCON.0	Reset Watchdog Timer
	EWT	WDCON.1	Enable Watchdog Timer Reset
	WTRF	WDCON.2	Watchdog Timer Reset flag
	WDIF	WDCON.3	Watchdog Interrupt flag

4.9.2 Watchdog Interrupt

The watchdog interrupt can be turned on/off by EIE register, and set into high/low priority group by EIP register. Please refer to section 4.8 for details. Upon enabled, the watchdog interrupt flag is reported in WDIF bit (WDCON.3). This bit that generates interrupts can be set or cleared by software, with the same result as if they had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be cancelled by software.

Watchdog Control Register (WDCON, 0xD8)

Bit	7	6	5	4	3	2	1	0
Name	Reserved				WDIF	WTRF	EWT	RWT
Reset Value	0x00							

Bit	Name	Access	Description
0	RWT	R/W	Reset Watchdog Timer. 1: Setting RWT resets the watchdog timer count. Timed Access procedure must be used to set this bit before the watchdog timer expires, or a watchdog timer reset and/or interrupt will be generated if enabled. 0: After software sets this bit, the hardware will automatically clear it.
1	EWT	R/W	Enable Watchdog Timer Reset. The reset of CPU by watchdog timer is controlled by this bit. This bit has no effect on the ability of the watchdog timer to generate a watchdog interrupt. Timed Access procedure must be used to modify this bit. 1: Watchdog timer timeout resets CPU. 0: Watchdog timer timeout doesn't reset CPU.
2	WTRF	R/W	Watchdog Timer Reset Flag. 1: When set by hardware, indicates that a watchdog timer reset has occurred. Set by software do not generate a watchdog timer reset. 0: It is cleared by chip reset pin, RST_N, but otherwise must be cleared by software. The watchdog timer has no effect on this bit, when EWT bit is cleared.
3	WDIF	R/W	Watchdog Interrupt Flag. 1: WDIF in conjunction with the Enable Watchdog Interrupt bit (EIE.5), and EWT, indicates if a watchdog timer event has occurred and what action should be taken. Setting WDIF in software will generate a watchdog interrupt if enabled. Timed access registers procedure can be used to modify this bit. 0: This bit must be cleared by software before exiting the interrupt service routine, or another interrupt is generated.
7:4	Reserved		

A Watchdog timeout reset will not disable the Watchdog Timer, but restarts the timer. In general, software should set the Watchdog to whichever state is desired, just to be certain of its state.

Table below summarizes Watchdog Control bits and taken operation concerned to theirs values.

EWT	EWDI	WDIF	Result
X	X	0	No watchdog event.
0	0	1	Watchdog time-out has expired. No interrupt has been generated.
0	1	1	Watchdog interrupt has occurred.
1	0	1	Watchdog time-out has expired. No interrupt has been generated. Watchdog timer reset will occur in 512 clock periods (of operating system clock) if RWT is not strobed.
1	1	1	Watchdog interrupt has occurred. Watchdog timer reset will occur in 512 clock periods (of operating system clock) if RWT is not set using Timed Access procedure.

Table 16: Watchdog Bits And Actions

4.9.3 Watchdog Timer Reset

The Watchdog Timer Reset function works as follows. After initializing the correct timeout interval, software first restarts the Watchdog using RWT bit (WDCON.0) and then enables the reset mode by setting the EWT bit (WDCON.1). At any time prior to reaching its user selected terminal value, software can set the RWT bit (WDCON.0). If RWT is set before the timeout is reached, the timer will start over. If the timeout is reached without RWT bit being set, the Watchdog will reset the CPU. Hardware will automatically clear RWT after software sets it. When the reset occurs, the WTRF bit (WDCON.2) will automatically be set to indicate the cause of the reset, however software must clear this bit manually. A Watchdog timeout reset will not disable the Watchdog Timer, but restarts the timer. In general, software should set the Watchdog to whichever state is desired, just to be certain of its state.

4.9.4 Simple Timer

The Watchdog Timer is a free running timer. When used as a simple timer with both the reset (EWT=0, WDCON.1) and interrupt functions disabled (EWDI=0, EIE.5), the timer will continue to set the Watchdog Interrupt flag each time the timer completes the selected timer interval as programmed by WD[1:0] bits (CKCON.7-6). Restarting the timer using the RWT bit (WDCON.0), allows software to use the timer in a polled timeout mode. The WDIF bit is cleared by software or any reset. The Watchdog Interrupt is also available for applications that do not need a true Watchdog Reset but simply a very long timer. The interrupt is enabled using the EWDI bit (EIE.5). When the timeout occurs, the Watchdog Timer will set the WDIF bit (WDCON.3), and an interrupt will occur if the global interrupt enable, EA bit (IE.7) is set. A potential Watchdog Reset is executed 512 clocks after setting of WDIF flag. The WDIF flag indicates the source of the interrupt, and software must clear WDIF flag. Proper use of the Watchdog Interrupt with the Watchdog Reset allows interrupt software to survey the system for errant conditions.

4.9.5 System Monitor

When using the Watchdog Timer as a system monitor, the Watchdog Reset function should be used. If the Interrupt function were used, the purpose of the watchdog would be defeated. For example, assume the system is executing errant code prior to the Watchdog Interrupt. The interrupt would temporarily force the system back into control by vectoring the CPU to the interrupt service routine. Restarting the Watchdog and exiting by an RETI or RET, would return the processor to the lost position prior to the interrupt. By using the Watchdog Reset function, the CPU is restarted from the beginning of the program, and therefore placed into a known state.

4.9.6 Clock Control

The Watchdog timeout selection is made using WD[1:0] bits in Clock control register, CKCON (0x8E), which select Watchdog timer timeout period. The Watchdog is clocked directly from operating system clock, and PMM mode directly affects its timeout period. It is increased 100 times slower when the CPU is in PMM mode (because operating system clock is running 1/100 of original frequency in PMM). This allows the watchdog period to remain synchronized with device operation. Number of clocks needed for timeout does not depend on PMM, and is constant as shown in table in below register. The Watchdog has four timeout selections based on the operating system clock frequency. The selections are a pre-selected number of clocks. Therefore, the actual timeout interval is dependent on the operating system clock frequency. Note that the periods shown above are for the interrupt events. The Reset, when enabled, is generated 512 clocks later regardless of whether the interrupt is used. Therefore, the actual Watchdog timeout period is the number shown above plus 512 clocks (of operating system clock).

Clock Control Register (CKCON, 0x8E)

<table>
<tr><th>Bit</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td><b>Name</b></td><td colspan="2">WD</td><td>T2M</td><td>T1M</td><td>T0M</td><td colspan="3">MD</td></tr>
<tr><td><b>Reset Value</b></td><td colspan="8">0x07</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Name</th><th>Access</th><th>Description</th></tr>
<tr><td>2:0</td><td>MD</td><td>R/W</td><td>This adjusts the stretch cycles of XDATWR_N and XDATRD_N signals during MOVX instruction for External Data Memory write and read access cycles. The Minimal read/write pulse length is equal to 1 clock period (MD = 000) and maximal 8 clock periods (MD = 111). The MD bits can be changed any time during program execution.

Based on operating system clock frequency and typical SRAM chip with 8ns access time, the recommended setting is as below,
<table>
<tr><th rowspan="2">System Clock</th><th colspan="2">Data Memory Wait State Setting, MD</th></tr>
<tr><th>Program Code Shadow Mode</th><th>Non-Shadow Mode</th></tr>
<tr><td>25Mhz</td><td>001</td><td>001</td></tr>
<tr><td>50Mhz</td><td>001</td><td>001</td></tr>
<tr><td>100Mhz</td><td>001</td><td>001</td></tr>
</table></td></tr>
<tr><td>3</td><td>T0M</td><td>R/W</td><td>This bit controls the division of the system clock that drives Timer 0.
1: Timer 0 uses a divide-by-4 of the operating system clock frequency.
0: Timer 0 uses a divide-by-12 of the operating system clock frequency.</td></tr>
<tr><td>4</td><td>T1M</td><td>R/W</td><td>This bit controls the division of the system clock that drives Timer 1.
1: Timer 1 uses a divide-by-4 of the operating system clock frequency.
0: Timer 1 uses a divide-by-12 of the operating system clock frequency.</td></tr>
<tr><td>5</td><td>T2M</td><td>R/W</td><td>This bit controls the division of the system clock that drives Timer 2. This bit has no effect when the timer is in baud rate generator mode.
1: Timer 2 uses a divide-by-4 of the operating system clock frequency.
0: Timer 2 uses a divide-by-12 of the operating system clock frequency.</td></tr>
<tr><td>7:6</td><td>WD</td><td>R/W</td><td>WD bits select Watchdog timer timeout period.
<table>
<tr><th>WD</th><th>Watchdog Interval</th><th>Number of Clocks</th></tr>
<tr><td>00</td><td>2^{17}</td><td>131072</td></tr>
<tr><td>01</td><td>2^{20}</td><td>1048576</td></tr>
<tr><td>10</td><td>2^{23}</td><td>8388608</td></tr>
<tr><td>11</td><td>2^{26}</td><td>67108864</td></tr>
</table></td></tr>
</table>

4.9.7 Timed Access Register

Timed Access registers have built in mechanism preventing them from accidental writes. TA is located at 0xEB SFR address. To do a correct write to such register the following sequence has to be applied:

```
MOV TA, #0xAA
MOV TA, #0x55
; Any direct addressing instruction writing timed access register.
```

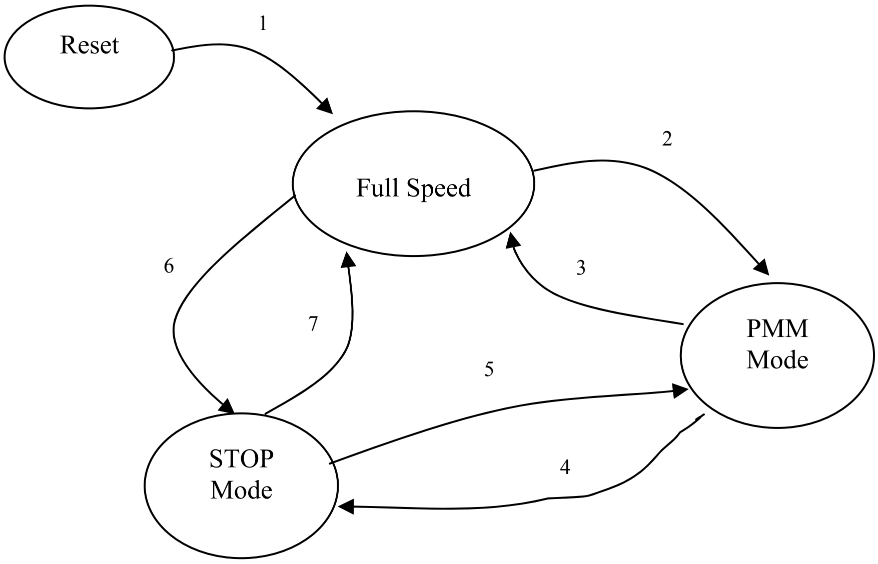
The time elapsed between first, second, and third operation does not matter (any number of Program Wait Sates is allowed). The only correct sequence is required. Any third instruction causes protection mechanism to be turned on. This means that time protected register is opened for write only for single instruction. Reading from such register is never protected. Timed Access registers are listed in table below.

Register name	Description
WDCON (0xD8)	Watchdog Configuration.
ACON (0x9D)	Address Control register.

Table 17: Timed Access Registers

4.10 Power Management Unit

The figure below shows 3 possible modes of operation of AX11025. The Full Speed mode is when the operating system clock is running at full clock rate (i.e., 25Mhz, 50Mhz, or 100Mhz, depending on SYSCK_SEL[1:0] setting). The PMM (Power Management Mode) is when the operating system clock is running at 1/100 of full clock rate. The STOP mode is when the operating system clock is turned off and the CPU is in complete stop mode. For typical power consumption of AX11025 in these operating modes, please refer to section 5.2.



Symbol	Description
1	Reset condition puts the chip in Full Speed mode after the reset removal.
2	Software sets PMM bit = 1 (PCON.0) with SWB bit = 1 or 0 (PCON.2) to enter the PMM mode with switchback enabled or disabled.
3	See section 4.10.2 for detailed description.
4, 6	Software sets STOP bit = 1 (PCON.1) to enter STOP mode.
5,7	See section 4.10.3 for detailed description.

Figure 52: AX11025 Operating Mode Transition Diagram

4.10.1 Power Management Unit SFR Register Map

Address	Name	Description
0x87	PCON	Power Configuration Register.
0xE9	STATUS	Status Register.

Table 18: Power Management Unit SFR Register Map

4.10.2 Power Management Mode

The Power Management Mode (PMM) feature allows software dynamically matches operating frequency and current consumption with the need for processing power. Instead of the full clock rate provided to the CPU core and most system logic, the PMM mode tells the clock generation block to divide the operating system clock frequency by 100 to operate the chip in reduced speed to conserve power.

The Switchback feature allows the CPU almost immediately returning to full speed mode upon acknowledgment of an external interrupt or a falling edge on a serial port receiver pin, RXD0/RXD1 of UART0/UART1. Additionally, CPU operating in PMM would normally be unable to sample an incoming serial transmission and properly receive it. The switchback feature allows the CPU to return to full speed operation in time to receive incoming serial port data and process interrupts with no loss in performance.

STATUS (0xE9) register is incorporated to prevent the software from accidentally reducing the clock rate during the servicing of an external interrupt or serial port activity. This register can be interrogated to determine if a high priority, or low priority interrupt is in progress, or if serial port activity is occurring. Based on this information the software can delay or reject a planned change in the clock divider rate in clock generation block of AX11025.

STATUS Register (STATUS, 0xE9)

Bit	7	6	5	4	3	2	1	0
Name	Reserved	HIP	LIP	Reserved	SPTA1	SPRA1	SPTA0	SPRA0
Reset Value	0x00							

Bit	Name	Access	Description
0	SPRA0	RO	UART0 receiver activity status. 1: UART0 receiver active. 0: UART0 receiver inactive.
1	SPTA0	RO	UART0 transmitter activity status. 1: UART0 transmitter active. 0: UART0 transmitter inactive.
2	SPRA1	RO	UART1 receiver activity status. 1: UART1 receiver active. 0: UART1 receiver inactive.
3	SPTA1	RO	UART1 transmitter activity status. 1: UART1 transmitter active. 0: UART1 transmitter inactive.
4	Reserved		
5	LIP	RO	Low Priority (LP) interrupt status. 1: LP interrupt progressing. 0: no LP interrupt.
6	HIP	RO	High Priority (HP) interrupt status. 1: HP interrupt progressing. 0: no HP interrupt.
7	Reserved		

When PMM is invoked via setting PMM bit (PCON.0), it controls the clock generation block to divide the operating system clock frequency by 100. Note that all internal functions, on-chip timers (including serial port baud rate generation), watchdog timer, millisecond timer, and software timing loops will run at the reduced speed. In addition, use of the switchback feature is possible to affect a return from PMM to the full speed mode. This allows both hardware and software to cause an exit from PMM. It is the responsibility of the software to test for UART activity before attempting to change speed, as a modification of the operating system clock during a UART operation will corrupt the data. In general, it is not possible to generate standard baud rates while in PMM, and the user is advised to avoid PMM or use the Switchback feature if UART operation is desired.

Power Configuration Register (PCON, 0x87)

Bit	7	6	5	4	3	2	1	0
Name	SMOD0	SMOD1	Reserved	PWE	RSM	SWB	STOP	PMM
Reset Value	0x00							

Bit	**Name**	**Access**	**Description**
0	PMM	R/W	Power Management Mode Enable bit. 1: PMM entered. 0: PMM disabled.
1	STOP	R/W	STOP mode bit. 1: STOP mode entered. 0: Disabled.
2	SWB	R/W	Switchback enable. 1: Enabled interrupts and serial ports cause switchback. PMM bit is cleared. 0: Interrupts and serial ports don't affect PMM bit. Note that after leaving PMM mode, the software shall also clear this bit.
3	RSM	R/W	Regulator Standby Mode. 1: Set the internal 3.3V to 1.8V regulator to operate at standby mode (when the 1.8V current drawn is less than 30mA) for better conversion efficiency. 0: Set the internal 3.3V to 1.8V regulator to full operating mode (when the 1.8V current drawn is more than 30mA) for better conversion efficiency.
4	PWE	R/W	Program memory Write Enable bit. 1: Enable Program Memory write access signal activity during MOVX instructions. 0: Disabled.
5	Reserved	R/W	
6	SMOD1	R/W	UART1 double baud rate bit.
7	SMOD0	R/W	UART0 double baud rate bit.

Switchback Feature

The Switchback feature solves one of the most vexing dilemmas faced by power conscious systems. Many applications are unable to use STOP mode because they require constant computation. The feature allows a system to operate at a relatively slow speed, and burst to a faster mode when required by an external event. Enable this feature by setting the SWB bit (PCON.2), a qualified interrupt (interrupt which has occurred and been acknowledged) or serial port reception or transmission cause the CPU to return to full speed mode. An interrupt must be enabled and not blocked by a higher priority interrupt. Software should manually return the CPU to PMM after the event is completed. The following events can trigger AX11025 switchback to full speed mode from PMM:

1. Receive interrupt on external interrupt pin, INT0 or INT1 if enabled in EX0 bit (IE.0) or EX1 bit (IE.2)
2. Detect falling-edge transition (start bit) on RXD0 pin of UART0 or RXD1 pin of UART1 if enabled in REN0 bit (SCON0.4) or REN1 bit (SCON1.4).
3. Transmit buffer loaded in UART0 or UART1.
4. Watchdog timer reset.

In addition, the following events can also trigger AX11025 switchback to full speed mode from PMM, via INT 6:

1. Receive rising-edge signal on external remote-wakeup trigger input pin, EXT_WKUP, if enabled in EPWT bit (SPWIE.5).
2. Receive Magic packet from Ethernet, if enabled in RWMP bit (SPWIE.4).
3. Receive pre-defined Wakeup frame from Ethernet, if enabled in MWFE bit (SPWIE.6) and EWFF0/1 bit (WFCR.0/2).
4. Detect link-up signal from the embedded Ethernet PHY, if enabled in PPLWE bit (SPWIE.0).

5. Detect link-up signal from secondary PHY, if enabled in SPLWE bit (SPWIE.2).
6. Detect falling-edge transition (start bit) on RXD2 pin or falling-edge transition on RI pin of UART 2, if enabled in WE bit (HSIER.4, 0xE2).

In the case of a UART0/1-initiated switchback, the switchback is not generated by the associated interrupt. This is because the AX11025 operating in PMM will not be able correctly receiving a byte of data to generate an interrupt. Instead, Switchback is generated by a UART0/1 reception on the falling edge associated with the start bit, if the associated receiver enable bit (SCON0.4 or SCON1.4) is set. For UART transmissions, a switchback is generated when the UART0/1 buffer is loaded. This ensures the CPU will be operating in full speed mode when the data is being transmitted, and eliminates the need for a write to the PMM bit (PCON.0) to exit PMM before transmitting. The Switchback feature is unaffected by the state of the serial port interrupt flags.

Switchback Feature Timing

The timing of the Switchback is dependent on the source. Interrupt–initiated (such as INT0, INT1, INT6) switchbacks will occur at the start of the first cycle following the event initiating the Switchback. If the current instruction in progress is a write to the IE, IP, EIE, or EIP registers, interrupt processing will be delayed until the completion of the following instruction. UART-initiated (such as UART0/1) switchbacks occur at the start of the instruction following the MOV that loads SBUF0 or SBUF1. UART-initiated (such as UART0/1) switchbacks occur during the cycle in which the falling edge was detected.

There are a few points that must be considered when using a serial port reception to generate Switchback. Under normal circumstances, noise on the line or an aborted transmission would cause the serial port to timeout and the data to be ignored. This presents a problem if the Switchback is used, however, because Switchback would occur but there is no indication to the system that one has occurred. If PMM and serial port Switchback functions are used in a noisy environment, the user is advised periodically checking if AX11025 has accidentally exited PMM. A similar problem can occur if multiprocessor communication protocols are used in conjunction with PMM. The problem is that an invalid address that should be ignored by a particular processor will still generate Switchback. As a result, it is not recommended to use a multiprocessor communication scheme in conjunction with PMM. If the system power considerations will allow for an occasional erroneous Switchback, a polling scheme can be used to place the AX11025 back into PMM.

4.10.3 STOP Mode

The STOP mode is the lowest power state that the AX11025 can enter. This is achieved by cutting off the clock feeding to the CPU core and the peripheral logics. When entering the STOP mode, the TOFFOP bit (Flag.1) in I2C EEPROM determines whether the 25Mhz oscillator and internal PLL will be disabled or not during STOP mode. When AX11025 is running in Full Speed or PMM mode, software can enter STOP mode by setting STOP bit (PCON.1).

If the STOP mode is entered with 25Mhz oscillator and PLL completely disabled (TOFFOP bit (Flag.1)= 1), the STOP mode can be exited in following ways:

1. Receive rising-edge signal on external remote-wakeup trigger pin, EXT_WKUP, if enabled in EPWT bit (SPWIE.5).
2. Detect falling-edge transition (start bit) on RXD2 pin or falling-edge transition on RI pin of UART 2, if enabled in WE bit (HSIER.4).
3. Receive hardware reset on RST_N pin (CPU operation will resume execution at address 0x00_0000).

Please note that if software sets both EPWT bit (SPWIE.5) and WE bit (HSIER.4, 0xE2) to “0” prior to entering STOP mode while the TOFFOP bit = 1, then the chip can not be awaked by any event at all and only the hardware chip reset can remove the chip from STOP mode back to normal functional mode.

If the STOP mode is entered with 25Mhz oscillator and PLL still running (TOFFOP bit (Flag.1) = 0), the STOP mode can be exited in following ways, depending on software configuration before entering the STOP mode:

1. Receive rising-edge signal on external remote-wakeup trigger pin, EXT_WKUP, if enabled in EPWT bit (SPWIE.5).
2. Receive Magic packet from Ethernet, if enabled in RWMP bit (SPWIE.4).
3. Receive pre-defined Wakeup frame from Ethernet, if enabled in MWFE bit (SPWIE.6) and EWFF0/1 bit (WFCR.0/2).
4. Detect link-up signal from the embedded Ethernet PHY, if enabled in PPLWE bit (SPWIE.0).
5. Detect link-up signal from secondary PHY, if enabled in SPLWE bit (SPWIE.2).
6. Detect falling-edge transition (start bit) on RXD2 pin or falling-edge transition on RI pin of UART 2, if enabled in WE bit (HSIER.4).
7. Receive hardware reset on RST_N pin (CPU operation will resume execution at address 0x00_0000).

Example Programming Procedure

Case 1: Using a rising-edge signal on EXT_WKUP pin to awake up the chip, the software shall:

1. Enable wakeup interrupt by setting EPWT bit (SPWIE.5) and EINT6 bit (EIE.4), disable Ethernet PHY to reduce power consumption by setting PHY register’s Power-Down bit (BMCR.11), and stop oscillator and PLL during STOP mode by setting TOFFOP bit (Flag.1) = 1.
2. Then set STOP bit (PCON.1) to enter STOP mode. Now the system clock will be turned off and the oscillator and PLL will be stopped too.
3. Upon detecting a rising-edge on EXT_WKUP pin, the oscillator and PLL will first resume running and the clock generation block will re-enable the system clock after it is stabled enough, and then the INT 6 will be asserted to notify CPU.

Case 2: Using receiving Magic packet or Wakeup frame to awake up the chip, the software shall:

1. Enable wakeup interrupt by having RWMP bit (SPWIE.4) or MWFE bit (SPWIE.6), keep Ethernet PHY power-on to allow receiving Ethernet packet by clearing PHY register’s Power-Down bit (BMCR.11), keep oscillator and PLL running by setting TOFFOP bit (Flag.1) = 0.
2. If Wakeup frame wakeup event is used, also define the Wakeup frame pattern in related registers.
3. Then set STOP bit (PCON.1) to enter STOP mode. Now the system clock will be turned off while oscillator and PLL keeps on running.
4. Upon receiving Magic packet or Wakeup frame from Ethernet, the clock generation block will re-enable the system clock and then the INT 6 will be asserted to notify CPU.

Case 3: Using the link-up event of the embedded Ethernet PHY to awake up the chip, the software shall:

1. Enable wakeup interrupt by setting PPLWE bit (SPWIE.0) and EINT6 bit (EIE.4), keep Ethernet PHY power-on to allow link-up detection by clearing PHY register’s Power-Down bit (BMCR.11), keep oscillator and PLL running by setting TOFFOP bit (Flag.1) = 0.
2. Then set STOP bit (PCON.1) to enter STOP mode. Now the system clock will be turned off while oscillator and PLL keeps on running.
3. Upon detecting the link-up event on embedded Ethernet PHY, the clock generation block will re-enable the system clock and then the INT 6 will be asserted to notify CPU.

4.11 Timers and Counters

The AX11025 contains three 16-bit timer/counters, namely, Timer 0, Timer 1, and a fully compatible with the standard 8052 Timer 2, and one dedicated Millisecond Timer which is programmable with 1ms resolution for software use.

In the "timer mode", timer registers are incremented every 12 or 4 operating system clock periods when appropriate timer is enabled. In the "counter mode" the timer registers are incremented every falling transition on their corresponding input pins: TM0_CK, TM1_CK, or TM2_CK. The input pins are sampled every operating system clock period. The following table shows Timer 0, 1, 2 pin description. All pins are input direction and no three-state output pin.

Pin	I/O	Polarity	Description
TM0_CK	I	Falling	Timer 0 external clock input
TM0_GT	I	High	Timer 0 clock input gate control input to facilitate pulse width measurements.
TM1_CK	I	Falling	Timer 1 external clock input
TM1_GT	I	High	Timer 1 clock input gate control input to facilitate pulse width measurements.
TM2_CK	I	Falling	Timer 2 external clock input
TM2_GT	I	High	Timer 2 clock input gate control input.

Table 19: Timers 0, 1, 2 Pin Description

4.11.1 Timers 0, 1, 2 Related SFR Register Map

Address	Name	Description
0x89	TMOD	Timer 0,1 Control Mode Register.
0x88	TCON	Timer 0,1 Configuration Register.
0x8E	CKCON	Clock Control Register.
0x8A	TL0	Timer 0 Low Byte Register.
0x8C	TH0	Timer 0 High Byte Register.
0x8B	TL1	Timer 1 Low Byte Register.
0x8D	TH1	Timer 1 High Byte Register.
0xA8	IE	Interrupt Enable Register.
0xB8	IP	Interrupt Priority Register.
0xC8	T2CON	Timer 2 Configuration Register.
0xCA	RLDL	Timer 2 Reload Low Byte Register.
0xCB	RLDH	Timer 2 Reload High Byte Register.
0xCC	TL2	Timer 2 Low Byte Register.
0xCD	TH2	Timer 2 High Byte Register.

Table 20: Timers 0, 1, 2 Related SFR Register Map

4.11.2 Timer 0, 1

Timer 0 and Timer 1 are fully compatible with the standard 8051 timers. Each timer consists of two 8-bit registers, and they are TH0 (0x8C), TL0 (0x8A), TH1 (0x8D), and TL1 (0x8B). Timers 0 and Timer 1 work in the same four modes, namely, Mode 0, Mode 1, Mode 2, and Mode 3, as described in following TMOD register description.

Timer 0, 1 Control Mode Register (TMOD, 0x89)

Bit	7	6	5	4	3	2	1	0
Name	GATE	CT	M1	M0	GATE	CT	M1	M0
Reset Value	0x00							

<table>
<tr><th>Bit</th><th>Name</th><th>Access</th><th>Description</th></tr>
<tr><td>1:0</td><td>M1, M0</td><td>R/W</td><td>Timer 0 mode select bits. The table below shows the 4 operating modes of Timer 0.
<table>
<tr><th>Mode</th><th>M1</th><th>M0</th><th>Timer 0 Operating Mode Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>TH0 operates as 8-bit timer/counter with a divide by 32 prescaler served by lower 5-bit of TL0.</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>16-bit timer/counter. TH0 and TL0 are cascaded.</td></tr>
<tr><td>2</td><td>1</td><td>0</td><td>TL0 operates as 8-bit timer/counter with 8-bit auto-reload by TH0.</td></tr>
<tr><td>3</td><td>1</td><td>1</td><td>TL0 is configured as 8-bit timer/counter controlled by the standard Timer 0 bits. TH0 is an 8-bit timer controlled by the Timer 1 controls bits. Timer 1 holds its count.</td></tr>
</table></td></tr>
<tr><td>2</td><td>CT</td><td>R/W</td><td>Timer 0 Counter or Timer select bit.
1: Counter mode, Timer 0 clock source from TM0_CK pin.
0: Timer mode, internally clocked.</td></tr>
<tr><td>3</td><td>GATE</td><td>R/W</td><td>Timer 0 Gating control.
1: Timer 0 enabled while TM0_GT pin is high and TR0 control bit (TCON.4) is set, to facilitate pulse width measurements.
0: Timer 0 enabled while TR0 control bit (TCON.4) is set.</td></tr>
<tr><td>5:4</td><td>M1, M0</td><td>R/W</td><td>Timer 1 mode select bits. The table below shows the 4 operating modes of Timer 1.
<table>
<tr><th>Mode</th><th>M1</th><th>M0</th><th>Timer 1 Operating Mode Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>TH1 operates as 8-bit timer/counter with a divide by 32 prescaler served by lower 5-bit of TL1.</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>16-bit timer/counter. TH1 and TL1 are cascaded.</td></tr>
<tr><td>2</td><td>1</td><td>0</td><td>TL1 operates as 8-bit timer/counter with 8-bit auto-reload by TH1.</td></tr>
<tr><td>3</td><td>1</td><td>1</td><td>TL0 is configured as 8-bit timer/counter controlled by the standard Timer 0 bits. TH0 is an 8-bit timer controlled by the Timer 1 controls bits. Timer 1 holds its count.</td></tr>
</table></td></tr>
<tr><td>6</td><td>CT</td><td>R/W</td><td>Timer 1 Counter or Timer select bit.
1: Counter mode, Timer 1 clock source from TM1_CK pin.
0: Timer mode, internally clocked.</td></tr>
<tr><td>7</td><td>GATE</td><td>R/W</td><td>Timer 1 Gating control.
1: Timer 1 enabled while TM1_GT pin is high and TR1 control bit (TCON.6) is set, to facilitate pulse width measurements.
0: Timer 1 enabled while TR1 control bit (TCON.6) is set.</td></tr>
</table>

Timer 0, 1 Configuration Register (TCON, 0x88)

Bit	7	6	5	4	3	2	1	0
Name	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Reset Value	0x00							

Bit	Name	Access	Description
0	IT0	R/W	INT0 level or edge sensitivity. 0: Level triggered. 1: Edge triggered.
1	IE0	RO	INT0 interrupt flag. This bit is cleared by hardware automatically when CPU branches to interrupt routine.
2	IT1	R/W	INT1 level or edge sensitivity. 0: Level triggered. 1: Edge triggered.
3	IE1	RO	INT1 interrupt flag. This bit is cleared by hardware automatically when CPU branches to interrupt routine.
4	TR0	R/W	Timer 0 run control bit. 1: Enabled. 0: Disabled.

5	TF0	RO	Timer 0 interrupt (overflow) flag. This bit is cleared by hardware when CPU branches to interrupt routine.
6	TR1	R/W	Timer 1 run control bit. 1: Enabled. 0: Disabled.
7	TF1	RO	Timer 1 interrupt (overflow) flag. This bit is cleared by hardware when CPU branches to interrupt routine.

In the “timer mode”, timer registers are incremented every 12 or 4 operating system clock periods, configured by T0M and T1M bits (CKCON.3~4).

The Timer 0 and Timer 1 interrupt enable registers are ET0 bit (IE.1, 0xA8) and ET1 bit (IE.3), respectively, and their interrupt priority registers are PT0 bit (IP.1) and PT1 bit (IP.3), respectively. The Timer 0 and Timer 1 interrupt (overflow) flag are TF0 bit (TCON.5) and TF1 (TCON.7), respectively. Please refer to section 4.8 for details.

Clock Control Register (CKCON, 0x8E)

Bit	7	6	5	4	3	2	1	0
Name	WD		T2M	T1M	T0M	MD		
Reset Value	0x07							

<table>
<tr><th>Bit</th><th>Name</th><th>Access</th><th>Description</th></tr>
<tr><td>2:0</td><td>MD</td><td>R/W</td><td>This adjusts the stretch cycles of XDATWR_N and XDATRD_N signals during MOVX instruction for External Data Memory write and read access cycles. The Minimal read/write pulse length is equal to 1 clock period (MD = 000) and maximal 8 clock periods (MD = 111). The MD bits can be changed any time during program execution.
Based on operating system clock frequency and typical SRAM chip with 8ns access time, the recommended setting is as below,
<table>
<tr><th rowspan="2">System Clock</th><th colspan="2">Data Memory Wait State Setting, MD</th></tr>
<tr><th>Program Code Shadow Mode</th><th>Non-Shadow Mode</th></tr>
<tr><td>25Mhz</td><td>001</td><td>001</td></tr>
<tr><td>50Mhz</td><td>001</td><td>001</td></tr>
<tr><td>100Mhz</td><td>001</td><td>001</td></tr>
</table></td></tr>
<tr><td>3</td><td>T0M</td><td>R/W</td><td>This bit controls the division of the system clock that drives Timer 0.
1: Timer 0 uses a divide-by-4 of the operating system clock frequency.
0: Timer 0 uses a divide-by-12 of the operating system clock frequency.</td></tr>
<tr><td>4</td><td>T1M</td><td>R/W</td><td>This bit controls the division of the system clock that drives Timer 1.
1: Timer 1 uses a divide-by-4 of the operating system clock frequency.
0: Timer 1 uses a divide-by-12 of the operating system clock frequency.</td></tr>
<tr><td>5</td><td>T2M</td><td>R/W</td><td>This bit controls the division of the system clock that drives Timer 2. This bit has no effect when the timer is in baud rate generator mode.
1: Timer 2 uses a divide-by-4 of the operating system clock frequency.
0: Timer 2 uses a divide-by-12 of the operating system clock frequency.</td></tr>
<tr><td>7:6</td><td>WD</td><td>R/W</td><td>WD bits select Watchdog timer timeout period.
<table>
<tr><th>WD</th><th>Watchdog Interval</th><th>Number of Clocks</th></tr>
<tr><td>00</td><td>2^{17}</td><td>131072</td></tr>
<tr><td>01</td><td>2^{20}</td><td>1048576</td></tr>
<tr><td>10</td><td>2^{23}</td><td>8388608</td></tr>
<tr><td>11</td><td>2^{26}</td><td>67108864</td></tr>
</table></td></tr>
</table>

Timer 0 - Mode 0

In this mode, the Timer 0 register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, Timer 0 interrupt flag TF0 is set. The counted input is enabled to the Timer 0 when TCON.4 = 1 and either TMOD.3 = 0 or TM0_GT = 1. (Setting TMOD.3 = 1 allows the Timer 0 to be controlled by external input TM0_GT0, to facilitate pulse width measurements). The 13-bit register consists of all 8 bits of TH0 and the lower 5 bits of TL0. The upper 3 bits of TL0 are indeterminate and should be ignored.

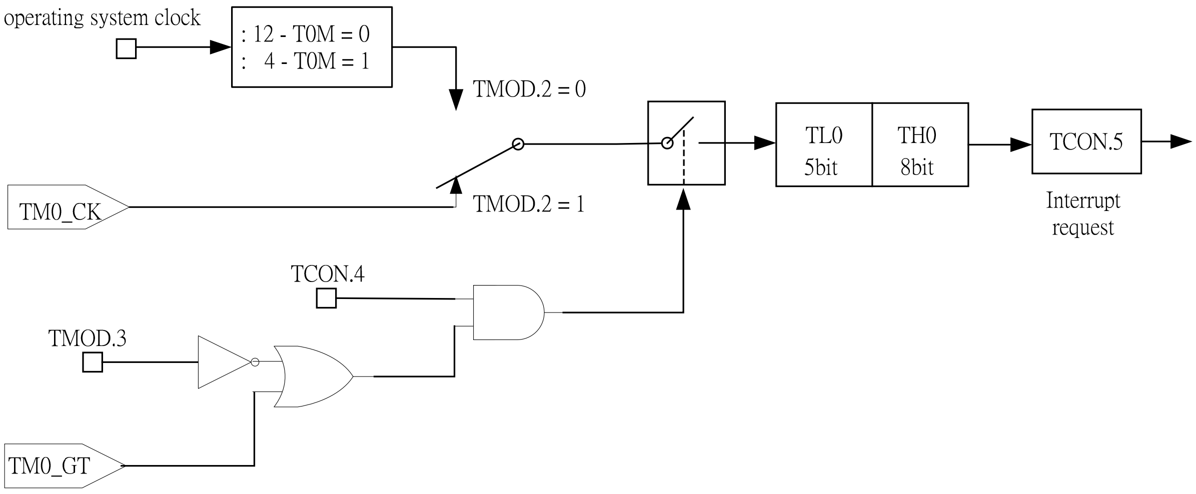


Figure 53: Timer/Counter 0, Mode 0: 13-Bit Timer/Counter

Timer 0 - Mode 1

Mode 1 is the same as Mode 0, except that the timer register is running with all 16 bits. Mode 1 is shown in figure below.

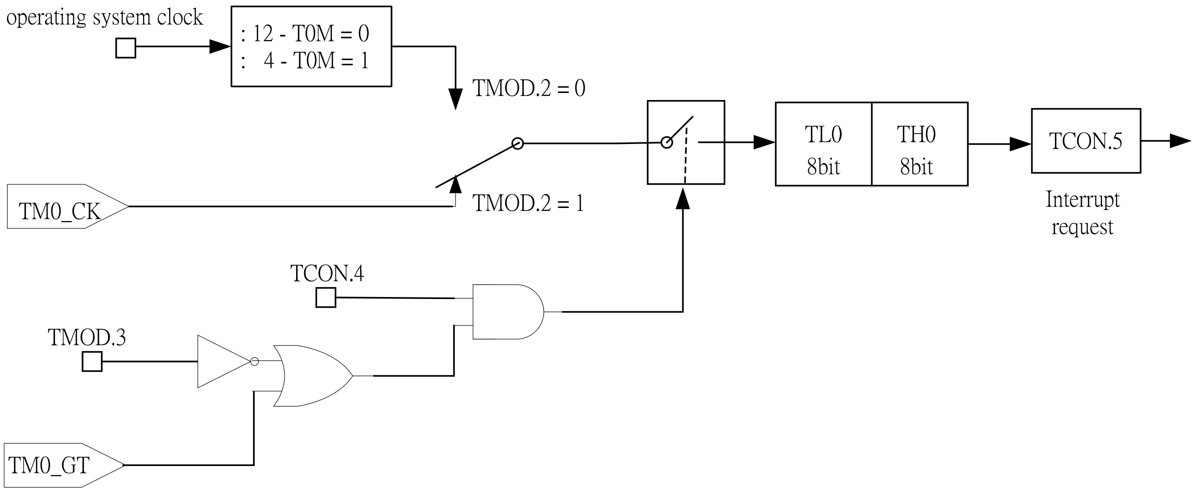


Figure 54: Timer/Counter 0, Mode 1: 16-Bit Timer/Counter

Timer 0 - Mode 2

Mode 2 configures the timer register as an 8-bit counter (TL0) with automatic reloads, as shown in figure below. Overflow from TL0 not only sets TF0, but also reloads TL0 with the contents of TH0, which is loaded by software. The reload leaves TH0 unchanged.

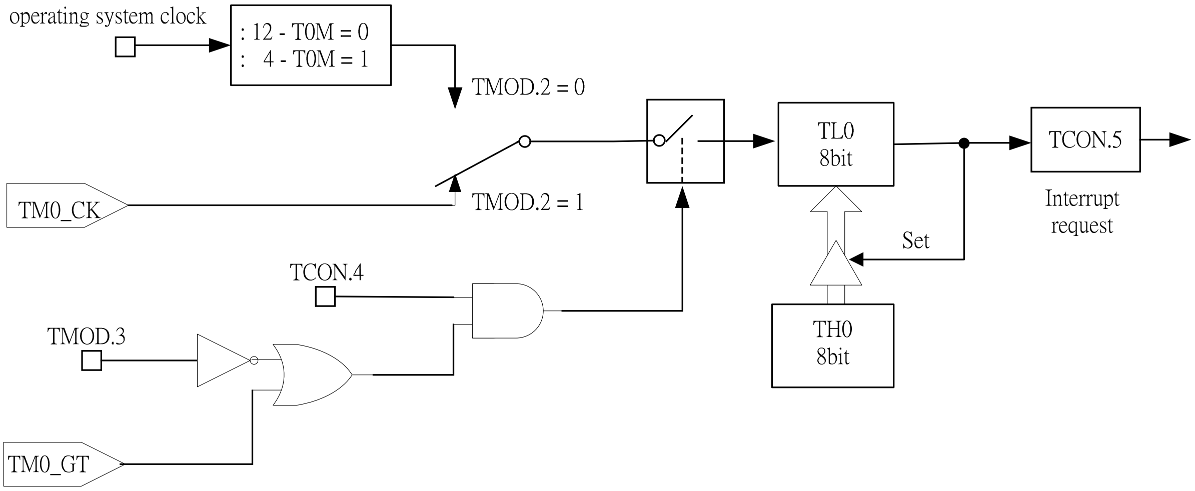


Figure 55: Timer/Counter 0, Mode 2: 8-Bit Timer/Counter With Auto-Reload

Timer 0 - Mode 3

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in figure below. TL0 uses the Timer 0 control bits: CT, GATE, TR0, TM0_GT and TF0. TH0 is locked into a timer function and uses the TR1 and TF1 flags from Timer 1 and controls Timer 1 interrupt. Mode 3 is provided for applications requiring an extra 8-bit timer/counter. When Timer 0 is in Mode 3, Timer 1 can be turned off by switching it into its own Mode 3, or can still be used by the serial channel (UART0/1) as a baud rate generator, or in any application where interrupt from Timer 1 is not required.

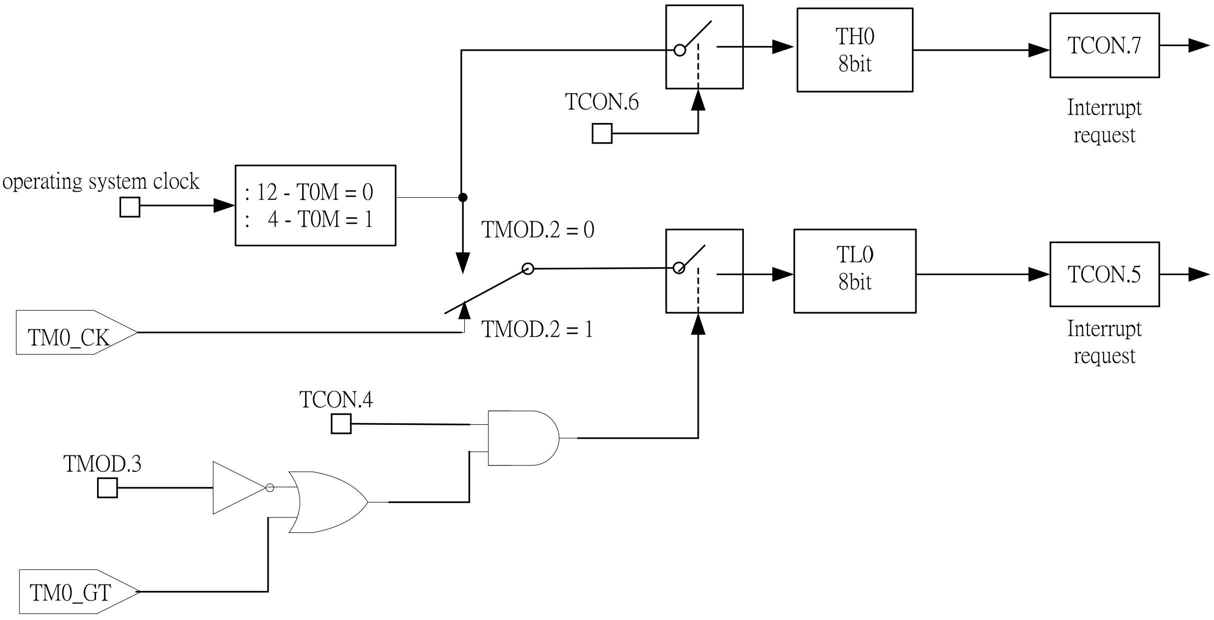


Figure 56: Timer/Counter 0, Mode 3: Two 8-Bit Timers/Counters

Timer 1 - Mode 0

In this mode, the Timer 1 register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, Timer 1 interrupt flag TF1 is set. The counted input is enabled to the Timer 1 when TCON.6 = 1 and either TMOD.6 = 0 or TM1_GT = 1. (Setting TMOD.7 = 1 allows the Timer 1 to be controlled by external input TM1_GT, to facilitate pulse width measurements). The 13-bit register consists of all 8 bits of TH1 and the lower 5 bits of TL1. The upper 3 bits of TL1 are indeterminate and should be ignored.

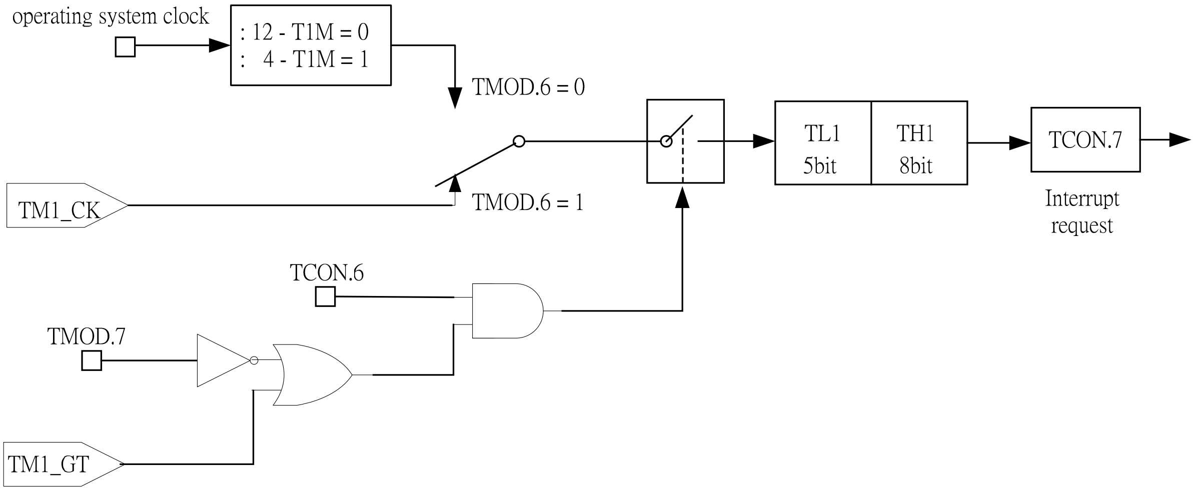


Figure 57: Timer/Counter 1, Mode 0: 13-Bit Timers/Counters

Timer 1 - Mode 1

Mode 1 is the same as Mode 0, except that the timer register is running with all 16 bits. Mode 1 is shown in figure below.

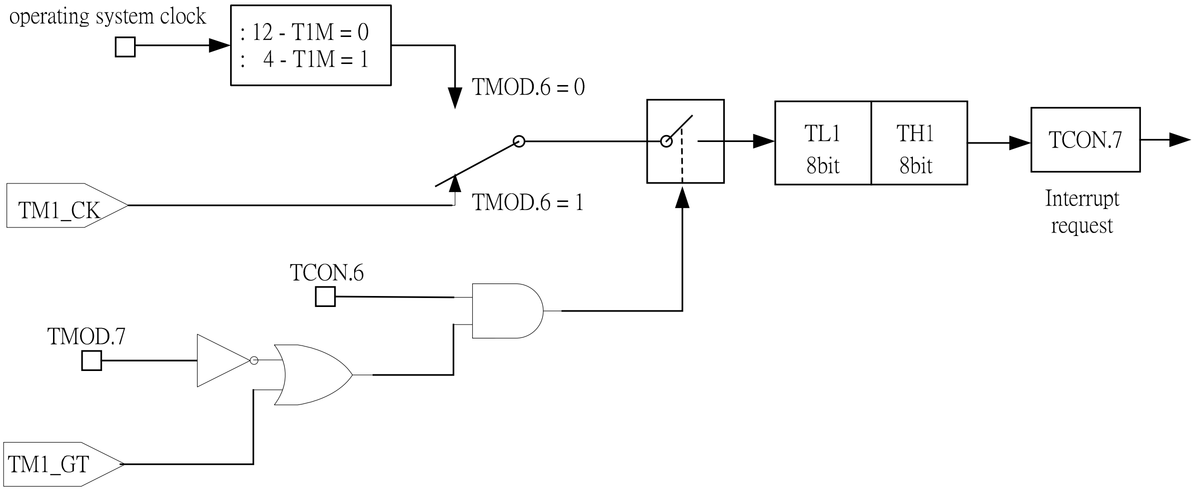


Figure 58: Timer/Counter 1, Mode 1: 16-Bit Timers/Counters

Timer 1 - Mode 2

Mode 2 configures the timer register as an 8-bit counter (TL1) with automatic reloads, as shown in figure below. Overflow from TL1 not only sets TF1, but also reloads TL1 with the contents of TH1, which is loaded by software. The reload leaves TH1 unchanged.

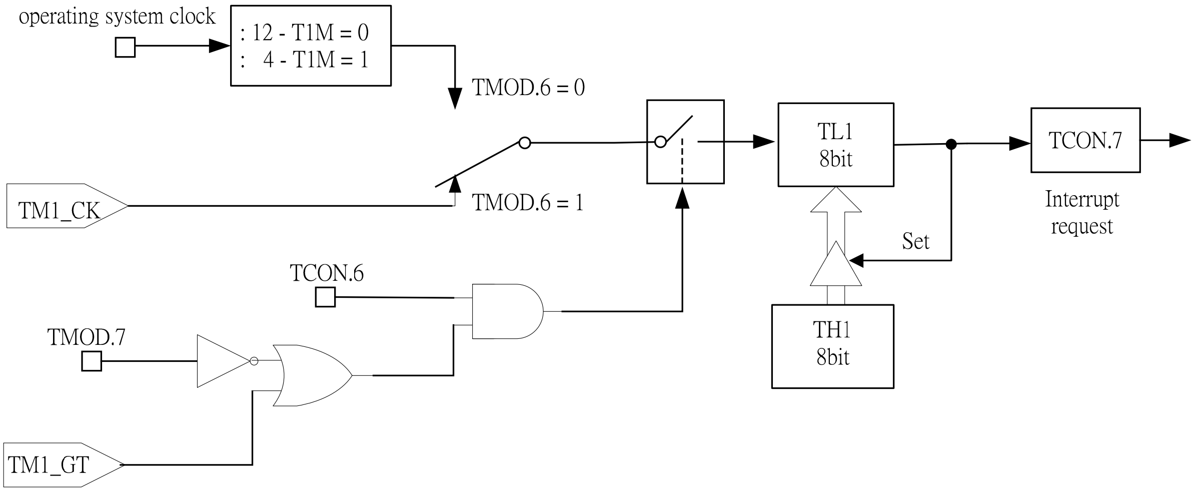


Figure 59: Timer/Counter 1, Mode 2: 8-Bit Timer/Counter With Auto-Reload

Timer 1 - Mode 3

Timer 1 in Mode 3 is held counting. The effect is the same as setting TR1=0.

4.11.3 Timer 2

Timer 2 is fully compatible with the standard 8052 Timer 2. Totally five SFR control the Timer 2 operation: TH2/TL2 (0xCD/0xCC) counter registers, RLDH/RLDL (0xCB/0xCA) capture registers and T2CON (0xC8) control register. Timer 2 works in the three modes selected by T2CON bits as shown in Table 21 below.

RCLK, TCLK	CPRL2	TR2	Timer 2 Operating Mode Description
0	0	1	16-bit auto-reload mode. The Timer 2 overflow sets TF2 bit and the TH2, TL2 registers are reloaded 16-bit value from RLDH, RLDL.
0	1	1	16-bit capture mode. The Timer 2 overflow sets TF2 bit. When the EXEN2=1 the TH2, TL2 register values are stored into RLDH, RLDL while falling edge is detected on TM2_GT pin.
1	X	1	Baud rate generator for the UART0 interface.
X	X	0	Timer 2 is off.

Table 21: Timer 2 Mode of Operation

Timer 2 Configuration Register (T2CON, 0xC8)

Bit	7	6	5	4	3	2	1	0
Name	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	CT2	CPRL2
Reset Value	0x00							

Bit	Name	Access	Description
0	CPRL2	R/W	Capture/reload select. 1: TM2_GT pin falling edge causes capture to occur when EXEN2=1. 0: Automatic reload occurs: on Timer 2 overflow or falling edge of TM2_GT pin when EXEN2=1. When RCLK or TCLK is set this bit is ignored and automatic reload on Timer 2 overflow is forced.
1	CT2	R/W	Timer/counter select. 1: External event counter. Clock source is TM2_CK pin. 0: Timer internally clocked.
2	TR2	R/W	Start/stop Timer 2. 1: Start. 0: Stop.
3	EXEN2	R/W	Enable TM2_GT pin functionality. 1: Allows capture or reload as a result of TM2_GT pin falling edge. 0: Ignore T2EX events.
4	TCLK	R/W	Transmit clock enable. 1: UART0 transmitter is clocked by Timer 2 overflow pulses. 0: UART0 transmitter is clocked by Timer 1 overflow pulses.
5	RCLK	R/W	Receive clock enable. 1: UART0 receiver is clocked by Timer 2 overflow pulses. 0: UART0 receiver is clocked by Timer 1 overflow pulses.
6	EXF2	R/W	Falling edge indicator on TM2_GT pin when EXEN2=1. Must be cleared by software.
7	TF2	R/W	Timer 2 interrupt (overflow) flag. Must be cleared by software. The flag will not be set when either RCLK or TCLK is set.

The Timer 2 interrupt enable register is ET2 bit (IE.5, 0xA8) and its interrupt priority register is PT2 bit (IP.5, 0xB8). The Timer 2 interrupt (overflow) flag is TF2 bit (T2CON.7). The TF2 bit that generates interrupts can be set or cleared by software, with the same result as if they had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be cancelled by software. Please refer to section 4.8 for details.

Timer 2 in Timer Mode

Timer 2 related bits are shown in Figure 60 below. The timer register can be clocked by the external TM2_CK pin or internal operating system clock. If internal operating system clock is used, it can be incremented every 12 or 4 operating system clock periods, configured by T2M bit (CKCON.5, 0x8E).

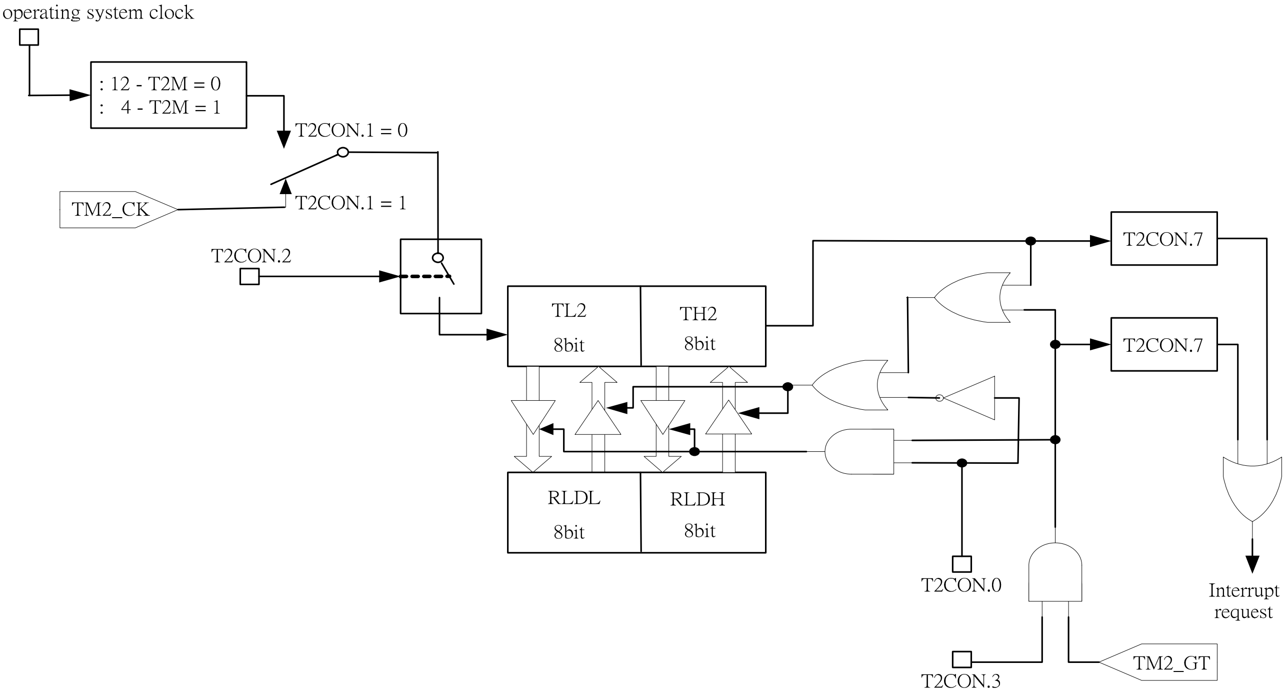


Figure 60: Timer 2 Block Diagram In Timer Mode

Timer 2 in UART0 Baud Rate Generator Mode

Interrupt is also generated at falling edge of TM2_GT pin, while EXEN2 bit is set. This interrupt doesn't set TF2 flag, but EXF2 only and used 0x2B vector. Please see Figure 61 below.

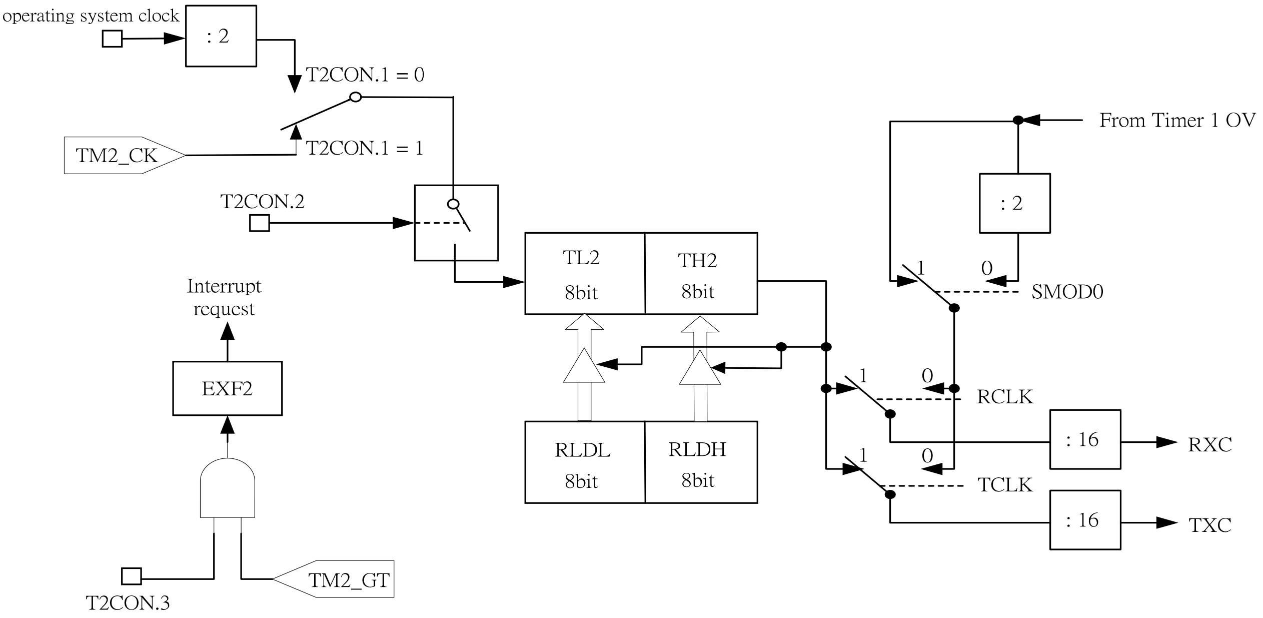


Figure 61: Timer 2 Block Diagram As UART0 Baud Rate Generator

4.11.4 Millisecond Timer

The dedicated Millisecond Timer can be used as a coarse timing reference in software programming (such as TCP/IP protocol stack processing) and is programmable with 1ms resolution. The timeout period can be programmed from 1ms to 1024ms. When Millisecond Timer times out, upon enabled, an interrupt on INT5 will be generated to CPU. The INT5 is shared with Software DMA transfer. For detailed register description related to Millisecond Timer, please refer to section 4.7.2, on Millisecond Timer Register, MSTR (0x0C).

The divider ratio of internal 1ms timing pulse in Millisecond Timer is generated based on the setting of SYSCK_SEL[1:0] pins, which also determine the operating system clock frequency.

SYSCK_SEL [1:0] Setting	Clock Divider Ratio for Internal 1ms Timing Pulse	Description
00	25,000	This sets the operating system clock frequency to be 25Mhz (if internal osclk is used as main clock source) or close to 25Mhz (if LB_CLK is used as clock source). Please note that if LB_CLK input pin is used as clock source, e.g., input frequency = 24Mhz, then the 1ms timing pulse will be about 1.04ms instead.
01	50,000	This sets the operating system clock frequency to be 50Mhz (if internal osclk is used as main clock source) or close to 50Mhz (if LB_CLK is used as clock source). Please note that if LB_CLK input pin is used as clock source, e.g., input frequency = 48Mhz, then the 1ms timing pulse will be about 1.04ms instead.
11	100,000	This sets the operating system clock frequency to be 100Mhz (if internal osclk is used as main clock source) or close to 100Mhz (if LB_CLK is used as clock source). Please note that if LB_CLK input pin is used as clock source, e.g., input frequency = 96Mhz, then the 1ms timing pulse will be about 1.04ms instead.
10	Reserved	Reserved.

Table 22: Millisecond Timer Divider Ratio

4.12 UARTs

AX11025 supports 3 UART interfaces, namely, UART 0, UART 1, and UART 2. The UART 0 and UART 1 have the same functionality as standard 8051 UARTs. Each is full duplex, meaning it can transmit and receive concurrently. Each is receive double-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register. The UART 2 is designed to be maximally compatible with standard 16550. It can communicate with MODEM or other external device (e.g. computer) by using RS-232 protocol. The UART 2 has 16-bytes deep transmit/receive FIFO and its transfer rate can be up to 921600 bps.

4.12.1 UART 0, 1 SFR Register Map

Address	Name	Description
0x98	SCON0	UART 0 Configuration Register.
0x99	SBUF0	UART 0 Buffer Register.
0x87	PCON	Power Configuration Register.
0xA8	IE	Interrupt Enable Register.
0xB8	IP	Interrupt Priority Register.
0xC0	SCON1	UART 1 Configuration Register.
0xC1	SBUF1	UART 1 Buffer Register.

Table 23: UART 0, 1 SFR Register Map

4.12.2 UART 0

The UART 0 has the same functionality as a standard 8051 UART 0. The UART 0 serial port is full duplex, receive double-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register. The UART 0 can operate in 4 modes: one synchronous and three asynchronous modes.

- Mode 0, synchronous mode
- Mode 1, 8-bit UART, variable baud rate, Timer 1 or Timer 2 clock source
- Mode 2, 9-bit UART, fixed baud rate
- Mode 3, 9-bit UART, variable baud rate, Timer 1 or Timer 2 clock source

Mode 2 and 3 has a special feature for multiprocessor communications. The feature is enabled by setting SM02 bit in SCON0 register. The master processor first sends out an address byte, which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM02 = 1, no slave will be interrupted by a data byte. An address byte will interrupt all slaves. The addressed slave will clear its SM02 bit and prepare to receive the data bytes that will be coming. The slaves that were not being addressed leave their SM02 set and ignoring the incoming data.

The UART 0 related registers are: SBUF0 (0x99), SCON0 (0x98), PCON (0x87), IE (0xA8) and IP (0xB8). The UART0 data buffer (SBUF0) consists of two separate registers: transmit and receive registers. A data writes into the SBUF0 sets this data in UART0 output register and starts a transmission. A data reads from SBUF0, reads data from the UART0 receive register. Writing to SBUF0 loads the transmit register, and reading SBUF0 reads a physically separate receive register.

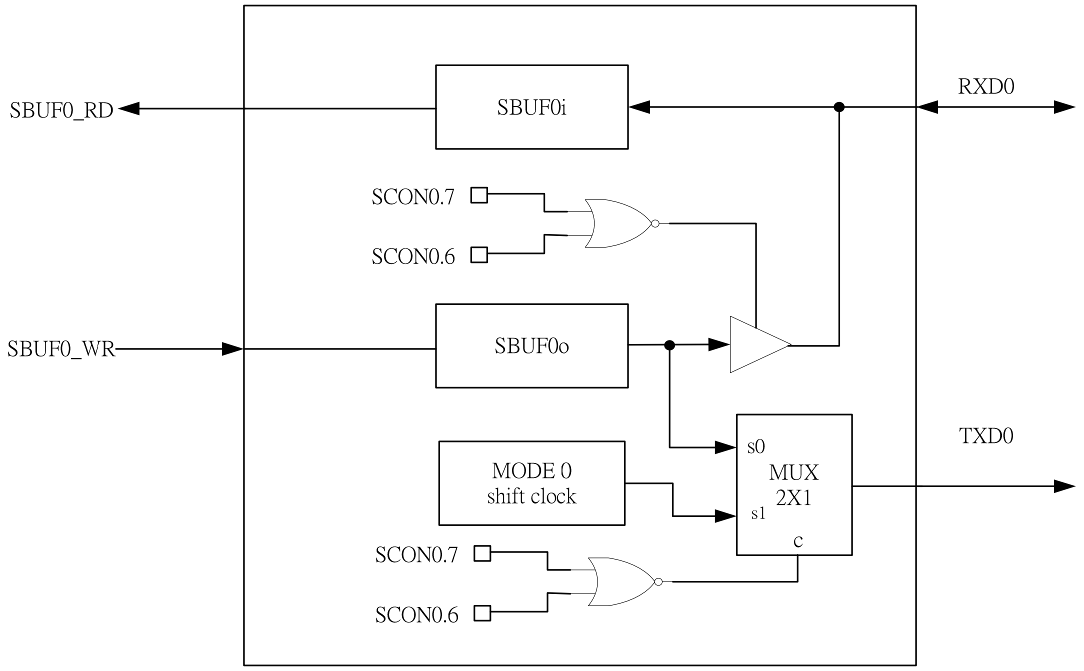


Figure 62: UART 0 Block Diagram

UART 0 Configuration Register (SCON0, 0x98)

Bit	7	6	5	4	3	2	1	0
Name	SM00	SM01	SM02	REN0	TB08	RB08	TI0	RI0
Reset Value	0x00							

<table>
<tr><th>Bit</th><th>Name</th><th>Access</th><th>Description</th></tr>
<tr><td>0</td><td>RI0</td><td>R/W</td><td>Receive interrupt flag, set by hardware after completion of a serial reception. Must be cleared by software.</td></tr>
<tr><td>1</td><td>TI0</td><td>R/W</td><td>Transmit interrupt flag, set by hardware after completion of a serial transfer. Must be cleared by software.</td></tr>
<tr><td>2</td><td>RB08</td><td>R/W</td><td>In Modes 2 and 3 it is the 9th data bit received. In Mode 1, if SM02 is 0, RB08 is the stop bit. In Mode 0 this bit is not used.</td></tr>
<tr><td>3</td><td>TB08</td><td>R/W</td><td>The 9th transmitted data bit in Modes 2 and 3. Set or cleared by the CPU, depending on the function it performs (parity check, multiprocessor communication etc.)</td></tr>
<tr><td>4</td><td>REN0</td><td>R/W</td><td>If set, enables serial reception. Cleared by software to disable reception.</td></tr>
<tr><td>5</td><td>SM02</td><td>R/W</td><td>Enables a multiprocessor communication feature.</td></tr>
<tr><td>6</td><td>SM01</td><td rowspan="2">R/W</td><td rowspan="2">Sets baud rate.
<table>
<tr><th>Mode</th><th>SM00</th><th>SM01</th><th>Description</th><th>UART 0 Baud Rate</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Shift register</td><td>Fsys_clk/12</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>8-bit UART</td><td>Variable.
<table>
<tr><th>SMOD0 bit (PCON.7)</th><th>Baud Rate</th></tr>
<tr><td>0</td><td>T1ov/32 or T2ov/16</td></tr>
<tr><td>1</td><td>T1ov/16 or T2ov/16</td></tr>
</table>
T1ov is Timer 1 overflow rate, and T2ov is Timer 2 overflow rate.</td></tr>
<tr><td>2</td><td>1</td><td>0</td><td>9-bit UART</td><td>
<table>
<tr><th>SMOD0 bit (PCON.7)</th><th>Baud Rate</th></tr>
<tr><td>0</td><td>Fsys_clk//64</td></tr>
<tr><td>1</td><td>Fsys_clk/32</td></tr>
</table></td></tr>
<tr><td>3</td><td>1</td><td>1</td><td>9-bit UART</td><td>Variable, same as Mode 1 above</td></tr>
</table>
Note: Fsys_clk is operating system clock frequency.</td></tr>
<tr><td>7</td><td>SM00</td></tr>
</table>

The UART 0 interrupt enable register is ES0 bit (IE.4, 0xA8) for both RI0 and TI0 interrupt flags in SCON0. Its interrupt priority register is PS0 bit (IP.4, 0xB8). The RI0 and TI0 bits that generates interrupts can be set or cleared by software, with the same result as if they had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be cancelled by software. Please refer to section 4.8 for details.

UART 0 Buffer Register (SBUF0, 0x99)

Bit	7	6	5	4	3	2	1	0
Name	SB0							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	SB0	R/W	A data writes into the SBUF0 sets this data in UART0 output register and starts a transmission. A data reads from SBUF0, reads data from the UART0 receive register.

Mode 0, Synchronous Mode

Pin RXD0 serves as input and output. TXD0 output is a shift clock. The baud rate is fixed at 1/12 of the operating system clock frequency. Eight bits are transmitted with LSB first. Reception is initialized by setting the flags in SCON0 as follows: RI0 = 0 and REN0 = 1.

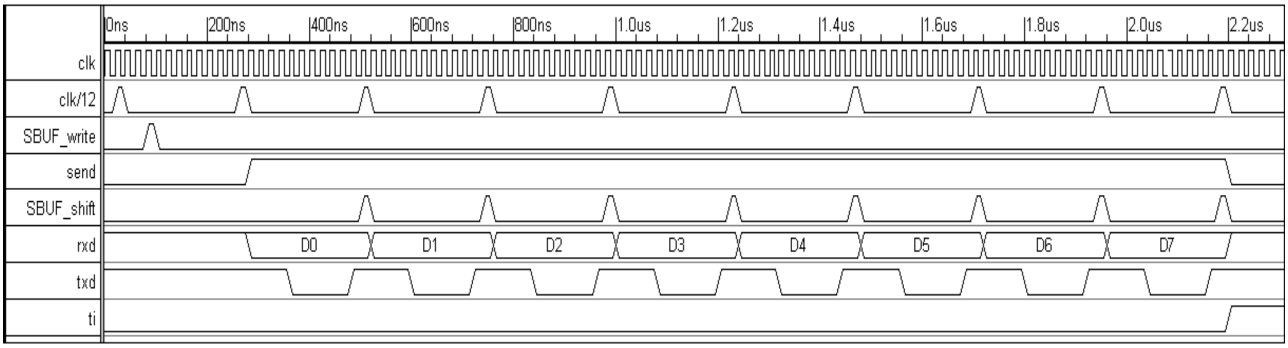


Figure 63: UART 0, Mode 0 Transmit Timing Diagram

Mode 1, 8-bit UART, Variable Baud Rate, Timer 1 or Timer 2 Clock Source

Pin RXD0 serves as input, and TXD0 serves as serial output. 10 bits are transmitted: a start bit (always 0), 8 data bits (LSB first), and a stop bit (always 1). On receive, a start bit synchronizes the transmission, 8 data bits are available by reading SBUF0, and stop bit sets the flag RB08 in the SFR register SCON0. The baud rate is variable and depends from Timer 1 or Timer 2 mode. To enable Timer 2 clocking set the TCLK, RCLK bits located in T2CON (0xC8) register.

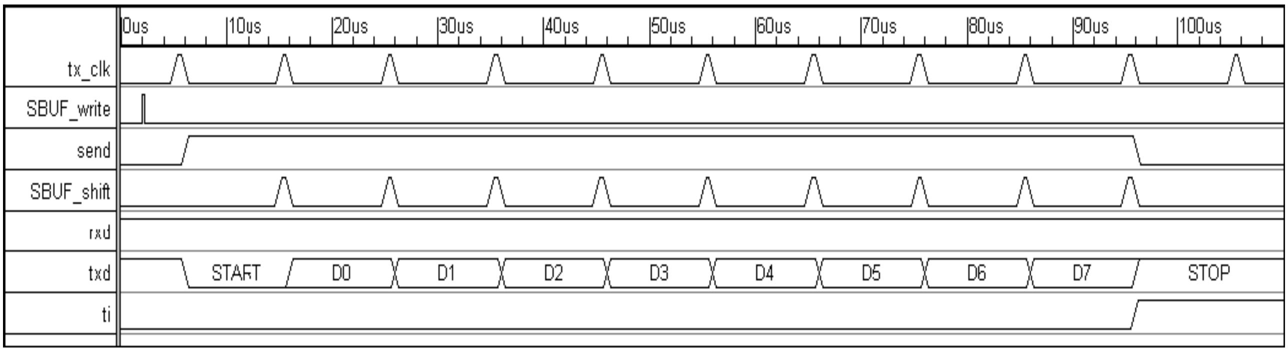


Figure 64: UART 0, Mode 1 Transmit Timing Diagram

Mode 2, 9-bit UART, Fixed Baud Rate

This mode is similar to Mode 1 with two differences. The baud rate is fixed at 1/32 or 1/64 of operating system clock frequency, and 11 bits are transmitted or received: a start bit (0), 8 data bits (LSB first), a programmable 9th bit, and a stop bit (1). The 9th bit can be used to control the parity of the UART0 interface: at transmission, bit TB08 in SCON0 is output as the 9th bit, and at receive, the 9th bit affects RB08 in SCON0.

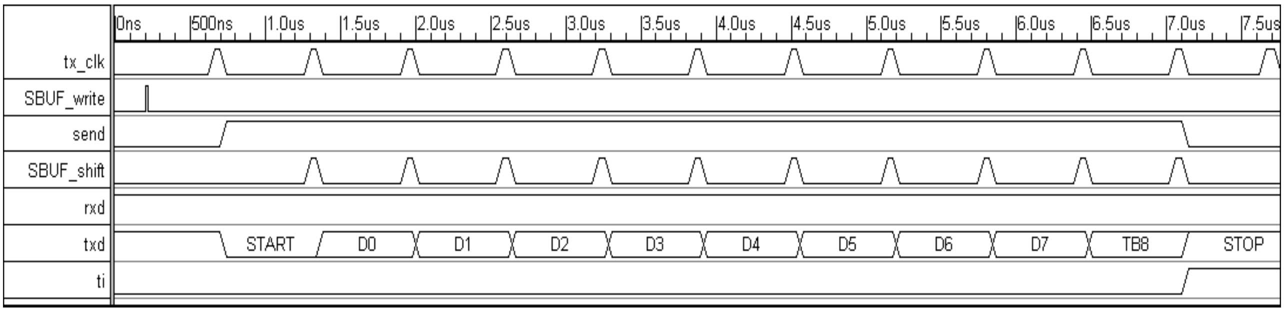


Figure 65: UART 0, Mode 2 Transmit Timing Diagram

Mode 3, 9-bit UART, Variable Baud Rate, Timer 1 or Timer 2 Clock Source

The only difference between Mode 2 and Mode 3 is that the baud rate is a variable in Mode 3. When REN0 = 1 data receiving is enabled. The baud rate is variable and depends from Timer 1 or Timer 2 mode. To enable Timer 2 clocking set the TCLK, RCLK bits located in T2CON (0xC8) register.

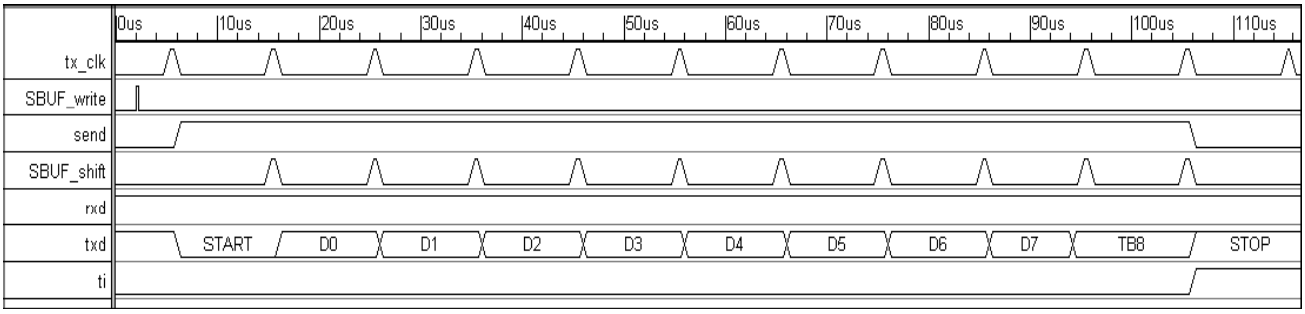


Figure 66: UART 0, Mode 3 Transmit Timing Diagram

4.12.3 UART 1

The UART1 has the same functionality as a standard 8051 UART1. The UART 1 serial port is full duplex, receive double-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register. The UART 1 can operate in 4 modes: one synchronous and three asynchronous modes.

- Mode 0, synchronous mode
- Mode 1, 8-bit UART, variable baud rate, Timer 1 clock source
- Mode 2, 9-bit UART, fixed baud rate
- Mode 3, 9-bit UART, variable baud rate, Timer 1 clock source

Mode 2 and 3 has a special feature for multiprocessor communications. The feature is enabled by setting SM12 bit in SCON1 register. The master processor first sends out an address byte, which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM12 = 1, no slave will be interrupted by a data byte. An address byte will interrupt all slaves. The addressed slave will clear its SM12 bit and prepare to receive the data bytes that will be coming. The slaves that were not being addressed leave their SM12 set and ignoring the incoming data.

The UART1 related registers are: SBUF1 (0xC1), SCON1 (0xC0), PCON (0x87), IE (0xA8) and IP (0xB8). The UART1 data buffer (SBUF1) consists of two separate registers: transmit and receive registers. A data writes into the SBUF1 sets this data in UART1 output register and starts a transmission. A data reads from SBUF1, reads data from the UART1 receive register. Writing to SBUF1 loads the transmit register, and reading SBUF0 reads a physically separate receive register.

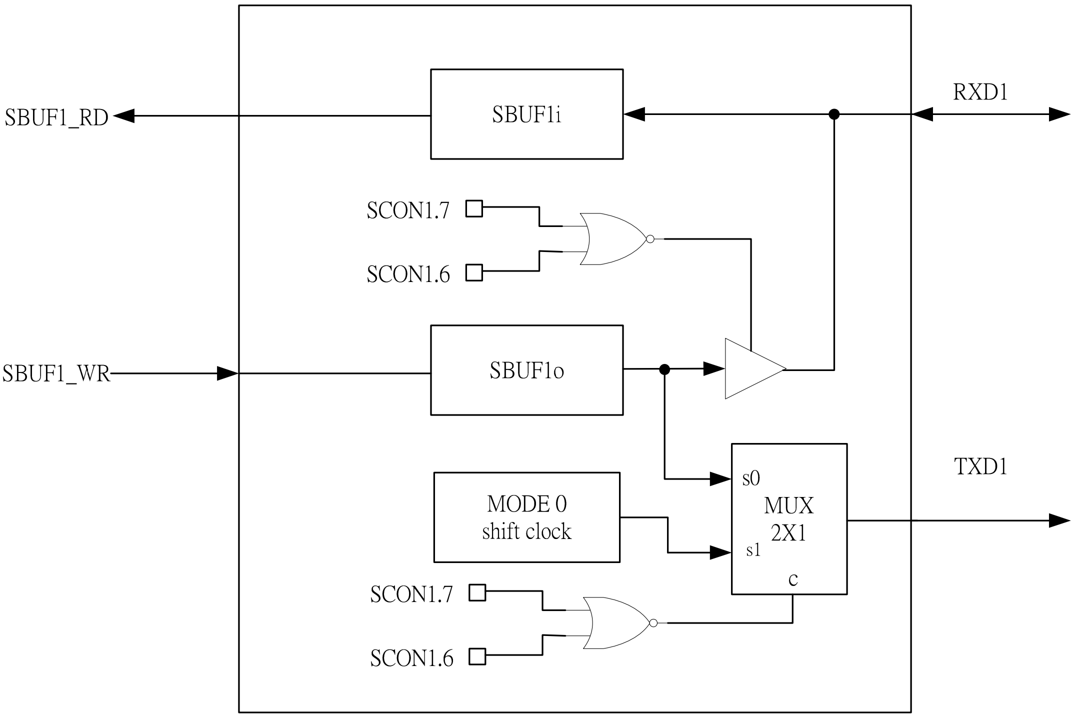


Figure 67: UART 1 Block Diagram

UART 1 Configuration register (SCON1, 0xC0)

Bit	7	6	5	4	3	2	1	0
Name	SM10	SM11	SM12	REN1	TB18	RB18	TI1	RI1
Reset Value	0x00							

<table>
<tr><th>Bit</th><th>Name</th><th>Access</th><th>Description</th></tr>
<tr><td>0</td><td>RI1</td><td>R/W</td><td>Receive interrupt flag, set by hardware after completion of a serial reception. Must be cleared by software.</td></tr>
<tr><td>1</td><td>TI1</td><td>R/W</td><td>Transmit interrupt flag, set by hardware after completion of a serial transfer. Must be cleared by software.</td></tr>
<tr><td>2</td><td>RB18</td><td>R/W</td><td>In Modes 2 and 3 it is the 9th data bit received. In Mode 1, if SM12 is 0, RB18 is the stop bit. In Mode 0 this bit is not used.</td></tr>
<tr><td>3</td><td>TB18</td><td>R/W</td><td>The 9th transmitted data bit in Modes 2 and 3. Set or cleared by the CPU, depending on the function it performs (parity check, multiprocessor communication etc.)</td></tr>
<tr><td>4</td><td>REN1</td><td>R/W</td><td>If set, enables serial reception. Cleared by software to disable reception.</td></tr>
<tr><td>5</td><td>SM12</td><td>R/W</td><td>Enables a multiprocessor communication feature</td></tr>
<tr><td>6</td><td>SM11</td><td rowspan="2">R/W</td><td rowspan="2">Sets baud rate
<table>
<tr><th>Mode</th><th>SM10</th><th>SM11</th><th>Description</th><th>UART 0 Baud Rate</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Shift register</td><td>Fsys_clk/12</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>8-bit UART</td><td>Variable.
<table>
<tr><th>SMOD1 bit (PCON.6)</th><th>Baud Rate</th></tr>
<tr><td>0</td><td>T1ov/32</td></tr>
<tr><td>1</td><td>T1ov/16</td></tr>
</table>
T1ov is Timer 1 overflow rate.</td></tr>
<tr><td>2</td><td>1</td><td>0</td><td>9-bit UART</td><td>
<table>
<tr><th>SMOD1 bit (PCON.6)</th><th>Baud Rate</th></tr>
<tr><td>0</td><td>Fsys_clk//64</td></tr>
<tr><td>1</td><td>Fsys_clk/32</td></tr>
</table></td></tr>
<tr><td>3</td><td>1</td><td>1</td><td>9-bit UART</td><td>Variable, same as Mode 1 above</td></tr>
</table>
Note: Fsys_clk is operating system clock frequency.</td></tr>
<tr><td>7</td><td>SM10</td></tr>
</table>

The UART 1 interrupt enable register is ES1 bit (IE.6, 0xA8) for both RI1 and TI1 interrupt flags in SCON1. Its interrupt priority register is PS1 bit (IP.6, 0xB8). The RI1 and TI1 bits that generates interrupts can be set or cleared by software, with the same result as if they had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be cancelled by software. Please refer to section 4.8 for details.

UART 1 Buffer Register (SBUF1, 0xC1)

Bit	7	6	5	4	3	2	1	0
Name	SB1							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	SB1	R/W	A data writes into the SBUF1 sets this data in UART1 output register and starts a transmission. A data reads from SBUF1, reads data from the UART1 receive register.

Mode 0, Synchronous Mode

Pin RXD1 serves as input and output. TXD1 output is a shift clock. The baud rate is fixed at 1/12 of the operating system clock frequency. Eight bits are transmitted with LSB first. Reception is initialized by setting the flags in SCON1 as follows: RI1 = 0 and REN1 = 1.

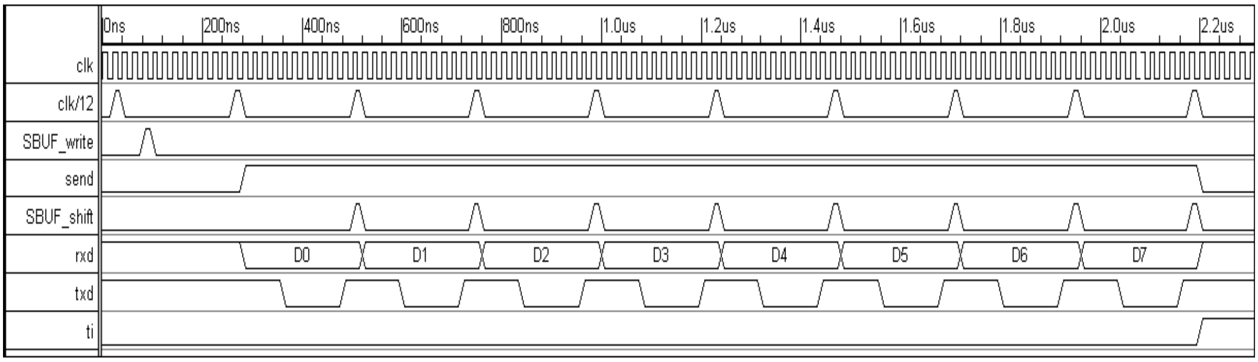


Figure 68: UART 1, Mode 0 Transmit Timing Diagram

Mode 1, 8-bit UART, Variable Baud Rate, Timer 1 Clock Source

Pin RXD1 serves as input, and TXD1 serves as serial output. 10 bits are transmitted: a start bit (always 0), 8 data bits (LSB first), and a stop bit (always 1). On receive, a start bit synchronizes the transmission, 8 data bits are available by reading SBUF1, and stop bit sets the flag RB18 in the SCON1. The baud rate is variable and depends from Timer 1.

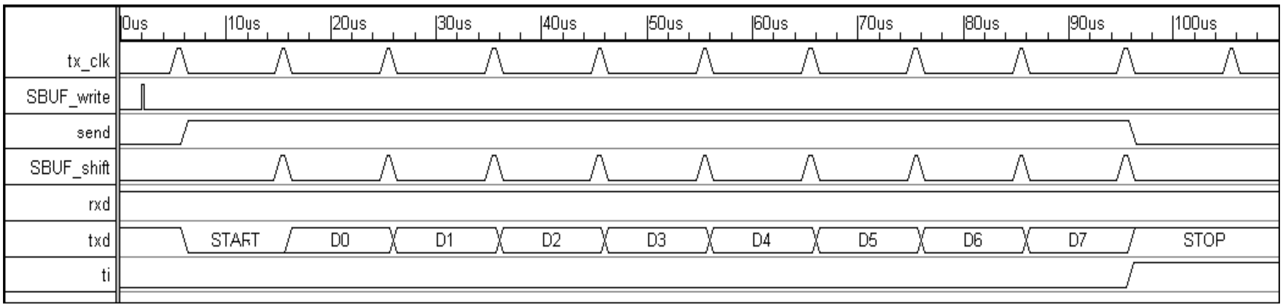


Figure 69: UART 1, Mode 1 Transmit Timing Diagram

Mode 2, 9-bit UART, Fixed Baud Rate

This mode is similar to Mode 1 with two differences. The baud rate is fixed at 1/32 or 1/64 of CLK clock frequency, and 11 bits are transmitted or received: a start bit (0), 8 data bits (LSB first), a programmable 9th bit, and a stop bit (1). The 9th bit can be used to control the parity of the UART 1 interface: at transmission, bit TB18 in SCON1 is output as the 9th bit, and at receive, the 9th bit affects RB08 in SCON1.

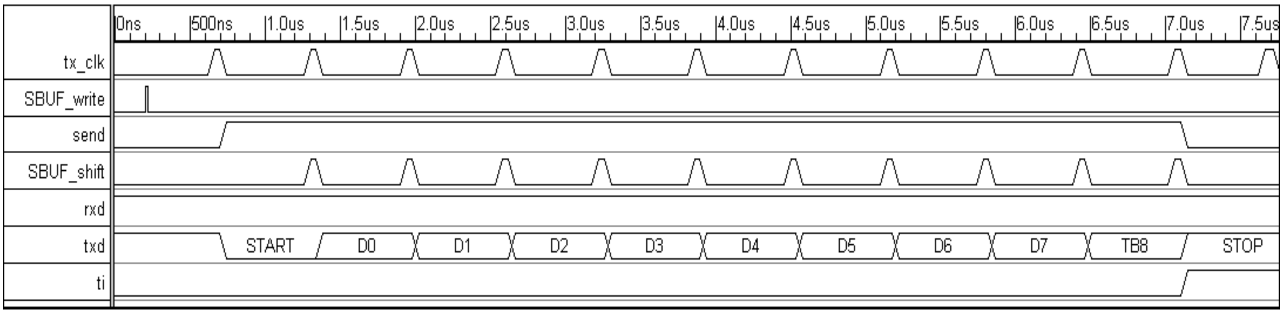


Figure 70: UART1, Mode 2 Transmit Timing Diagram

Mode 3, 9-bit UART, Variable Baud Rate, Timer 1 Clock Source

The only difference between Mode 2 and Mode 3 is that the baud rate is a variable in Mode 3. When REN1=1 data receiving is enabled. The baud rate is variable and depends from Timer 1.

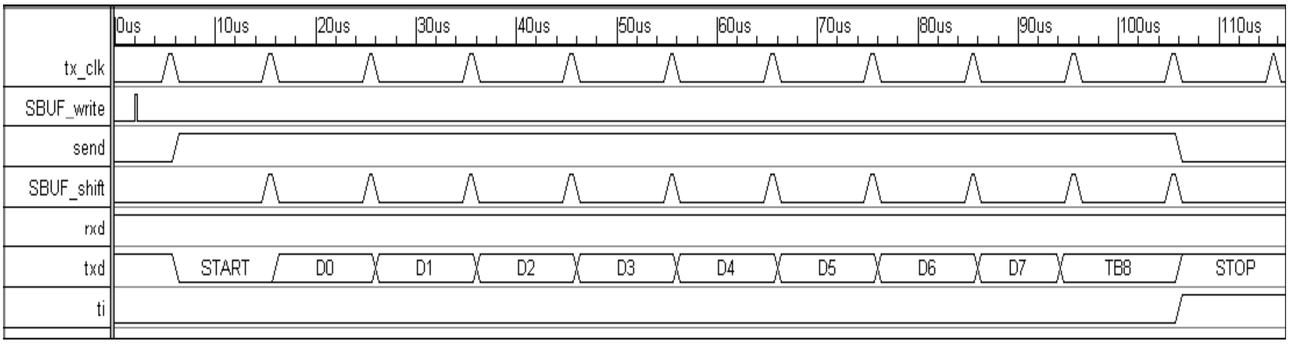


Figure 71: UART 1, Mode 3 Transmit Timing Diagram

4.12.4 UART 2

The UART 2 of AX11025 is designed to be maximally compatible with standard 16550. It provides serial communication capabilities, which allow communication with modem or other external device (e.g. computer) by using RS232 protocol. It contains 16-bytes deep transmit FIFO and receive FIFO and its transfer rate can be up to 921600 bps. It includes a programmable baud rate generator capable of dividing the operating system clock by (27*N), where N = 1~65535, for generating wide range of baud rate for the internal transmitter/receiver logic.

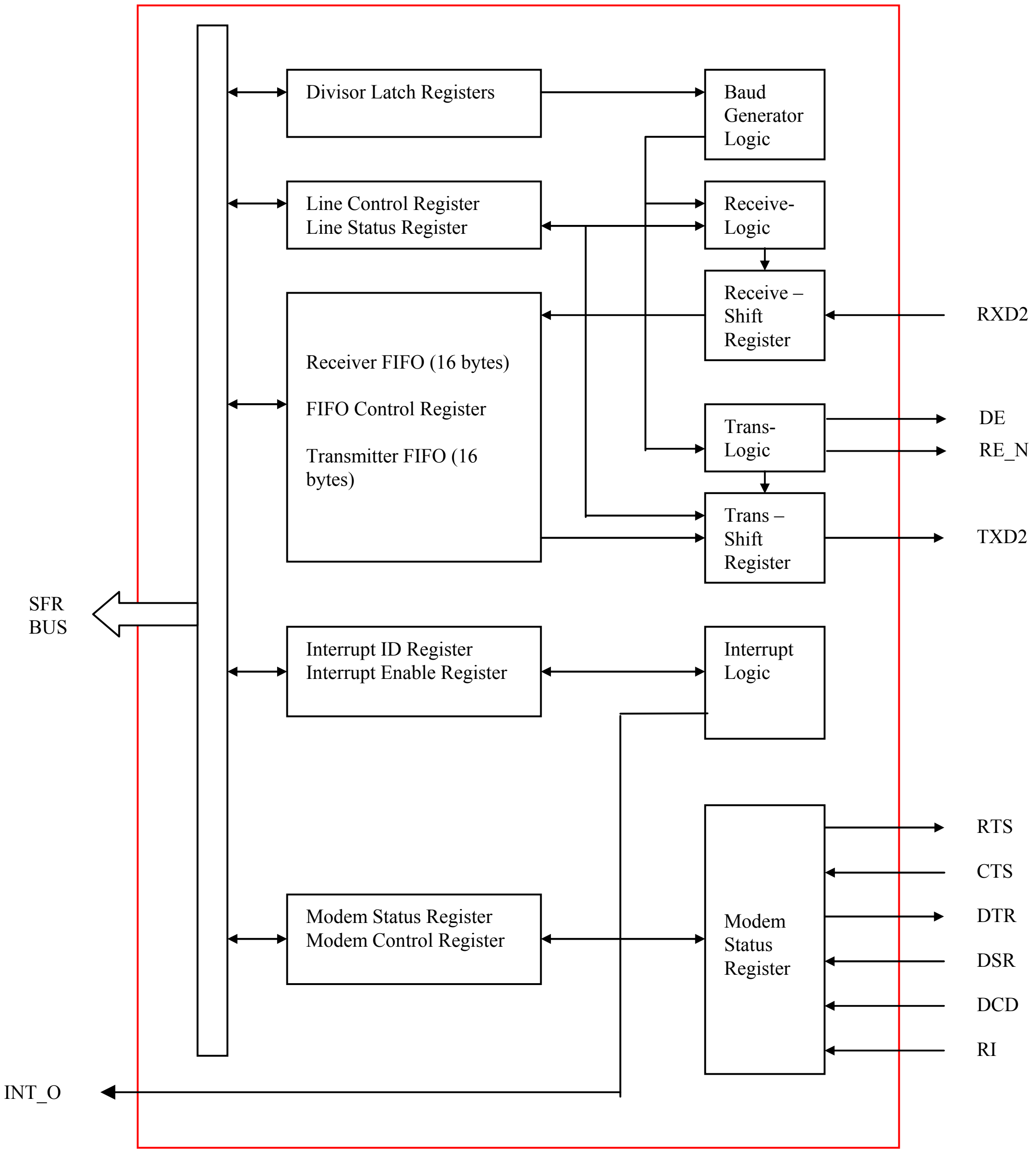


Figure 72: UART 2 Block Diagram

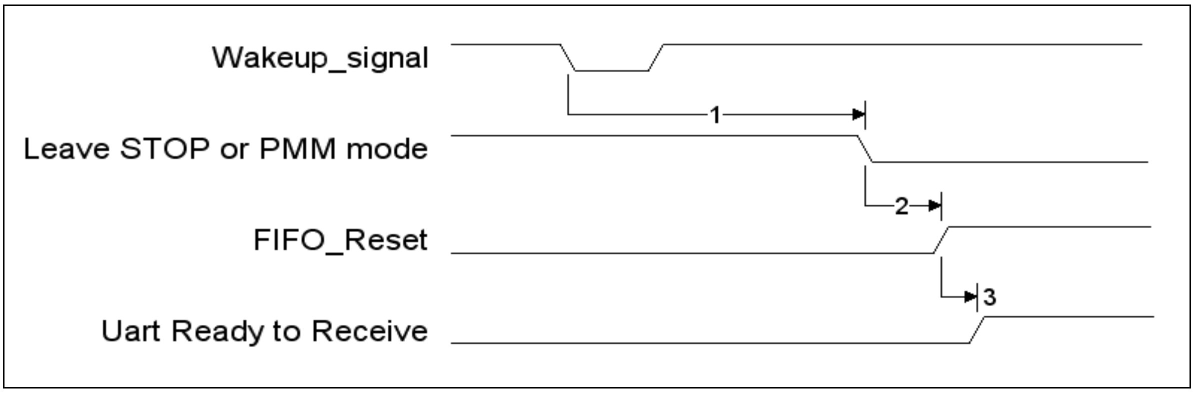
The main features of UART 2 are listed below,

- 16 bytes deep receive and transfer FIFO
- Support up to 921600 bps baud
- Detection of bad data in the receiver FIFO
- Full-duplex asynchronous channel
- Automatic send data control (ASDC) for automatically transmitter/receiver enable control for RS-485
- Modem control functions (CTS, RTS, DSR, DTR, RI and DCD)
- Fully programmable serial interface
  - Even, odd, no parity bit generation and detection
  - 5, 6, 7, 8 data bit
  - 1, 1.5, 2 stop bit generation
- Line break generation and detection
- Internal diagnostic capabilities (loopback controls, break, parity, overrun and framing error)
- Transmit, receive, line status, and data set interrupts independently controlled
- Complete status reporting capabilities
- Remote wakeup by detecting falling-edge transition (start bit) on RXD2 pin or falling-edge transition on RI pin

Wakeup Function

The UART 2 supports remote wakeup function. Upon detecting wakeup signals, it can wake up the CPU of AX11025 from PMM or STOP mode (with or without OSC/PLL turned off). The wakeup signals can be either a falling-edge transition (start bit) on RXD2 pin when receiving a byte of serial data or a falling-edge transition on RI pin from a modem ring signal. Note that because baud rate of UART 2 can be incorrect during PMM or STOP mode, therefore, the receiver FIFO normally requires a reset after wakeup to ensure proper operation.

Upon detecting the wakeup signal, normally the system clock may need some time to become stable, so the UART 2 may not be able to receive serial data properly during this time period. Following describes the timing requirements for awaking from PMM and STOP modes.



Case 1: Wakeup from STOP mode with 25Mhz oscillator and PLL completely stopped

During this STOP mode with OSC/PLL stopped, the first receive serial data byte on RXD2 or a low pulse on RI is being used as wakeup event to awake up the CPU and will not be received correctly into receiver FIFO. To correctly receive the 2nd serial data byte on RXD2, it should be separated by at least 100us (operating system clock = 100Mhz), 200us (operating system clock = 50Mhz), or 400us (operating system clock = 25Mhz), respectively, from the first wakeup byte. Because after detecting the wakeup events, the internal system clock will need abovementioned time frame to resume running (to allow the internal OSC/PLL to become stabilized), any RXD2 activity within the mentioned time range will not be received properly. Also, after awaking up the CPU, the software should generate a FIFO reset command to reset the receiver FIFO via RFR bit (HS_FCR.1) after the first wakeup byte has been completely received through.

Symbol	Description	Operating System Clock	Min	Typ	Max	Unit
1	Falling-edge transition of the wakeup signal to the CPU leaving STOP mode.	100 Mhz	100			us
		50 Mhz	200			us
		25 Mhz	400			us
2	The CPU leaving STOP mode to the time software should generate receiver FIFO reset to allow the wakeup byte to be completely received through.		3			byte time (1)
3	The time software should generate receiver FIFO reset to the UART 2 ready to receive data.		10			system clocks

Note: 1. The byte time is the time period to receive one serial data byte.

Case 2: Wakeup from STOP mode with 25Mhz oscillator and PLL still running

Symbol	Description	Operating System Clock	Min	Typ	Max	Unit
1	Falling-edge transition of the wakeup signal to the CPU leaving STOP mode.	100 Mhz	100			ns
		50 Mhz	200			ns
		25 Mhz	400			ns
2	The CPU leaving STOP mode to the time software should generate receiver FIFO reset to allow the wakeup byte to be completely received through.		3			byte time (1)
3	The time software should generate receiver FIFO reset to the UART 2 ready to receive data.		10			system clocks

Note: 1. The byte time is the time period to receive one serial data byte.

Case 3: Wakeup from PMM mode with switchback enabled

Symbol	Description	Operating System Clock	Min	Typ	Max	Unit
1	Falling-edge transition of the wakeup signal to the CPU leaving PMM mode.	100 Mhz	5			us
		50 Mhz	10			us
		25 Mhz	20			us
2	The CPU leaving PMM mode to the time software should generate receiver FIFO reset to allow the wakeup byte to be completely received through.		3			byte time (1)
3	The time software should generate receiver FIFO reset to the UART 2 ready to receive data.		10			system clocks

Note: 1. The byte time is the time period to receive one serial data byte.

UART 2 SFR Register Map

Address	Name		Access	Description
0xE1	HS_RTD	HS_RBR	RO	Receiver Buffer Register: Receiver FIFO output.
0xE1		HS _THR	WO	Transmitter Holding Register: Transmit FIFO input.
0xE1		HS _DLLR	R/W	Divisor Latch Low Register: The LSB of the Divisor Latch Register. This register can be accessed after setting the 7th(DLAB) bit of the Line Control Register to 1.
0xE2	HS_ID	HS _IER	R/W	Interrupt Enable Register: Enable/Mask Interrupts generated by UART.
0xE2		HS _DLHR	R/W	Divisor Latch High Register: The MSB of the Divisor Latch Register. This register can be accessed after setting the 7th(DLAB) bit of the Line Control Register to 1.
0xE3	HS_IF	HS _IIR	R	Interrupt Identification Register: Get interrupt information.
0xE3		HS _FCR	W	FIFO Control Register: Control FIFO options.
0xE4	HS _LCR		R/W	Line Control Register: Control connection.
0xE5	HS _MCR		W	Modem Control Register: Control modem.
0xE6	HS _LSR		R	Line Status Register: Status information.
0xE7	HS _MSR		R	Modem Status Register: Modem Status.

Note: The Divisor Latch Register is 16-bit register. It can be accessed after setting the 7th(DLAB) bit of the Line Control Register to 1. After finish setting Divisor Latch Register, please set the 7th(DLAB) bit of the Line Control Register to 0.

Table 24: UART 2 SFR Register Map

HS Receive Buffer Register (HS_RBR, 0xE1)

Bit	7	6	5	4	3	2	1	0
Name	RBR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	RBR	RO	Receive Buffer Register only active at Reading.

HS Transmit Holding Register (HS THR, 0xE1)

Bit	7	6	5	4	3	2	1	0
Name	THR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	THR	WO	Transmit Holding Register.

HS Divisor Latch Low Register (HS DLLR, 0xE1)

Bit	7	6	5	4	3	2	1	0
Name	DLLR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	DLLR	R/W	The LSB (7:0 bits) of the Divisor Latch Register. This register can be accessed after setting the 7th bit of LCR to '1'. You should set this bit to '0' after finish setting the divisor latches. Divisor Latch Register is a 2 bytes register. The HS_DLHR (Divisor Latch High Register) register in conjunction with HS_DLLR (Divisor Latch Low Register) forms a 16-bit Divisor Latch Register that contains the baud rate divisor for the UART 2. The internal counter starts

to work when the LSB of Divisor Latch Register is written, so when setting the divisor, write the MSB first and the LSB last. The output baud rate is equal to the operating system clock frequency divided by (27 times the value of the baud rate divisor), shown as follows.

$$\text{Baud Rate} = \frac{\text{Operating System Clock Frequency}}{\mathbf{27} \; * \; \text{Divisor Latch Register}}$$

Following is suggested Divisor Latch Register value for different baud rate,

Operating System Clock = 25 Mhz			
Baud Rate	**Divisor Latch Register**	**Actual Baud**	**Tolerance %**
921600	0x0001	925926	0.47%
115200	0x0008	115741	0.47%
57600	0x0010	57870	0.47%
38400	0x0018	38580	0.47%
19200	0x0030	19290	0.47%
9600	0x0060	9645	0.47%
7200	0x0081	7178	0.31%
4800	0x00c1	4798	0.05%
3600	0x0101	3603	0.08%

Operating System Clock = 50 Mhz			
Baud Rate	**Divisor Latch Register**	**Actual Baud**	**Tolerance %**
921600	0x0002	925926	0.47%
115200	0x0010	115741	0.47%
57600	0x0020	57870	0.47%
38400	0x0030	38580	0.47%
19200	0x0060	19290	0.47%
9600	0x00c1	9595	0.05%
7200	0x0101	7206	0.08%
4800	0x0182	4798	0.05%
3600	0x0202	3603	0.08%

Operating System Clock = 100 Mhz			
Baud Rate	**Divisor Latch Register**	**Actual Baud**	**Tolerance %**
921600	0x0004	925926	0.47%
115200	0x0020	115741	0.47%
57600	0x0040	57870	0.47%
38400	0x0060	38580	0.47%
19200	0x00c1	19190	0.05%
9600	0x0182	9595	0.05%
7200	0x0202	7206	0.08%
4800	0x0304	4798	0.05%
3600	0x0405	3599	0.02%

HS Interrupt Enable Register (HS IER, 0xE2)

Bit	7	6	5	4	3	2	1	0
Name	Reserved		WS	WE	MSI	RLSI	THRI	RDI
Reset Value	0x00							

Bit	Name	Access	Description
0	RDI	R/W	Received Data available interrupt. 1: Enabled. 0: Disabled.
1	THRI	R/W	Transmitter Holding Register empty Interrupt. 1: Enabled. 0: Disabled.
2	RLSI	R/W	Receiver Line Status Interrupt. 1: Enabled. 0: Disabled.
3	MSI	R/W	Modem Status Interrupt. 1: Enabled. 0: Disabled.
4	WE	RW	Wakeup Enable to wakeup the CPU from PMM or STOP mode. Whenever RXD2 or RI pin become active (high to low transition), the UART 2 will generate interrupt to INT 6 to wakeup the CPU (and to re-enable OSC/PLL and system clock). 1: Enabled. 0: Disabled.
5	WS	CR	Wakeup Status. 1: When reading back "1", this bit indicate that the CPU is awaked up by UART 2. 0: This bit will be cleared automatically after software reads it.
7:6	Reserved		

HS Divisor Latch High Register (HS DLHR, 0xE2)

Bit	7	6	5	4	3	2	1	0
Name	DLHR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	DLHR	R/W	The MSB (15:8 bits) of the Divisor Latch Register. See HS_DLLR for details.

HS Interrupt Identification Register (HS_IIR, 0xE3)

Bit	7	6	5	4	3	2	1	0
Name	Reserved				BIT3	BIT2	BIT1	BIT0
Reset Value	1	1	0	0	0	0	0	1

Bit	Name	Access	Description
0	BIT0	RO	Please see Table 25 below for more description.
1	BIT1	RO	
2	BIT2	RO	
3	BIT3	RO	
4	Reserved	RO	Always 0.
5		RO	Always 0.
6		RO	Always 1.
7		RO	Always 1.

Bit3	Bit2	Bit1	Bit0	Priority	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	0	1	-	None	None	-
0	1	1	0	1st	Receiver Line Status	Parity, Overrun or Framing errors or Break Interrupt	Reading the Line Status Register
0	1	0	0	2nd	Receiver Data available	FIFO trigger level reached	FIFO drops below trigger level
1	1	0	0	2nd	Timeout Indication	There's at least 1 character in the FIFO but no character has been input to the FIFO or read from it for the last 4 Char times	Reading from the FIFO (Receiver Buffer Register)
0	0	1	0	3rd	Transmitter Holding Register empty	Transmitter Holding Register Empty	Writing to the Transmitter Holding Register (HS_THR) or reading HS_IIR
0	0	0	0	4th	Modem Status	CTS, DSR, RI or DCD	Reading the Modem status register (HS_MSR)

Table 25: UART2 Interrupt Identification Register

HS FIFO Control Register (HS_FCR, 0xE3)

Bit	7	6	5	4	3	2	1	0
Name	FITL		Reserved			TFR	RFR	FIFOE
Reset Value	1100_0001							

Bit	Name	Access	Description
0	FIFOE	WO	This UART only supports FIFO mode, so always write 1 to this bit.
1	RFR	W1	Writing 1 to this bit clears the Receiver FIFO and resets its logic. But it doesn't clear the shift register, receiving of the current character continues. This bit will be cleared by chip hardware automatically.
2	TFR	W1	Writing 1 to this bit clears the Transmitter FIFO and resets its logic. The shift register is not cleared, transmitting of the current character continues. This bit will be cleared by chip hardware automatically.
5:3	Reserved	WO	
7:6	FITL	WO	FIFO Trigger level: Define the Receiver FIFO interrupt level. '00': 1 byte. '01': 4 bytes. '10': 8 bytes. '11': 14 bytes.

HS Line Control Register (HS LCR, 0xE4)

Bit	7	6	5	4	3	2	1	0
Name	DLAB	BCB	SPB	EPS	PE	NSB	NBPC	
Reset Value	0	0	0	0	0	0	11	

Bit	Name	Access	Description
0:1	NBPC	R/W	The number of bits per character in each transmitted or received serial character. '00': 5 bits. '01': 6 bits. '10': 7 bits. '11': 8 bits.
2	NSB	R/W	Specify the number of generated stop bits. Note that the receiver always checks the first stop bit only. 1: 1.5 stop bits when 5-bit character length selected and 2 bits otherwise. 0: 1 stop bit.
3	PE	R/W	Parity Enable. 1: Parity bit is generated on each outgoing character and is checked on each incoming one. 0: No parity.
4	EPS	R/W	Even Parity select 1: Even number of logic 1 is transmitted in each word. 0: Odd number of logic1 is transmitted and checked in each word (data and parity combined). In other words, if the data has an even number of logic 1 in it, then the parity bit is logic 1.
5	SPB	R/W	Stick Parity bit. 1: If bits 3 and 4 are logic 1, the parity bit is transmitted and checked as logic 0. If bit 3 is logic 1 and bit 4 is logic 0 then the parity bit is transmitted and checked as logic 1. 0: Stick Parity disabled.
6	BCB	R/W	Break Control bit. 1: The serial out is forced into logic 0 (break state). 0: Break is disabled.
7	DLAB	R/W	Divisor Latch Access bit. 1: The divisor latches can be accessed. 0: The normal registers are accessed.

HS Modem Control Register (HS MCR, 0xE5)

Bit	7	6	5	4	3	2	1	0
Name	DEREC		Reserved	LOOPB	OUT2	OUT1	RTS	DTR
Reset Value	0x00							

Bit	Name	Access	Description
0	DTR	WO	Data Terminal Ready (DTR) Signal Control. 1: DTR will output logic 0. 0: DTR will output logic 1.
1	RTS	WO	Request To Send (RTS) signal control. 1: RTS will output logic 0. 0: RTS will output logic 1.
2	OUT1	WO	Out1. In Loopback mode, connected to Ring Indicator (RI) signal input.
3	OUT2	WO	Out2. In Loopback mode, connected to Data Carrier Detect (DCD) signal input.
4	LOOPB	WO	Loopback mode. 1: Loopback mode. When in loopback mode, the serial output signal (TXD2) is set to logic 1. The signal of the transmitter shift register is internally connected to the input of the receiver shift register.

<table>
<tr><td></td><td></td><td></td><td>The following connections are made during loopback mode:
DTR → DSR
RTS → CTS
Out1 → RI
Out2 → DCD
0: Normal operation.</td></tr>
<tr><td>5</td><td>Reserved</td><td></td><td></td></tr>
<tr><td>7:6</td><td>DEREC</td><td>WO</td><td>Transceiver Driver Enable (DE pin) and Receiver Enable Control (RE_N pin):
<table>
<tr><th>DEREC</th><th>Mode</th><th>DE pin</th><th>RE_N pin</th></tr>
<tr><td>00</td><td>Sleep</td><td>DE keeps output logic 0</td><td>RE_N keeps output logic 1</td></tr>
<tr><td>01</td><td>Single Twisted Pair Half Duplex</td><td>In this mode DE will automatically output logic 1 whenever TX FIFO is non-empty</td><td>RE_N will output logic 1 whenever transmitting data and output logic 0 when it is not transmitting.</td></tr>
<tr><td>10</td><td>Single Twisted Pair Half Duplex or Double Twisted Pair Full Duplex (Slave)</td><td>In this mode DE will automatically output logic 1 whenever TX FIFO is non-empty</td><td>RE_N keeps output logic 0</td></tr>
<tr><td>11</td><td>Double Twisted Pair Full Duplex (Master)</td><td>DE keeps output logic 1</td><td>RE_N keeps output logic 0</td></tr>
</table>
Note: The DE pin is high active, and RE_N pin is low active.</td></tr>
</table>

HS Line Status Register (HS LSR, 0xE6)

Bit	7	6	5	4	3	2	1	0
Name	FERR	TEMT	THRE	BI	FE	PE	OE	DR
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	DR	RO	Data Ready (DR) indicator. 1: At least one character has been received and is in the FIFO. 0: No characters in the FIFO.
1	OE	CR	Overrun Error (OE) indicator. 1: If the FIFO is full and another character has been received in the receiver shift register. If another character is starting to arrive, it will overwrite the data in the shift register but the FIFO will remain intact. The bit is cleared upon reading from the register. This will generates Receiver Line Status interrupt. 0: No overrun state.
2	PE	CR	Parity Error (PE) indicator. 1: The character that is currently at the top of the FIFO has been received with parity error. The bit is cleared upon reading from the register. This will generate Receiver Line Status interrupt. 0: No parity error in the current character.
3	FE	CR	Framing Error (FE) indicator. 1: The received character at the top of the FIFO did not have a valid stop bit. Of course, generally, it might be that all the following data is corrupted. The bit is cleared upon reading from the register. This will generate Receiver Line Status interrupt. 0: No framing error in the current character.
4	BI	CR	Break Interrupt (BI) indicator 1: A break condition has been reached in the current character. The break occurs when the line is held in logic 0 for a time of one character (start bit + data + parity + stop bit). In that case, one zero character enters the FIFO and the UART2 waits for a valid start bit to receive next character. The bit is cleared upon reading from the register. This will generate Receiver Line Status interrupt.

			0: No break condition in the current character.
5	THRE	RO	Transmit FIFO is empty. 1: The transmitter FIFO is empty. This will generate Transmitter Holding Register Empty interrupt. The bit is cleared when data is being written to the transmitter FIFO. 0: Otherwise.
6	TEMT	RO	Transmitter Empty indicator. 1: Both the transmitter FIFO and transmitter shift register are empty. The bit is cleared when data is being written to the transmitter FIFO. 0: Otherwise.
7	FERR	CR	1: At least one parity error, framing error or break indications have been received and are inside the FIFO. The bit is cleared upon reading from the register. 0: Otherwise.

HS Modem Status Register (HS MSR, 0xE7)

Bit	7	6	5	4	3	2	1	0
Name	DCD	RI	DSR	CTS	DDCD	TERI	DDSR	DCTS
Reset Value	0	0	0	0	0	0	0	0

Bit	**Name**	**Access**	**Description**
0	DCTS	CR	Delta Clear To Send (DCTS) indicator. 1: The CTS line has changed its state.
1	DDSR	CR	Delta Data Set Ready (DDSR) indicator. 1: The DSR line has changed its state.
2	TERI	CR	Trailing Edge of Ring Indicator (TERI) detector. 1: The RI line has changed its state from low to high state.
3	DDCD	CR	Delta Data Carrier Detect (DDCD) indicator. 1: The DCD line has changed its state.
4	CTS	RO	Complement of the CTS input or equals to RTS in Loopback mode.
5	DSR	RO	Complement of the DSR input or equals to DTR in Loopback mode.
6	RI	RO	Complement of the RI input or equals to Out1 in Loopback mode.
7	DCD	RO	Complement of the DCD input or equals to Out2 in Loopback mode.

4.13 GPIO

The AX11025 supports four 8-bit bi-directional, open-drain, general purpose input and output ports, namely, P0 [7:0], P1 [7:0], P2 [7:0] and P3 [7:0]. Each port bit can be individually accessed by bit addressable instructions. The driving strength of the GPIO ports is programmable (4mA or 8mA, via I2C Configuration EEPROM offset 0x04, see section 3.1.4 for details).

The Table 26 below shows GPIO pin list.

Pin	I/O	Polarity	Ref. Clock	Description
P07 ~ P00	B	-	Operating system clock	Port 0 input/output, bi-directional pins.
P17 ~ P10	B	-	Operating system clock	Port 1 input/output, bi-directional pins.
P27 ~ P20	B	-	Operating system clock	Port 2 input/output, bi-directional pins.
P37 ~ P30	B	-	Operating system clock	Port 3 input/output, bi-directional pins.

Table 26: General Purpose I/O Ports Pins Description

4.13.1 GPIO SFR Register Map

Address	Name	Description
0x80	P0	Port0 Register.
0x90	P1	Port1 Register.
0xA0	P2	Port2 Register.
0xB0	P3	Port3 Register.

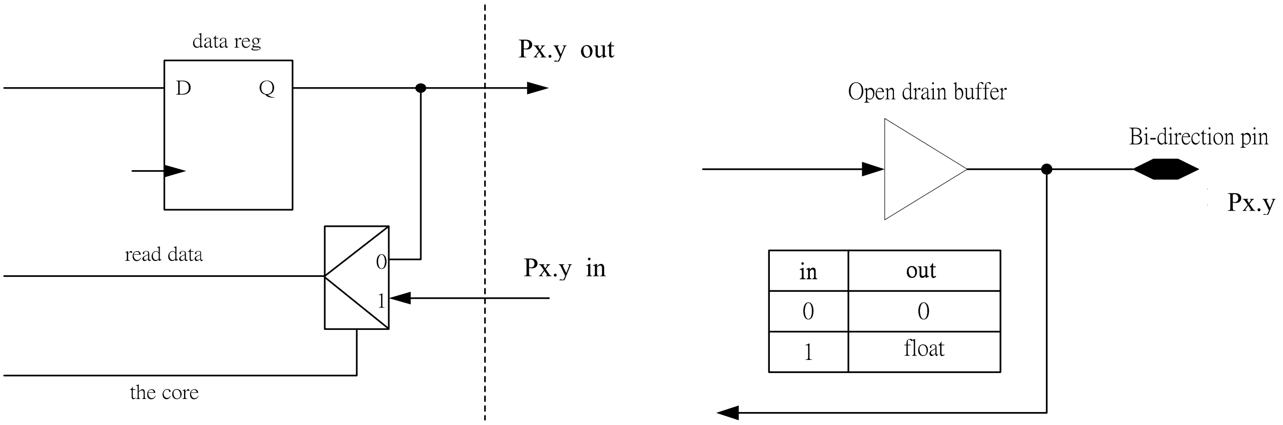
Table 27: GPIO SFR Register Map

Read and write accesses to the I/O ports are performed via their corresponding SFRs, namely, P0 (0x80), P1 (0x90), P2 (0xA0), and P3 (0xB0). Some port-reading instructions read the data register and others read the port's pin. The "Read-Modify-Write" instructions are directed to the data registers and are shown below. All the other instructions used to read a port exclusively read the port's pin.

Instruction	Function description
ANL	Logic AND.
ORL	Logic OR.
XRL	Logic eXclusive OR.
JBC	Jump if bit is set and clear.
CPL	Complement bit.
INC, DEC	Increment, decrement byte.
DJNZ	Decrement and jump if not zero.
MOV Px.y, C	Move carry bit to bit y of port x.
CLR Px.y	Clear bit y of port x.
SETB Px.y	Set bit y of port x.

Table 28: Read-Modify-Write instructions

The port’s pin logic and timing diagrams are shown in figures below.



in	out
0	0
1	float

Figure 73: Ports Pin Logic

Figure 74: Data Register Accessed by Read-Modify-Write Instructions

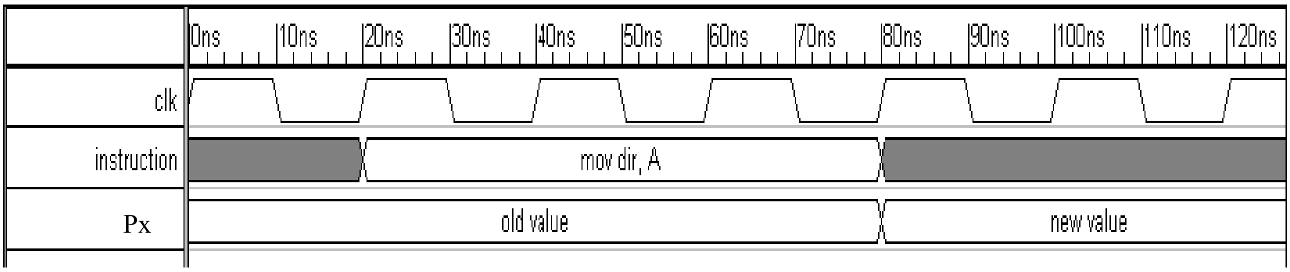


Figure 75: Ports write timing diagram

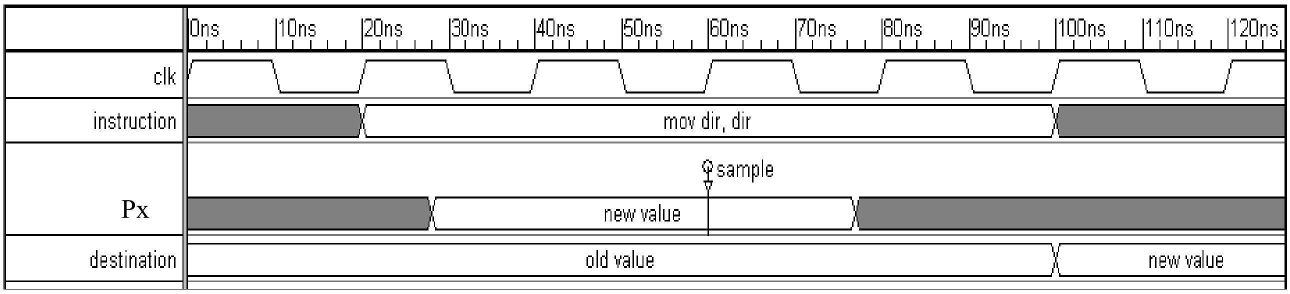


Figure 76: Ports read timing diagram

Port0 Register (P0, 0x80)

Bit	7	6	5	4	3	2	1	0
Name	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
Reset Value	0xFF							

Bit	Name	Access	Description
7:0	P0.[7:0]	R/W	Write 1 then P0.y is tri-state. Write 0 then P0.y is low. If P0.y is tri-state and P0.y_in = 0, then read P0.y is 0. If P0.y is tri-state and P0.y_in = 1, then read P0.y is 1.

Port1 Register (P1, 0x90)

Bit	7	6	5	4	3	2	1	0
Name	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
Reset Value	0xFF							

Bit	Name	Access	Description
7:0	P1.[7:0]	R/W	Write 1 then P1.y is tri-state. Write 0 then P1.y is low. If P1.y is tri-state and P1.y_in = 0, then read P1.y is 0. If P1.y is tri-state and P1.y_in = 1, then read P1.y is 1.

Port2 Register (P2, 0xA0)

Bit	7	6	5	4	3	2	1	0
Name	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
Reset Value	0xFF							

Bit	Name	Access	Description
7:0	P2.[7:0]	R/W	Write 1 then P2.y is tri-state. Write 0 then P2.y is low. If P2.y is tri-state and P2.y_in = 0, then read P2.y is 0. If P2.y is tri-state and P2.y_in = 1, then read P2.y is 1.

Port3 Register (P3, 0xB0)

Bit	7	6	5	4	3	2	1	0
Name	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
Reset Value	0xFF							

Bit	Name	Access	Description
7:0	P3.[7:0]	R/W	Write 1 then P3.y is tri-state. Write 0 then P3.y is low. If P3.y is tri-state and P3.y_in = 0, then read P3.y is 0. If P3.y is tri-state and P3.y_in = 1, then read P3.y is 1.

4.14 TCP/IP Offload Engine

The TCP/IP Offload Engine (TOE) of AX11025 supports some network layer 2 to 4 header processing functions in hardware. The TOE block diagram is shown in below Figure 77. The TOE can operate in two different modes - “Non-Transparent” mode and “Transparent” mode.

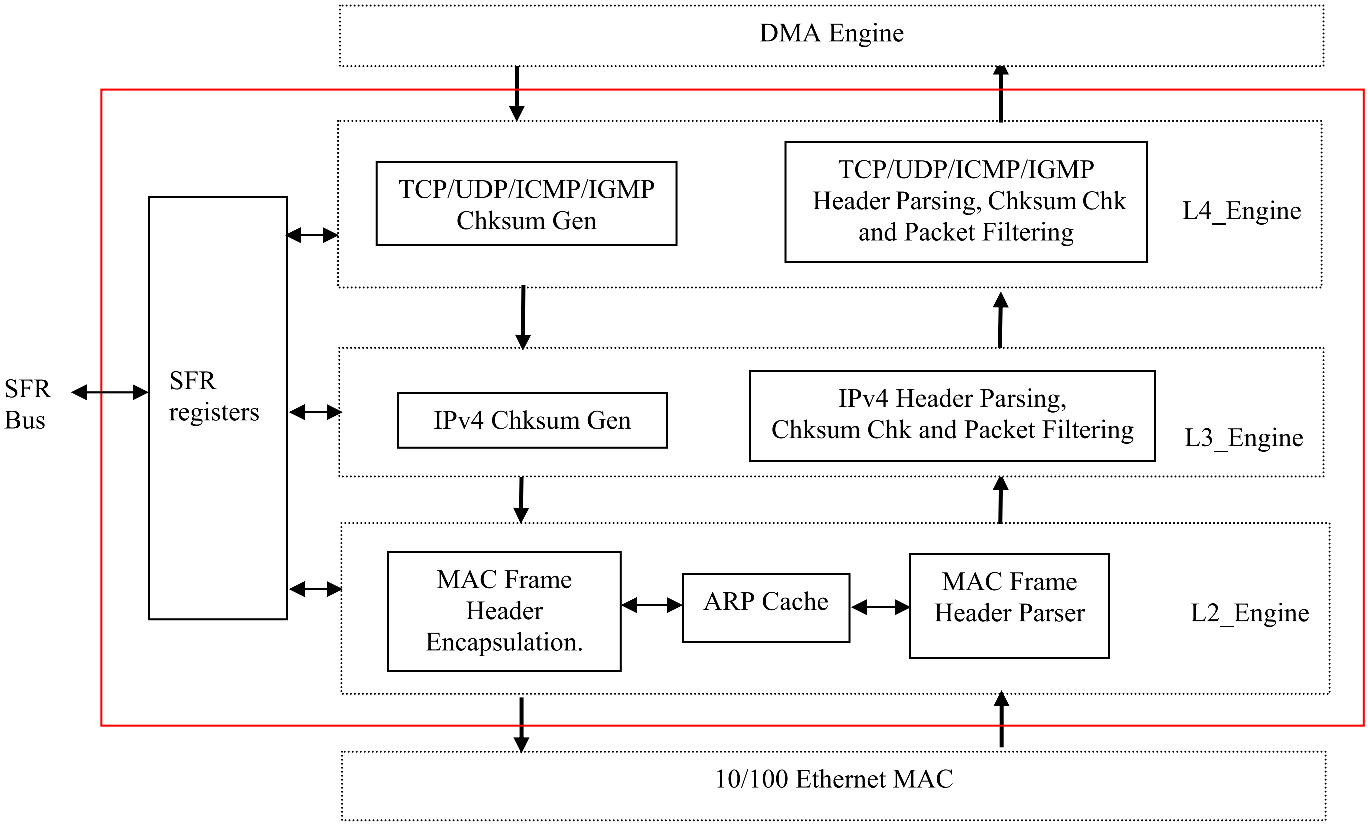

DMA Engine
TCP/UDP/ICMP/IGMP Chksum Gen
TCP/UDP/ICMP/IGMP Header Parsing, Chksum Chk and Packet Filtering
L4_Engine
SFR Bus
SFR registers
IPv4 Chksum Gen
IPv4 Header Parsing, Chksum Chk and Packet Filtering
L3_Engine
MAC Frame Header Encapsulation.
ARP Cache
MAC Frame Header Parser
L2_Engine
10/100 Ethernet MAC

Figure 77: TOE Block Diagram

When TOE operating in “Transparent” mode, the L2_Engine is in transparent mode where hardware processing Ethernet MAC header is disabled. It supports following features,

- VLAN ID filtering for received packets, if enabled
- On-the-fly IPv4 packet header checksum check and generation (with or without PPPoE header, RFC2516)
- Received packet filtering for IPv4 packets with error header checksum
- On-the-fly TCP and UDP segment checksum check and generation
- On-the-fly ICMP and IGMP message checksum check and generation
- Received packet filtering for TCP/UDP/ICMP/IGMP packets with error checksum

When TOE operating in “Non-Transparent” mode, L2_Engine is actively processing Ethernet MAC header. It supports following features,

- Layer-2 functions (the recognizable packet types are Ethernet II encapsulation (RFC894), IEEE 802.2/802.3 SNAP encapsulation (RFC1042), IEEE 802.2/802.3 encapsulation, and NetWare 802.3 RAW encapsulation)
    - Ethernet MAC frame header parsing and encapsulation, including DA, SA, Length/Type, VLAN Tag fields.
    - ARP Cache:
        - When receiving, automatically learns the source IP address and SA of the received Ethernet MAC frames into ARP Cache SRAM
        - When transmitting, automatically sends out ARP-Request packet when the ARP Cache is not found
        - Upon receiving ARP-Request packet, automatically responds with ARP-Reply packet and updates ARP Cache
        - Upon receiving ARP-Reply packet, automatically updates ARP Cache
        - Software programmable timeout value for ARP Cache Timeout
        - ARP Cache SRAM is software accessible
    - VLAN ID filtering for received packets and VLAN Tag insertion for transmit packets, if enabled
    - Received packet filtering for ARP-Request packet
    - Remove layer 2 header of receive IPv4-type packets before forwarding up to Layer-3 function
    - Append layer 2 header of transmit IPv4-type packets from Layer-3 function before passing down to Ethernet MAC
- Layer-3 functions:
    - IPv4 header parsing, including version, header length, total length, protocol, header checksum, source IP address, destination IP address fields
    - On-the-fly IPv4 header checksum check and generation (only when without PPPoE header bytes)
    - Received packet filtering for IPv4 packets with version not equal to 4 or error header checksum
    - Received packet filtering for IPv4 packets with wrong destination IP address (not equal to owned IP address, and not equal to broadcast IP address, and not equal to multicast IP address) and wrong source IP address (equal to broadcast IP address, or equal to multicast IP address)
- Layer 4 functions:
    - On-the-fly TCP and UDP segment checksum check and generation
    - On-the-fly ICMP and IGMP message checksum check and generation
    - Received packet filtering for TCP/UDP/ICMP/IGMP packets with error checksum

There are 5 different types of frame encapsulation that can be received from Ethernet MAC, namely, Ethernet II encapsulation (RFC894), IEEE 802.2/802.3 SNAP encapsulation (RFC1042), IEEE 802.2/802.3 encapsulation, NetWare 802.3 RAW encapsulation, and PPPoE encapsulation (RFC2516). In the first 4 encapsulation types there can be with or without VLAN Tag bytes. Therefore, this makes up total of 9 different encapsulation frame formats that can be received through TOE.

Internal to TOE, it classifies packets into two types, “**IP-type**” packet and “**Non-IP-type**” packet. The “IP-type” packets are those with IPv4 header in it and are most commonly used in TCP/IP protocol stack. They include IP, TCP, UDP, ICMP, and IGMP packets, etc. In its layer 2 header, they may use Ethernet II encapsulation (RFC894), IEEE 802.2/802.3 SNAP encapsulation (RFC1042), or with addition PPPoE header (RFC2516). The “Non-IP-type” packets are those without IPv4 header, such as IPX, IPv6, ARP-Request, ARP-Reply, NETBIOS packets, etc.

When sending or receiving "IP-type" packets, the TOE can process the header information. When sending or receiving "Non-IP-type" packets, the TOE will bypass those packets and have software do the header processing jobs.

Software configures xDATA memory of 1T 80390 CPU with two logical buffer rings, one buffer ring called Receive Packet Buffer Ring (RPBR) and another called Transmit Packet Buffer Ring (TPBR). When TOE receives packets from Ethernet MAC's receive buffer, the L2_Engine shall examine Ethernet header of the packet, the L3/L4_Engine will examine the packet's IPv4 header, validate the IP/TCP/UDP/ICMP/IGMP checksum and then en-queue the packet into RPBR via DMA transfer. Software will then be able to retrieve the packets out of the RPBR for further processing.

On transmit direction, software first en-queues the packet into TPBR, it then instructs TOE to de-queue the packet out of TPBR via DMA transfer and move it to Ethernet MAC's transmit buffer. During this process, the TOE will calculate the IP/TCP/UDP/ICMP/IGMP checksum and append the Ethernet MAC header for the transmit packets.

Following table summarizes how different packets are being processed in TOE.

L2 Engine in	Packet Type	TOE Operation
Transparent Mode	IP-type Packets	TOE retains Ethernet MAC header when receive or transmit and software can receive full packet data of receive packet. Software processes layer-2, layer-3, and layer-4 headers of the packets, but TOE performs IP/TCP/UDP/ICMP/IGMP checksum check and generation for packets with RFC894, RFC1042, and RFC2516 frame format. Packets treated as "IP-type" are: IP, TCP, UDP, ICMP, IGMP packets with RFC894 or RFC1042 frame format (with or without VLAN tag), or with PPPoE header (Eth Type = 8864, RFC2156).
	Non-IP-type Packets	TOE bypasses processing these packets and retains Ethernet MAC header when receive or transmit. Software can receive full packet data of received packet and should process packet headers of transmit and receive packets. Packets treated as "Non-IP-type" are: IPX, IPv6, NETBIOS, ARP packets, or packets with PPPoE header (Eth Type = 8863).
Non-Transparent Mode	IP-type Packets	TOE maintains the ARP function and processes ARP-Request and ARP-Reply packets. TOE strips out Ethernet MAC header when receive and appends Ethernet MAC header when transmit. Software processes layer-3 and layer-4 headers of the packets, but TOE performs IP/TCP/UDP/ICMP/IGMP checksum check and generation for packets with RFC894 and RFC1042 frame format. Packets treated as "IP-type" are: IP, TCP, UDP, ICMP, IGMP packets with RFC894 or RFC1042 frame format (with or without VLAN tag), and ARP-Request and ARP-Reply packets.
	Non-IP-type Packets	TOE bypasses processing these packets and retains Ethernet MAC header when receive or transmit. Software can receive full packet data of received packet and should process packet headers of transmit and receive packets. Packets treated as "Non-IP-type" are: IPX, IPv6, NETBIOS, or packets with PPPoE header (Eth Type = 8863 or 8864).

Table 29: TOE Operation Modes

4.14.1 TOE SFR Register Map

Address	Name	Description
0xAE	TCIR	TOE Command Index Register is used to indicate the address of to-be accessed TOE register.
0xAF	TDR	TOE Data Register is used to read data from or write data to specified TOE register.

Table 30: TOE SFR Register Map

TOE Command Index Register (TCIR, 0xAE)

Bit	7	6	5	4	3	2	1	0
Name	TCIR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	TCIR	WO	Indicate which of the TOE register as listed in Table 31 is to be accessed.

TOE Data Register (TDR, 0xAF)

Bit	7	6	5	4	3	2	1	0
Name	TDR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	TDR	R/W	Data Register is used to write data to or read data from the TOE registers.

TOE Register Indirect Access Method

Software shall use indirect access method through TCIR and TDR registers to do read and write access to the TOE registers as listed in Table 31 below.

Read a register from TOE:

Step 1. Write TCIR: Software indicates the TOE register address to be accessed as the data and write it to the SFR register TCIR.

Step 2. Read TDR: Software then read SFR register TDR. The data read from TDR is the TOE register data indicated in step 1. Keep reading from TDR if the TOE registers have more than one byte, in that case, the first byte being read back is LSB byte.

Write a register to TOE:

Step 1. Write TDR: Software writes the data you want to write into TOE registers to the SFR register TDR. Keep writing to TDR if the TOE registers have more than one byte, in that case, the first byte being written should be LSB byte.

Step 2. Write TCIR: After writing TOE register data to TDR, software then indicates the target TOE register address as data and write it to TCIR.

Note: While software is reading or writing TOE Registers during a sequence of SFR accesses, software can abort that process by writing TCIR with 0xFF.

TOE Register Map

Address	Register Name	Description
Layer 2 Related		
0x00	TL2CR	TOE L2 Control Register
0x01	Reserved	
0x02	TRVTR	TOE RX VLAN Tag Register (16 bits)
0x04	TTVTR	TOE TX VLAN Tag Register (16 bits)
0x06	TACSR	TOE ARP Cache Command Status Register
0x07	TACAR	TOE ARP Cache Address Register
0x08	TACDR	TOE ARP Cache Data Register (48 bits)
0x0E	TACTR	TOE ARP Cache Timeout Register
Layer 3 Related		
0x10	TSIAR	TOE Source IP Address Register (32 bits)
0x14	TSMR	TOE Subnet Mask Register (32 bits)
0x18	TDGIAR	TOE Default Gateway IP Address Register (32 bits)
0x1C	TCSR	TOE Checksum Status Register
Layer 4 Related		
0x20	TL4CR	TOE L4 Control Register
0x21	TL4CMR	TOE L4 Command Register
0x22	TL4BDPR	TOE L4 BDP pointer Register (16 bits)
0x24	TL4DGR	TOE L4 DMA Transfer Gap Register
Interrupt and Status Related		
0x30	TSR	TOE Status Register
0x31	TIER	TOE Interrupt Enable Register

Table 31: TOE Register Map

4.14.2 L2_Engine Function Description

ARP Cache

The ARP Cache SRAM as shown in Figure 78 supports up to 128 entries. Each entry stores the one-to-one mapping information of IP address and its associated MAC address. There is a timer value stored in each entry, which is used for cache timeout purpose. When this timer value reaches a predefined value set by software in TACTR register, it will cause the entry to be flushed out and become invalid (by making the “valid” bit to ‘0’).

When L2_Engine operates in Non-Transparent mode, the ARP Cache Arbiter arbitrates access request to ARP Cache SRAM among software, the Cache timeout timer, during receiving packet, and during sending out packet. Software can write or read to ARP Cache SRAM to create or delete a static entry, which will never time out. The Cache timeout timer is used to flush out the dynamic entries in the Cache SRAM whenever the entries are not being refreshed for a predefined time period. This timeout period is software programmable.

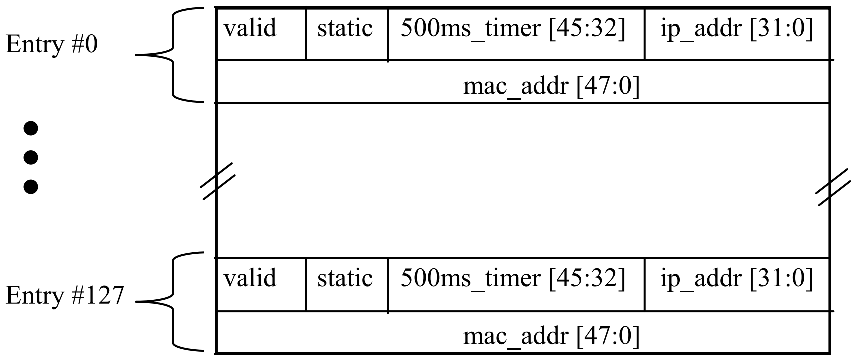
When receiving an “IP-type” packet from Ethernet MAC, the ARP Cache SRAM will be accessed once to refresh the timer for the entry corresponding to the received source IP address of the packet. When sending out an “IP-type” packet, the ARP Cache SRAM will be accessed once to retrieve the destination MAC address of the transmit packet, based on the destination IP address provided by L3 Engine. Note that when transmitting “Non-IP-type” packets, software should provide full MAC header in the packet and should not rely on L2_Engine to search into ARP Cache SRAM. In fact, these types of packet contain no valid destination IP address.

The ARP Cache SRAM supports up to 128 entries. Its address has [7:0] bits, within which the [7:1] bits are the Hash Key for indexing the 128 entries. The Hash Key is first generated based on "Linear Addressing" mode by using the lower 7 bits of the given IP address.

Hash Key [7:1] bit = IP address [6:0]

If the hash collision occurs, then the "XOR Addressing" mode is used next. The XOR Addressing mode is calculated as follows,

Hash Key [7] bit = IP address [31] ^ [23] ^ [15] ^ [7]
Hash Key [6] bit = IP address [30] ^ [22] ^ [14] ^ [6]
Hash Key [5] bit = IP address [29] ^ [21] ^ [13] ^ [5]
Hash Key [4] bit = IP address [28] ^ [20] ^ [12] ^ [4]
Hash Key [3] bit = IP address [27] ^ [19] ^ [11] ^ [3]
Hash Key [2] bit = IP address [26] ^ [18] ^ [10] ^ [2]
Hash Key [1] bit = IP address [25] ^ [17] ^ [9] ^ [1]



Note:

1. The "valid" bit indicates that the entry is valid.
2. The "static" bit means the entry is a static and fixed entry, which will not get expired. When software needs to configure the static ARP cache, set this bit to "1".
3. The "500ms_timer" field is the number of count the 500ms ARP Cache Timeout Timer has ticked so far. This counter number is increased by "1" on every 500ms.
4. The "ip_addr" field is the IP address values.
5. The "mac_addr" field is the MAC address values.

Figure 78: ARP Cache SRAM Memory Map

ARP Request and ARP Reply Packet Processing

When L2_Engine operates in Non-Transparent mode, upon receiving an ARP Request packet from Ethernet MAC with "Target IP address" matching with IP address of this chip set in TSIAR, the L2_Engine will automatically reply with ARP-Reply packet. At the same time, it will use the "Sender Ethernet address" and "Sender IP address" in the received ARP packet to update the entry for this packet in ARP Cache SRAM. When receiving an ARP Reply packet from Ethernet MAC, the L2_Engine will save the "Sender Ethernet address" and "Sender IP address" and then update the entry for this packet in ARP Cache SRAM.

IP Address Translation

Upon receiving "IP-type" packets, ARP-Request and ARP-Reply packets from Ethernet MAC, the L2_Engine will use the packet's source IP address if the IP address is within the same subnet. Otherwise, if it is not within the same sub-net (by comparing with the subnet mask) and the default gateway's IP address != 0.0.0.0 (provided in TDGIAR), then it will use the default gateway's IP address to generate the Hash Key for looking up ARP Cache SRAM. This can conserve the ARP Cache entry usage.

When transmitting "IP-type" packets to Ethernet MAC, the L2_Engine will check the destination IP address of the packet based on following rules to generate correct DA field of Ethernet MAC header. If the look-up fails two times, meaning the entry for the given destination IP address does not exist, the packet will be discarded and this event will be reported to software and an ARP-Request packet will be sent out automatically instead.

Destination IP Address	DA Field Generation Rule
If equal to broadcast destination IP address	Use DA = FFFFFFFFFFFF without looking up to the ARP Cache SRAM
If equal to multicast IP address	Use DA={01005e, {0, Dest_IP[22:0]}} without looking up to the ARP Cache SRAM
If equal to unicast IP address	Use the packet's destination IP address to look up to ARP Cache SRAM if the IP address is within the same subnet. Otherwise, if it is not within the same sub-net (by comparing with the subnet mask) and the default gateway's IP address != 0.0.0.0 (provided in TDGIAR), then use the default gateway's IP address to look up to ARP Cache SRAM.

Table 32: DA Field Generation Rule in Transmit Direction

TOE L2 Control Register (TL2CR, 0x00)

Bit	7	6	5	4	3	2	1	0
Name	TX_SO	TX_SNAP_EN	TX_TRANS	TX_VLAN_EN	RX_SO	Reserved	RX_TRANS	RX_VLAN_EN
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	RXVLAN_EN	R/W	RX VLAN Enable. 1: Setting "1" enables the receiving of VLAN-tagged frame on L2_Engine RX direction, when the TCI byte of received VLAN-tagged frame is matched with TRVTR register and the ETPID is equal 8100 (hex). 0: Setting "0" causes all VLAN-tagged frame to be discarded by L2_Engine.
1	RX_TRANS	R/W	RX Transparent. 1: Setting "1" enables the transparent mode on L2_Engine RX direction, which allows entire MAC frame of "IP-type" packets and ARP packets to be passed up to software. 0: Setting "0" enables L2_Engine function and causes the MAC frame header of "IP-type" packets and ARP-Request packets to be removed by L2_Engine and not being passed up to software.
2	Reserved	R/W	For normal operation, set to "0".
3	RX_SO	R/W	RX Start Operating. 1: Setting "1" enables the operation of RX path of L2/L3/l4 Engine. 0: Setting "0" disables the operation of RX path of L2/L3/l4 Engine.
4	TXVLAN_EN	R/W	TX VLAN Enable. 1: Setting "1" enables the VLAN Tag byte insertion on L2_Engine TX direction, and the content in TTVTR register is used to fill in the TCI bytes of transmitted VLAN-tagged frame.

			0: Setting "0" disables VLAN Tag byte insertion.
5	TX_TRANS	R/W	TX Transparent. 1: Setting "1" enables the transparent mode on L2_Engine TX direction, which means the software is responsible for inserting MAC header for "IP-type" packets and generating ARP packets. 0: Setting "0" enables L2_Engine function, which allows L2_Engine to insert Ethernet MAC header for IP-type packets.
6	TX_SNAP_EN	R/W	1: Setting "1" enables 802.2/802.3 SNAP encapsulation (RFC1042) mode on L2_Engine TX direction. 0: Disables SNAP encapsulation on L2_Engine TX direction.
7	TX_SO	R/W	TX Start Operating. 1: Setting "1" enables the operation of TX path of L2/L3/l4 Engine. 0: Disables the operation of TX path of L2/L3/l4 Engine.

Following is the truth table of L2_Engine RX settings and its behavior.

Bit Settings		**Resultant Packet Receive Conditions**	
RX_TRANS	**RX_VLAN_EN**	**When receiving packets of Ethernet II, SNAP, or Non-IP-type without VLAN Tag**	**When receiving packets of Ethernet II, SNAP, or Non-IP-type with VLAN Tag**
0	0	The "IP-type" packets will be received but the L2 header will be removed before passing up to software. The "non-IP-type" packets will be received and full packet will be passed up to software.	All 3 types of packet will be dropped.
0	1	The "IP-type" packets will be received but the L2 header will be removed before passing up to software. The "non-IP-type" packets will be received and full packet will be passed up to software.	The "IP-type" packets with ETPID = 8100 and TCI byte of received VLAN-tagged frame is matched with TRVTR register or with VID = 0x000 will be received, but the L2 header will be removed before passing up to software. The "non-IP-type" packets with ETPID = 8100 and TCI byte of received VLAN-tagged frame is matched with TRVTR register or with VID = 0x000 will be received and full packet will be passed up to software. Packets with other TCI byte values will be dropped.
1	0	All 3 types of packet will be received and the full packet will be passed up software.	All 3 types of packet will be dropped.
1	1	All 3 types of packet will be received and the full packet will be passed up software.	Packets with ETPID = 8100 and TCI byte of received VLAN-tagged frame is matched with TRVTR register or with VID = 0x000 will be received, and the full packet will be passed to software. Packet with other TCI byte values will be dropped.

Table 33: L2_Engine RX Truth Table

Following is the truth table of L2_Engine TX settings and its behavior.

Bit Settings			Resultant Packet Transmit Conditions
TX_TRANS	**TX_VLAN_EN**	**TX_SNAP_EN**	
0	0	0	Send IP-type packets with Ethernet II without VLAN Tag and without SNAP. Send Non-IP transparently.
0	0	1	Send IP-type packets with Ethernet II with SNAP, but without VLAN Tag. Send Non-IP transparently.
0	1	0	Send IP-type packets with Ethernet II with VLAN Tag (ETPID = 8100 and TCI byte = TTVTR), but without SNAP Send Non-IP transparently.
0	1	1	Send IP-type packets with Ethernet II with VLAN (ETPID = 8100 and TCI byte = TTVTR), and with SNAP Send Non-IP transparently.
1	0	0	Send IP-type packets with Ethernet II transparently. Software is responsible for adding the Ethernet II header for every transmitted packet. Send Non-IP transparently.
1	0	1	Send IP-type packets with Ethernet with SNAP transparently. Software is responsible for adding the Ethernet II and SNAP header for every transmitted packet. Send Non-IP transparently.
1	1	0	Send IP-type packets with Ethernet with VLAN transparently. Software is responsible for adding the Ethernet II and VLAN header for every transmitted packet. Send Non-IP transparently.
1	1	1	Send IP-type packets with Ethernet with VLAN and SNAP transparently. Software is responsible for adding the Ethernet II, VLAN, and SNAP header for every transmitted packet. Send Non-IP transparently.

Table 34: L2_Engine TX Truth Table

TOE RX VLAN Tag Register (TRVR, 0x02)

Bit	7	6	5	4	3	2	1	0
Name	TCI 0							
	Reserved				TCI 1			
Reset Value	Reset value is determined by the I2C EEPROM							

Bit	Name	Access	Description
7:0 11:8	TCI 0 TCI 1	R/W	The TCI 1~0 represents the VLAN ID of Tag Control Information bytes of VLAN-tagged frame. When setting RX_VLAN_EN bit (TL2CR.0) to “1”, the content in this register is used to compare against TCI byte of received VLAN-tagged frame. Only when matched, the received VLAN-tagged frame is passed up to software. Note that a special case of VID = 0x000 in received VLAN-tagged frame will also be passed up to software.

TOE TX VLAN Tag Register (TTVTR, 0x04)

Bit	7	6	5	4	3	2	1	0
Name	TCI 0							
	TCI 1							
Reset Value	Reset value is determined by the I2C EEPROM							

Bit	Name	Access	Description
7:0 15:8	TCI 0 TCI 1	R/W	The TCI 1~0 represents the Tag Control Information bytes of VLAN-tagged frame. When TX_VLAN_EN bit (TL2CR.4) = "1", the content in this register is used to insert the TCI byte of transmitted VLAN-tagged frame.

TOE ARP Cache Command Status Register (TACSR, 0x06)

Bit	7	6	5	4	3	2	1	0
Name	Reserved						GO	READ
Reset Value	00_0000						0	1

Bit	Name	Access	Description
0	READ	R/W	1: Setting READ bit to "1" indicates to read from ARP Cache SRAM. 0: Setting to "0"indicates to write to ARP Cache SRAM.
1	GO	W1/R	1: Setting GO to "1" initiates the ARP Cache SRAM read or write access request to the internal Cache SRAM arbiter. This bit will remain "1" while the access request is still in progress. 0: Arbiter hardware automatically clears this bit after current access request is completed.

TOE ARP Cache Address Register (TACAR, 0x07)

Bit	7	6	5	4	3	2	1	0
Name	SRAM_ADDR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	SRAM_ADDR	R/W	The read or write address of the ARP Cache SRAM.

TOE ARP Cache Data Register (TACDR, 0x08)

Bit	7	6	5	4	3	2	1	0
Name	CACHE_DATA 0							
	CACHE_DATA 1							
	CACHE_DATA 2							
	CACHE_DATA 3							
	CACHE_DATA 4							
	CACHE_DATA 5							
Reset Value	0x0000_0000_0000							

Bit	Name	Access	Description
7:0 … 47:40	CACHE_DATA 0 … CACHE_DATA 5	R/W	The CACHE_DATA 5~0 is the content of the ARP Cache SRAM where CACHE_DATA 5 represents bit 47~40 of the ARP Cache SRAM while CACHE_DATA 0 represents bit 7~0. When writing to the ARP Cache SRAM, software needs to first write desired data into this register before issuing TACSR register. When reading from the ARP Cache SRAM, software first issues TACSR register and then retrieves the SRAM data from this register.

TOE ARP Cache Timeout Register (TACTR, 0x0E)

Bit	7	6	5	4	3	2	1	0
Name	ARP_TIMEOUT							
Reset Value	Reset value is determined by the I2C EEPROM							

Bit	Name	Access	Description
7:0	ARP_TIMEOUT	R/W	Software setting of ARP cache timeout value. Each count is about 8 sec in time. For example, 0x01 = 8 sec. 0x02 = 16 sec. The maximum timeout is 2040 sec which is 34 min.

4.14.3 L3_Engine Function Description

The L3_Engine parses IPv4 header in received packets, recalculates the checksum of IPv4 header and compares it with received checksum bytes. The packets with wrong IP header checksum can be discarded by this block. The block also calculates and inserts the checksum for the transmitted IP header.

When L2_Engine in Non-Transparent mode, the L3 Engine will discard following received IP-type packets:

- Ethernet Type = 0800 but IP version != 4
- IP header checksum error
- Wrong destination IP address (not equal to source IP address of this chip in TSIAR register, and not equal to broadcast IP address, and not equal to multicast IP address). The valid broadcast IP in destination IP address fields are:
    - Limited broadcast: 255.255.255.255
    - Net-directed broadcast for class A: 0 + Net_ID (7 bits) + Host_ID (24 bits), where Host_ID = All ones
    - Net-directed broadcast for class B: 10 + Net_ID (14 bits) + Host_ID (16 bits), where Host_ID = All ones.
    - Net-directed broadcast for class C: 110 + Net_ID (21 bits) + Host_ID (8 bits), where Host_ID = All ones.
    - Subnet-directed broadcast: Net_ID + Subnet_ID + Host_ID, where Subnet_ID is a specific number and Host_ID = All ones.
    - All subnet-directed broadcast: Net_ID + Subnet_ID + Host_ID, where Subnet_ID = All ones, and Host_ID = All ones.
- Wrong source IP address (equal to broadcast IP address, or equal to multicast IP address)

TOE Source IP Address Register (TSIAR, 0x10)

Bit	7	6	5	4	3	2	1	0
Name	IP_ADDR 0							
	IP_ADDR 1							
	IP_ADDR 2							
	IP_ADDR 3							
Reset Value	Reset value is determined by the I2C EEPROM							

Bit	Name	Access	Description
7:0 … 31:24	IP_ADDR 0 … IP_ADDR 3	R/W	The IP_ADDR 3~0 is the IP address of this device where IP_ADDR 3 represents bit 31~24 of IP address while IP_ADDR 0 represents bit 7~0.

TOE Subnet Mask Register (TSMR, 0x14)

Bit	7	6	5	4	3	2	1	0
Name	SUBNET_MASK 0							
	SUBNET_MASK 1							
	SUBNET_MASK 2							
	SUBNET_MASK 3							
Reset Value	Reset value is determined by the I2C EEPROM							

Bit	Name	Access	Description
7:0 … 31:24	SUBNET_MASK 0 … SUBNET_MASK 3	R/W	The SUBNET_MASK 3~0 is the IP subnet mask of this device where SUBNET_MASK 3 represents bit 31~24 of subnet mask while SUBNET_MASK 0 represents bit 7~0.

TOE Default Gateway IP Address Register (TDGIAR, 0x18)

Bit	7	6	5	4	3	2	1	0
Name	GATEWAY_IP 0							
	GATEWAY_IP 1							
	GATEWAY_IP 2							
	GATEWAY_IP 3							
Reset Value	0x0000_0000							

Bit	Name	Access	Description
7:0 … 31:24	GATEWAY_IP 0 … GATEWAY_IP 3	R/W	The GATEWAY_IP 3~0 is the default gateway's IP address of this device. The GATEWAY_IP 3 represents the bit 31~24 of the default gateway's IP address while GATEWAY_IP 0 represents bit 7~0.

TOE Checksum Status Register (TCSR, 0x1C)

Bit	7	6	5	4	3	2	1	0
Name	Reserved			L4CSER	Reserved			L3CSER
Reset Value	000			0	000			0

Bit	Name	Access	Description
0	L3CSER	CR	L3 CheckSum ERror. 1: When reading "1", this bit indicates that there is at least one received packet with its IP header checksum error. 0: No IP header checksum error is found so far.
3:1	Reserved		
4	L4CSER	CR	L4 CheckSum ERror. 1: When reading "1", this bit indicates that there is at least one received packet with its TCP or UDP or ICMP or IGMP packet checksum error. 0: No TCP or UDP or ICMP or IGMP packet checksum error is found so far.
7:5	Reserved		

4.14.4 L4_Engine Function Description

The L4_Engine parses the TCP/ UDP/ICMP/IGMP header of received packets, recalculates checksum of received packet and then compares with received checksum. The packets with wrong checksum can be discarded by this block. This block also calculates and inserts the TCP/UDP/ICMP/IGMP checksum for the transmitted packets.

TOE L4 Control Register (TL4CR, 0x20)

Bit	7	6	5	4	3	2	1	0
Name	Reserved				ETCB	ERCB	EHCI	DPCE
Reset Value	0000				0	0	1	1

Bit	Name	Access	Description
0	DPCE	R/W	Drop Packet with Checksum Error. 1: Setting "1" enables TCP/UDP/ICMP/IGMP packet with checksum error to be dropped by L4 Engine. 0: Setting "0" allows those packets with checksum error to be received into RPBR.
1	EHCI	R/W	Enable Hardware Checksum Insertion. 1: Setting "1" enables hardware to generate and insert the Layer 3 and Layer 4 checksum fields for transmitted TCP/UDP/ICMP/IGMP packet. 0: Disables checksum insertion.
2	ERCB	R/W	Enable Receive packet to Cross RPBR Boundary. 1: Setting "1" enables receive packets to be stored in buffer pages crossing REPP boundary in RPBR, which allows the packet storing in xDATA memory as a ring fashion, i.e., non-contiguous memory range. Please refer to section 4.14.5 for RPBR description. 0: Setting "0" indicates that the receive packets are always stored in xDATA memory as non-ring fashion, i.e., contiguous memory range.
3	ETCB	R/W	Enable Transmit packet to Cross TPBR Boundary. 1: Setting "1" enables transmit packets to be stored in buffer pages crossing TEPP boundary in TPBR, which allows the packet storing in xDATA memory as a ring fashion, i.e., non-contiguous memory range. Please refer to section 4.14.5 for TPBR description. 0: Setting "0" to indicate that the transmit packets are always stored in xDATA memory as non-ring fashion, i.e., contiguous memory range.
7:4	Reserved	R/W	

TOE L4 Command Register (TL4CMR, 0x21)

Bit	7	6	5	4	3	2	1	0
Name	Reserved			SP	Reserved			RPR
Reset Value	0x00							

Bit	Name	Access	Description
0	RPR	W1/R	Resume Packet Receive. 1: Software can set this bit to "1" to resume the packet receive process on the entire RX direction of TOE. This is normally used after the entire RX direction of TOE is halted due to RPBR full condition and the software has de-queued some packets in a previously full RPBR. 0: This bit will be cleared to "0" after the L4 resumes packet receiving process.
3:1	Reserved	R/W	
4	SP	W1/R	Send Packet. 1: Setting this bit "1" tells the L3/L4 Engine to de-queue one packet from the TPBR based on the BDP in TL4BDPR register. This bit will remain "1" until the L4 Engine completes sending out the packet. 0: L4 Engine when done transmitting will clear this bit to "0" automatically.
7:5	Reserved	R/W	

TOE L4 BDP Pointer Register (TL4BDPR, 0x22)

Bit	7	6	5	4	3	2	1	0
Name	BDPP 0							
	BDPP 1							
Reset Value	0x0000							

Bit	Name	Access	Description
7:0 15:8	BDPP 0 BDPP 1	R/W	Software shall configure this register with the BDP pointer of the packet buffer ring so that when a packet is received or transmitted, the L4 Engine knows where the RPBR or TPBR in CPU's xDATA Memory. Please refer to section 4.14.5 for BDP description.

TOE L4 DMA Transfer Gap Register (TL4DGR, 0x24)

Bit	7	6	5	4	3	2	1	0
Name	DMA_GAP							
Reset Value	Reset value is determined by the I2C EEPROM							

Bit	Name	Access	Description
7:0	DMA_GAP	R/W	Software setting of time gap between each 256 bytes of DMA write/read transfer during packet receive or transmit. Each count is 64 system clocks. For example, 0x01 = 64 system clocks. 0x02 = 128 system clocks. The maximum time gap is 16320 system clocks.

TOE Status Register (TSR, 0x30)

Bit	7	6	5	4	3	2	1	0
Name	Reserved	CSP	TPBRE	ACNF	Reserved	RPBRF	RPBRNE	ACHC
Reset Value	0x00							

Bit	Name	Access	Description
0	ACHC	CR	ARP Cache Hashing Collision. 1: When reading "1", this bit indicates that the keys (after both Linear Addressing and XOR Addressing schemes are used as search key) being used to hash the ARP Cache SRAM have returned with result that the entries are pre-occupied by other IP address. 0: No ARP Cache hash collision.
1	RPBRNE	CR	RX Packet Buffer Ring is Not Empty. 1: When reading "1", this bit indicates that there is at least one packet being stored in RPBR. 0: RPBR is empty.
2	RPBRF	CR	RX Packet Buffer Ring is Full. 1: When reading "1", this bit indicates that the RPBR in TL4BDPR register has encountered buffer full condition. Most likely reason is that the L3/L4 Engine is receiving a packet into RPBR and there are no enough free pages to store the received packet. 0: RPBR is not full.
3	Reserved	R	
4	ACNF	CR	ARP Cache Not Found. 1: When reading "1", this bit indicates that the destination MAC address of the packet currently being sent can not be found in ARP Cache SRAM (after both Linear Addressing and XOR Addressing schemes are used as search key) and an ARP-Request packet has been sent out instead. 0: Normal status.
5	TPBRE	CR	TX Packet Buffer Ring is Empty.

			1: When reading "1", this bit indicates that TPBR in TL4BDPR register is empty. Most likely reason is that the software has not stored any packets in TPBR before setting the SP bit (TL4CMR.4) to ask L4 Engine to send out one packet. 0: TPBR is not empty.
6	CSP	CR	L4/L3 Engine has Completed sending out one Packet on TX. 1: When reading "1", after software sets the "SP" bit (TL4CMR.4), this bit indicates that the L4/L3 Engine has completed sending out one packet on TX. 0: TOE TX is still sending packet or in idle state.
7	Reserved	R	

TOE Interrupt Enable Register (TIER, 0x31)

Bit	7	6	5	4	3	2	1	0
Name	Reserved	EI_CSP	EI_TPBRE	EI_ACNF	Reserved	EI_RPBRF	EI_RPBRNE	EI_ACHC
Reset Value	0x00							

Bit	Name	Access	Description
0	EI_ACHC	R/W	Enable Interrupt whenever there is ARP Cache Hashing Collision. 1: Enables generating interrupt to INT4 whenever ACHC flag is set. 0: Disables interrupt.
1	EI_RPBRNE	R/W	Enable Interrupt when RPBR is a Not Empty. 1: Enables generating interrupt to INT4 whenever RPBRNE flag is set. 0: Disables interrupt.
2	EI_RPBRF	R/W	Enable Interrupt when RPBR is Full. 1: Enables generating interrupt to INT4 whenever RPBRF flag is set. 0: Disables interrupt.
3	Reserved	R/W	
4	EI_ACNF	R/W	Enable Interrupt whenever there is ARP Cache Not Found. 1: Enables generating interrupt to INT4 whenever ACNF flag is set. 0: Disables interrupt.
5	EI_TPBRE	R/W	Enable Interrupt for TPBR Empty condition. 1: Enables generating interrupt to INT4 whenever TPBRE flag is set. 0: Disables interrupt.
6	EI_CSP	R/W	Enable Interrupt whenever L4/L3 Engine completes sending out one packet on TX. 1: Enables generating interrupt to INT4 whenever CSP flag is set. 0: Disables interrupt.
7:6	Reserved	R/W	

4.14.5 Packet Buffer Ring in xDATA Memory of 1T 80390 CPU

During software initialization, software is responsible for partitioning the CPU xDATA memory into several logical memory pages (in terms of 256 bytes boundary). The packet buffer ring requires 1 Buffer Descriptor Page and 2 Packet Buffer Rings each having N pages (one Receive Packet Buffer Ring and one Transmit Packet Buffer Ring). This is as shown in Figure 79 below.

The Buffer Descriptor Page is used for storing buffer pointers. The RX/TX Packet Buffer Rings are used for storing actual packet data in a circular buffer fashion. Software shall configure Start Page Pointer and End Page Pointer of both RX and TX Packet Buffer Rings in Buffer Descriptor Page, to indicate the boundary of the two packet buffer rings.

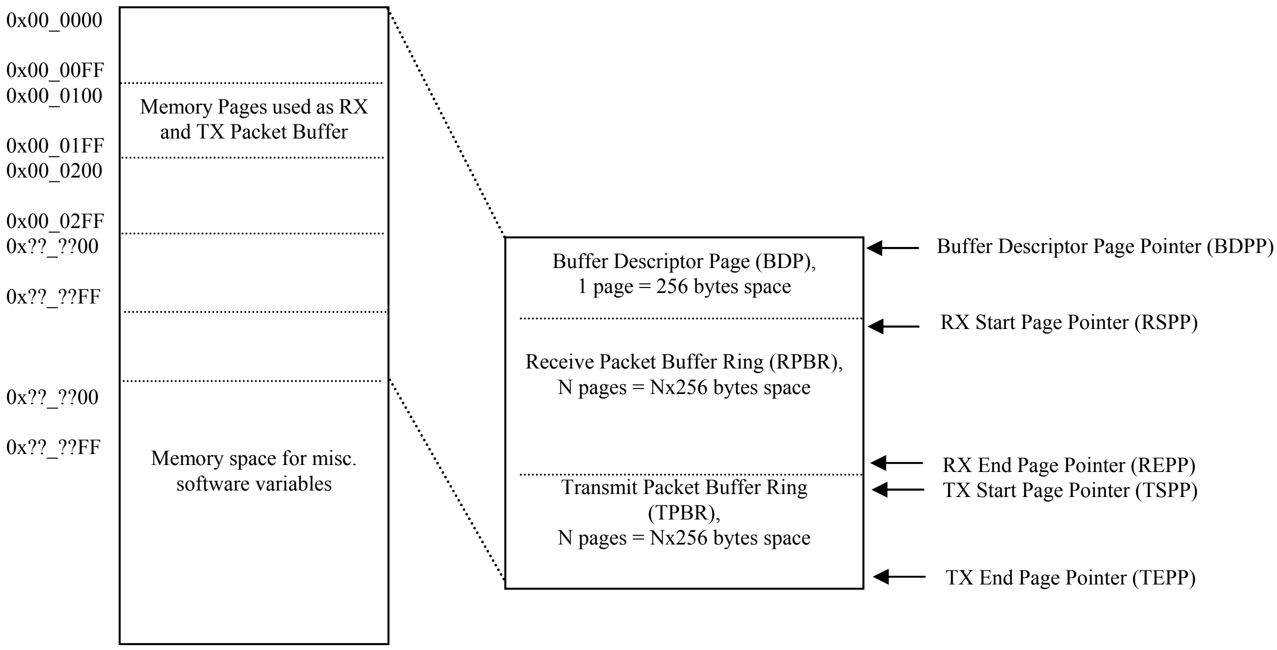
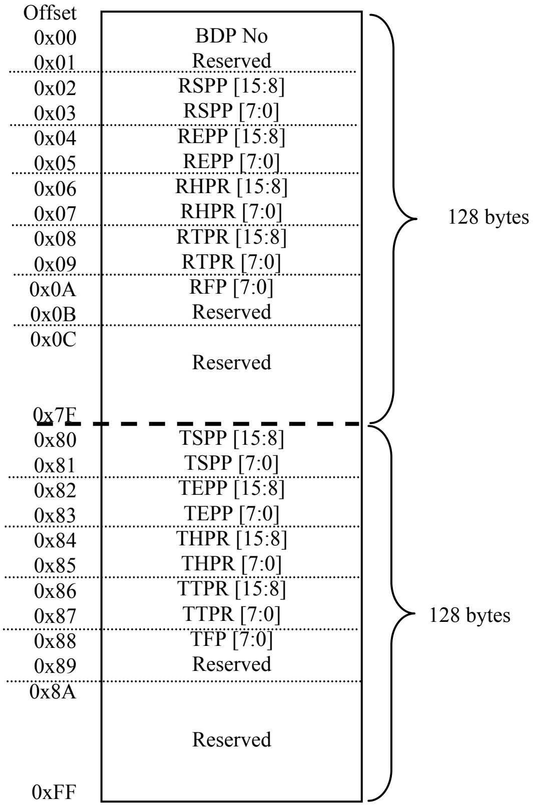

0x00_0000
0x00_00FF
0x00_0100
0x00_01FF
0x00_0200
0x00_02FF
0x??_??00
0x??_??FF
0x??_??00
0x??_??FF
Memory Pages used as RX and TX Packet Buffer
Memory space for misc. software variables
Buffer Descriptor Page (BDP), 1 page = 256 bytes space
Receive Packet Buffer Ring (RPBR), N pages = Nx256 bytes space
Transmit Packet Buffer Ring (TPBR), N pages = Nx256 bytes space
Buffer Descriptor Page Pointer (BDPP)
RX Start Page Pointer (RSPP)
RX End Page Pointer (REPP)
TX Start Page Pointer (TSPP)
TX End Page Pointer (TEPP)

Figure 79: The External Data (xDATA) Memory of CPU

The detailed pointer definition in BDP page is shown in Figure 80. Software should initialize these pointers prior to enabling TOE to active mode. During normal operation, some pointer fields are being updated by software and some are being updated by L4_Engine.

The RPBR and TPBR packet buffer area can operate in a ring fashion or non-ring fashion. This is programmable by software via ERCB bit (TL4CR.2) and ETCB bit (TL4CR.3). Using ring fashion allows xDATA memory use more efficient but may require slightly more complex software driver code. Using non-ring fashion makes software driver code slightly simpler but may waste memory. The two examples of RPBR and TPBR operating in ring fashion are shown in Figure 81 and Figure 82. The detailed pointer and data structure of RPBR and TPBR is described in section 4.14.6.



Offset	
0x00	BDP No
0x01	Reserved
0x02	RSPP [15:8]
0x03	RSPP [7:0]
0x04	REPP [15:8]
0x05	REPP [7:0]
0x06	RHPR [15:8]
0x07	RHPR [7:0]
0x08	RTPR [15:8]
0x09	RTPR [7:0]
0x0A	RFP [7:0]
0x0B	Reserved
0x0C 0x7E	Reserved
0x80	TSPP [15:8]
0x81	TSPP [7:0]
0x82	TEPP [15:8]
0x83	TEPP [7:0]
0x84	THPR [15:8]
0x85	THPR [7:0]
0x86	TTPR [15:8]
0x87	TTPR [7:0]
0x88	TFP [7:0]
0x89	Reserved
0x8A 0xFF	Reserved

Offset	Name	Description
0x00	BDP No	The number of this Buffer Descriptor Page. This number is filled by the software to identify the BDP in the CPU's xDATA memory.
0x02~03	RSPP	RX Start Page Pointer of RX Packet Buffer Ring (RPBR), indicating the beginning page of the RPBR in the xDATA memory.
0x04~05	REPP	RX End Page Pointer of RPBR, indicating the ending page of RPBR in the xDATA memory.
0x06~07	RHPR	RX Head Pointer of RPBR, pointing to the first page of first packet in RPBR. Initial value = RSPP. During packet receive process, software shall always de-queue the packet pointed by RHPR and then update RHPR to point to next packet once done de-queuing one packet.
0x08~09	RTPR	RX Tail Pointer of RPBR, pointing to the next empty page in RPBR. Initial value = RSPP. During packet receive process, the L4_Engine shall update RTPR whenever successfully en-queuing one packet into RPBR. The empty buffer ring condition is indicated by having RTPR = RHPR.
0x0A	RFP	The number of Free Pages remains available in RPBR. Initial value = REPP – RSPP. During packet receive process, the L4_Engine will decrease this value by the page count of the packet currently being en-queued, while software will increase this value by the page count of the packet currently being de-queued.
0x80~81	TSPP	TX Start Page Pointer of TX Packet Buffer Ring (TPBR), indicating the beginning page of the TPBR in the xDATA memory.
0x82~83	TEPP	TX End Page Pointer of TPBR, indicating the ending page of TPBR in the xDATA memory.
0x84~85	THPR	TX Head Pointer of TPBR, pointing to the first page of first packet in TPBR. Initial value = TSPP. During packet transmit process, the L4_Engine shall de-queue the packet pointed by THPR in TPBR and then update THPR to point to next packet once done de-queuing one packet.
0x86~87	TTPR	TX Tail Pointer of TPBR, pointing to the next empty page in TPBR. Initial value = TSPP. During packet transmit process, software shall update TTPR after en-queuing one packet into TPBR. The empty buffer ring condition is indicated by having TTPR = THPR.
0x88	TFP	The number of Free Pages remains available in TPBR. Initial value = TEPP – TSPP. During packet transmit process, software will decrease this value by page count of the packet currently being en-queued while L4_Engine will increase this value by the page count of the packet currently being de-queued.

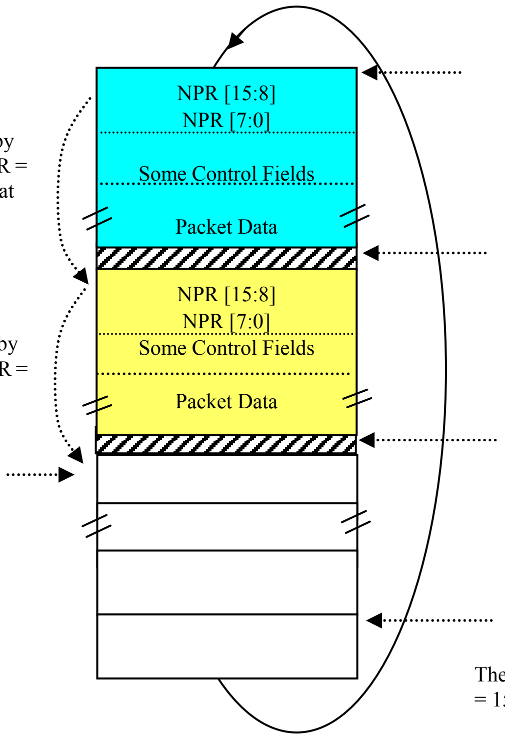
Figure 80: The Content of Buffer Descriptor Page (BDP)

Example L4_Engine Behavior

Assume packet #1 with 6 pages long is en-queued by the L4_Engine, the L4_Engine puts packet #1's NPR = 0x0027 and RTPR = 0x0027. This packet is stored at page pointed by initial RTPR = 0x0021.

Assume packet #2 with 3 pages long is en-queued by the L4_Engine, the L4_Engine puts packet #2's NPR = 0x002A, and RTPR = 0x002A.

After the L4_Engine en-queues 2 received packets into RPBR, we shall see RTPR = 0x002A.



Example Software Behavior

Initially, software sets RSPP = 0x0021, RHPR = 0x0021, and RTPR = 0x0021.

When RTPR is not equal to RHPR (meaning some packets are in the RPBR), software can start de-queuing the packet pointed by RHPR. Once done, software updates RHPR = NPR of packet #1.

After software de-queues packet #2, it updates RHPR = NPR of packet #2. Since RTPR now is equal to RHPR, this means that packet #2 is the last packet in the buffer ring, and the RPBR will become empty after de-queuing packet #2.

Initially, software set REPP = 0x002F

The size of RX Packet Buffer Ring = REPP- RSPP = 15 pages or 15x256 = 3840 bytes.

Figure 81: Example Ring Structure of Receive Packet Buffer Ring

Example SW Behavior

Initially, assume SW sets TSPP = 0x0030, THPR = 0x0030, and TTPR = 0x0030.

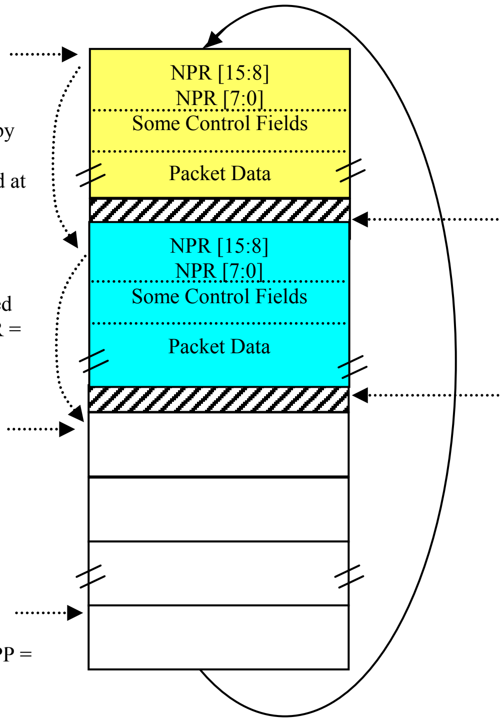
Assume packet #1 with 6 pages long is en-queued by software, so software shall put packet #1's NPR = 0x0036, and TTPR = 0x0036. (This packet is stored at the page pointed by initial TTPR = 0x0030)

Assuming packet #2 with 3 pages long is en-queued by software, so software shall put packet #2's NPR = 0x0039, and TTPR = 0x0039.

After software en-queues 2 transmitted packets into TPBR, we shall see TTPR = 0x0039.

Software sets TEPP = 0x003F

The size of TX Packet Buffer Ring = TEPP- TSPP = 16 pages or 16x256 = 4096 bytes.



Example L4_Engine Behavior

After software en-queues one packet, it can tell the L4_Engine to start de-queuing the packet pointed by THPR. After sending packet #1, the L4_Engine will update THPR = NPR of packet #1.

After software en-queues packet #2, it can tell the L4_Engine to start de-queuing the packet pointed by NPR of packet #1. After sending packet #2, the L4_Engine will update the THPR to point to the next packet.

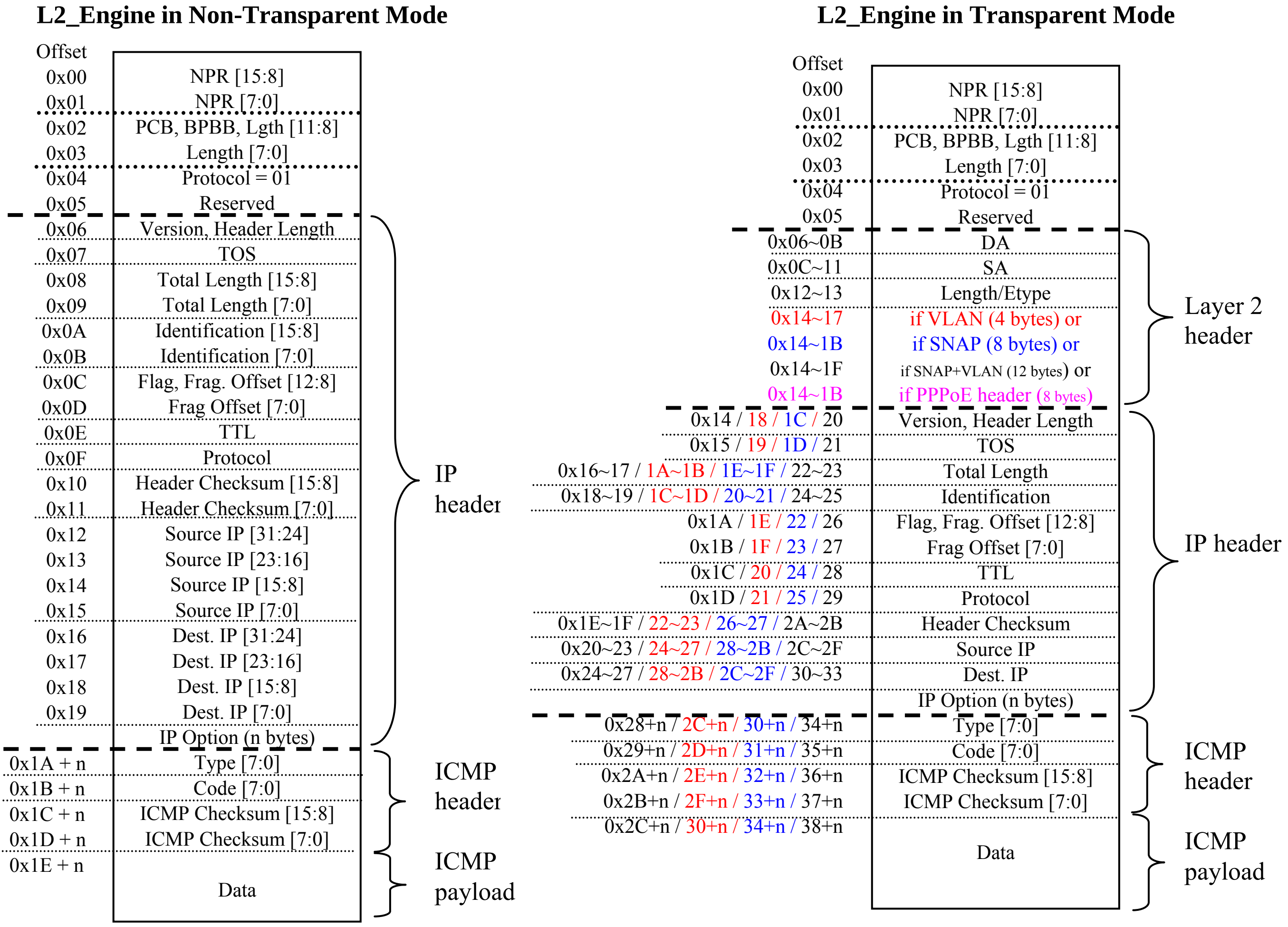
Figure 82: Example Ring Structure of Transmit Packet Buffer Ring

4.14.6 Packet Format in Packet Buffer Ring

Receive Packet Buffer Ring

In TOE RX direction, the L4_Engine is responsible for en-queuing the received packets into RPBR while software is responsible for de-queuing the received packets out of the RPBR. The packets in the ring are linked together via Next Pointer (NPR) field in each packet as shown below. The DMA transfer mechanism is used to move received packets from Ethernet MAC receive buffer through TOE to RPBR. Below Figure 83 to Figure 87 show the different packet format in RPBR.

1. ICMP Packet Format in RPBR



L2_Engine in Non-Transparent Mode

Offset	
0x00	NPR [15:8]
0x01	NPR [7:0]
0x02	PCB, BPBB, Lgth [11:8]
0x03	Length [7:0]
0x04	Protocol = 01
0x05	Reserved
0x06	Version, Header Length
0x07	TOS
0x08	Total Length [15:8]
0x09	Total Length [7:0]
0x0A	Identification [15:8]
0x0B	Identification [7:0]
0x0C	Flag, Frag. Offset [12:8]
0x0D	Frag Offset [7:0]
0x0E	TTL
0x0F	Protocol
0x10	Header Checksum [15:8]
0x11	Header Checksum [7:0]
0x12	Source IP [31:24]
0x13	Source IP [23:16]
0x14	Source IP [15:8]
0x15	Source IP [7:0]
0x16	Dest. IP [31:24]
0x17	Dest. IP [23:16]
0x18	Dest. IP [15:8]
0x19	Dest. IP [7:0]
	IP Option (n bytes)
0x1A + n	Type [7:0]
0x1B + n	Code [7:0]
0x1C + n	ICMP Checksum [15:8]
0x1D + n	ICMP Checksum [7:0]
0x1E + n	Data

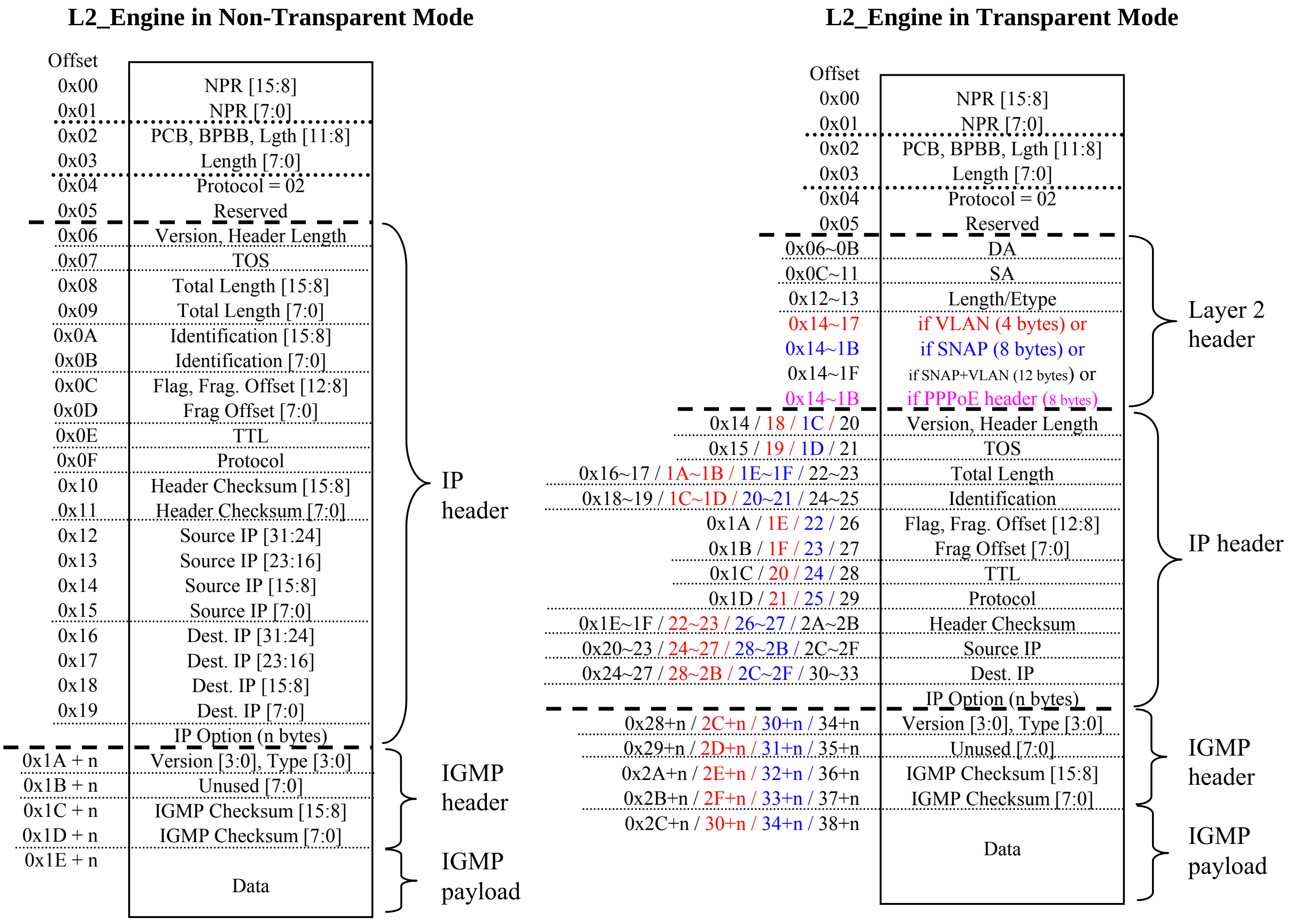
L2_Engine in Transparent Mode

Offset	
0x00	NPR [15:8]
0x01	NPR [7:0]
0x02	PCB, BPBB, Lgth [11:8]
0x03	Length [7:0]
0x04	Protocol = 01
0x05	Reserved
0x06~0B	DA
0x0C~11	SA
0x12~13	Length/Etype
0x14~17	if VLAN (4 bytes) or
0x14~1B	if SNAP (8 bytes) or
0x14~1F	if SNAP+VLAN (12 bytes) or
0x14~1B	if PPPoE header (8 bytes)
0x14 / 18 / 1C / 20	Version, Header Length
0x15 / 19 / 1D / 21	TOS
0x16~17 / 1A~1B / 1E~1F / 22~23	Total Length
0x18~19 / 1C~1D / 20~21 / 24~25	Identification
0x1A / 1E / 22 / 26	Flag, Frag. Offset [12:8]
0x1B / 1F / 23 / 27	Frag Offset [7:0]
0x1C / 20 / 24 / 28	TTL
0x1D / 21 / 25 / 29	Protocol
0x1E~1F / 22~23 / 26~27 / 2A~2B	Header Checksum
0x20~23 / 24~27 / 28~2B / 2C~2F	Source IP
0x24~27 / 28~2B / 2C~2F / 30~33	Dest. IP
	IP Option (n bytes)
0x28+n / 2C+n / 30+n / 34+n	Type [7:0]
0x29+n / 2D+n / 31+n / 35+n	Code [7:0]
0x2A+n / 2E+n / 32+n / 36+n	ICMP Checksum [15:8]
0x2B+n / 2F+n / 33+n / 37+n	ICMP Checksum [7:0]
0x2C+n / 30+n / 34+n / 38+n	Data

Offset	Bit	Field Name	Description
0x00	7:0	NPR [15:8]	The Next Pointers of RPBR: The NPR field indicates the first page of the next packet in the RPBR.
0x01	7:0	NPR [7:0]	
0x02	7	PCB	When TL4CR[ERCB] = 1, the PCB flag indicates that this Packet Crosses the packet buffer ring Boundary. In that case, when PCB flag is “1”, BPBB[2:0] field indicates the # of Buffer Pages being used for this packet Before REPP Boundary. When PCB flag is “0”, BPBB field is undefined. When TL4CR[ERCB] = 0, PCB and BPBB are undefined.
0x02	6:4	BPBB	
0x02	3:0	Length [11:8]	The Length field indicates the total length in bytes from (Version, Header Length) field to Data field in Non-Transparent mode, or from DA field to Data field in Transparent mode.
0x03	7:0	Length [7:0]	
0x04	7:0	Protocol	The Protocol = 0x01 indicating that the packet is an ICMP packet

Figure 83: ICMP Packet Format in RPBR

2. IGMP Packet Format in RPBR



L2_Engine in Non-Transparent Mode

Offset	
0x00	NPR [15:8]
0x01	NPR [7:0]
0x02	PCB, BPBB, Lgth [11:8]
0x03	Length [7:0]
0x04	Protocol = 02
0x05	Reserved
0x06	Version, Header Length
0x07	TOS
0x08	Total Length [15:8]
0x09	Total Length [7:0]
0x0A	Identification [15:8]
0x0B	Identification [7:0]
0x0C	Flag, Frag. Offset [12:8]
0x0D	Frag Offset [7:0]
0x0E	TTL
0x0F	Protocol
0x10	Header Checksum [15:8]
0x11	Header Checksum [7:0]
0x12	Source IP [31:24]
0x13	Source IP [23:16]
0x14	Source IP [15:8]
0x15	Source IP [7:0]
0x16	Dest. IP [31:24]
0x17	Dest. IP [23:16]
0x18	Dest. IP [15:8]
0x19	Dest. IP [7:0]
	IP Option (n bytes)
0x1A + n	Version [3:0], Type [3:0]
0x1B + n	Unused [7:0]
0x1C + n	IGMP Checksum [15:8]
0x1D + n	IGMP Checksum [7:0]
0x1E + n	Data

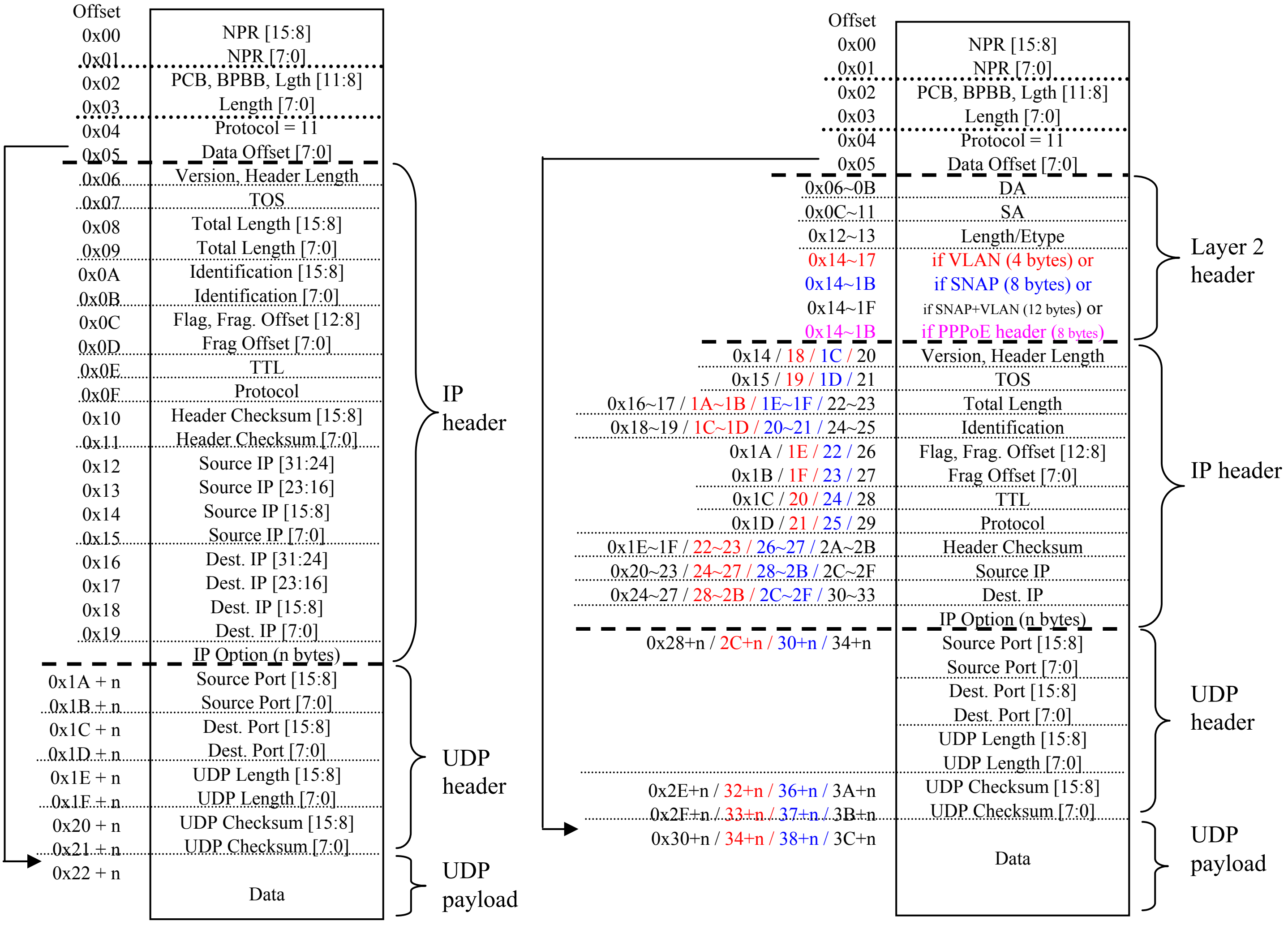
L2_Engine in Transparent Mode

Offset	
0x00	NPR [15:8]
0x01	NPR [7:0]
0x02	PCB, BPBB, Lgth [11:8]
0x03	Length [7:0]
0x04	Protocol = 02
0x05	Reserved
0x06~0B	DA
0x0C~11	SA
0x12~13	Length/Etype
0x14~17	if VLAN (4 bytes) or
0x14~1B	if SNAP (8 bytes) or
0x14~1F	if SNAP+VLAN (12 bytes) or
0x14~1B	if PPPoE header (8 bytes)
0x14 / 18 / 1C / 20	Version, Header Length
0x15 / 19 / 1D / 21	TOS
0x16~17 / 1A~1B / 1E~1F / 22~23	Total Length
0x18~19 / 1C~1D / 20~21 / 24~25	Identification
0x1A / 1E / 22 / 26	Flag, Frag. Offset [12:8]
0x1B / 1F / 23 / 27	Frag Offset [7:0]
0x1C / 20 / 24 / 28	TTL
0x1D / 21 / 25 / 29	Protocol
0x1E~1F / 22~23 / 26~27 / 2A~2B	Header Checksum
0x20~23 / 24~27 / 28~2B / 2C~2F	Source IP
0x24~27 / 28~2B / 2C~2F / 30~33	Dest. IP
	IP Option (n bytes)
0x28+n / 2C+n / 30+n / 34+n	Version [3:0], Type [3:0]
0x29+n / 2D+n / 31+n / 35+n	Unused [7:0]
0x2A+n / 2E+n / 32+n / 36+n	IGMP Checksum [15:8]
0x2B+n / 2F+n / 33+n / 37+n	IGMP Checksum [7:0]
0x2C+n / 30+n / 34+n / 38+n	Data

Offset	Bit	Field Name	Description
0x00	7:0	NPR [15:8]	Same as description for ICMP packet.
0x01	7:0	NPR [7:0]	
0x02	7	PCB	Same as description for ICMP packet.
0x02	6:4	BPBB	
0x02	3:0	Length [11:8]	Same as description for ICMP packet.
0x03	7:0	Length [7:0]	
0x04	7:0	Protocol	The Protocol = 0x02 indicating that the packet is an IGMP packet

Figure 84: IGMP Packet Format in RPBR

3. UDP Packet Format in RPBR



L2-Engine in Non-Transparent Mode

Offset	
0x00	NPR [15:8]
0x01	NPR [7:0]
0x02	PCB, BPBB, Lgth [11:8]
0x03	Length [7:0]
0x04	Protocol = 11
0x05	Data Offset [7:0]
0x06	Version, Header Length
0x07	TOS
0x08	Total Length [15:8]
0x09	Total Length [7:0]
0x0A	Identification [15:8]
0x0B	Identification [7:0]
0x0C	Flag, Frag. Offset [12:8]
0x0D	Frag Offset [7:0]
0x0E	TTL
0x0F	Protocol
0x10	Header Checksum [15:8]
0x11	Header Checksum [7:0]
0x12	Source IP [31:24]
0x13	Source IP [23:16]
0x14	Source IP [15:8]
0x15	Source IP [7:0]
0x16	Dest. IP [31:24]
0x17	Dest. IP [23:16]
0x18	Dest. IP [15:8]
0x19	Dest. IP [7:0]
	IP Option (n bytes)
0x1A + n	Source Port [15:8]
0x1B + n	Source Port [7:0]
0x1C + n	Dest. Port [15:8]
0x1D + n	Dest. Port [7:0]
0x1E + n	UDP Length [15:8]
0x1F + n	UDP Length [7:0]
0x20 + n	UDP Checksum [15:8]
0x21 + n	UDP Checksum [7:0]
0x22 + n	Data

L2-Engine in Transparent Mode

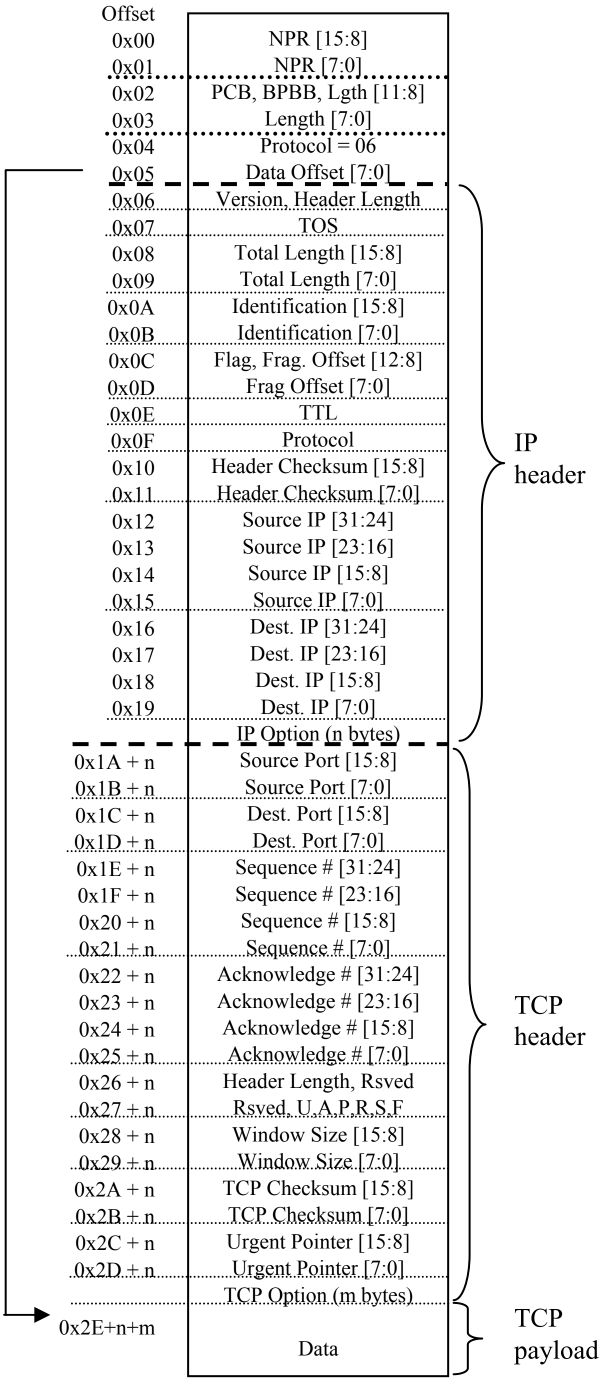
Offset	
0x00	NPR [15:8]
0x01	NPR [7:0]
0x02	PCB, BPBB, Lgth [11:8]
0x03	Length [7:0]
0x04	Protocol = 11
0x05	Data Offset [7:0]
0x06~0B	DA
0x0C~11	SA
0x12~13	Length/Etype
0x14~17	if VLAN (4 bytes) or
0x14~1B	if SNAP (8 bytes) or
0x14~1F	if SNAP+VLAN (12 bytes) or
0x14~1B	if PPPoE header (8 bytes)
0x14 / 18 / 1C / 20	Version, Header Length
0x15 / 19 / 1D / 21	TOS
0x16~17 / 1A~1B / 1E~1F / 22~23	Total Length
0x18~19 / 1C~1D / 20~21 / 24~25	Identification
0x1A / 1E / 22 / 26	Flag, Frag. Offset [12:8]
0x1B / 1F / 23 / 27	Frag Offset [7:0]
0x1C / 20 / 24 / 28	TTL
0x1D / 21 / 25 / 29	Protocol
0x1E~1F / 22~23 / 26~27 / 2A~2B	Header Checksum
0x20~23 / 24~27 / 28~2B / 2C~2F	Source IP
0x24~27 / 28~2B / 2C~2F / 30~33	Dest. IP
	IP Option (n bytes)
0x28+n / 2C+n / 30+n / 34+n	Source Port [15:8]
	Source Port [7:0]
	Dest. Port [15:8]
	Dest. Port [7:0]
	UDP Length [15:8]
	UDP Length [7:0]
0x2E+n / 32+n / 36+n / 3A+n	UDP Checksum [15:8]
0x2F+n / 33+n / 37+n / 3B+n	UDP Checksum [7:0]
0x30+n / 34+n / 38+n / 3C+n	Data

Offset	Bit	Field Name	Description
0x00	7:0	NPR [15:8]	Same as description for ICMP packet.
0x01	7:0	NPR [7:0]	
0x02	7	PCB	Same as description for ICMP packet.
0x02	6:4	BPBB	
0x02	3:0	Length [11:8]	Same as description for ICMP packet.
0x03	7:0	Length [7:0]	
0x04	7:0	Protocol	The Protocol = 0x11 indicating that the packet is an UDP packet
0x05	7:0	Data Offset	The Data Offset field indicates the address offset of where the first UDP payload byte is located. For example, if without IP Option field, this field normally is 28 (dec) for Non-Transparent or 42 (dec) for Transparent without VLAN/SNAP/PPPoE. Therefore, the real memory address of first UDP payload byte = the real memory address of Data Offset field + the value of Data Offset + 1.

Figure 85: UDP Packet Format in RPBR

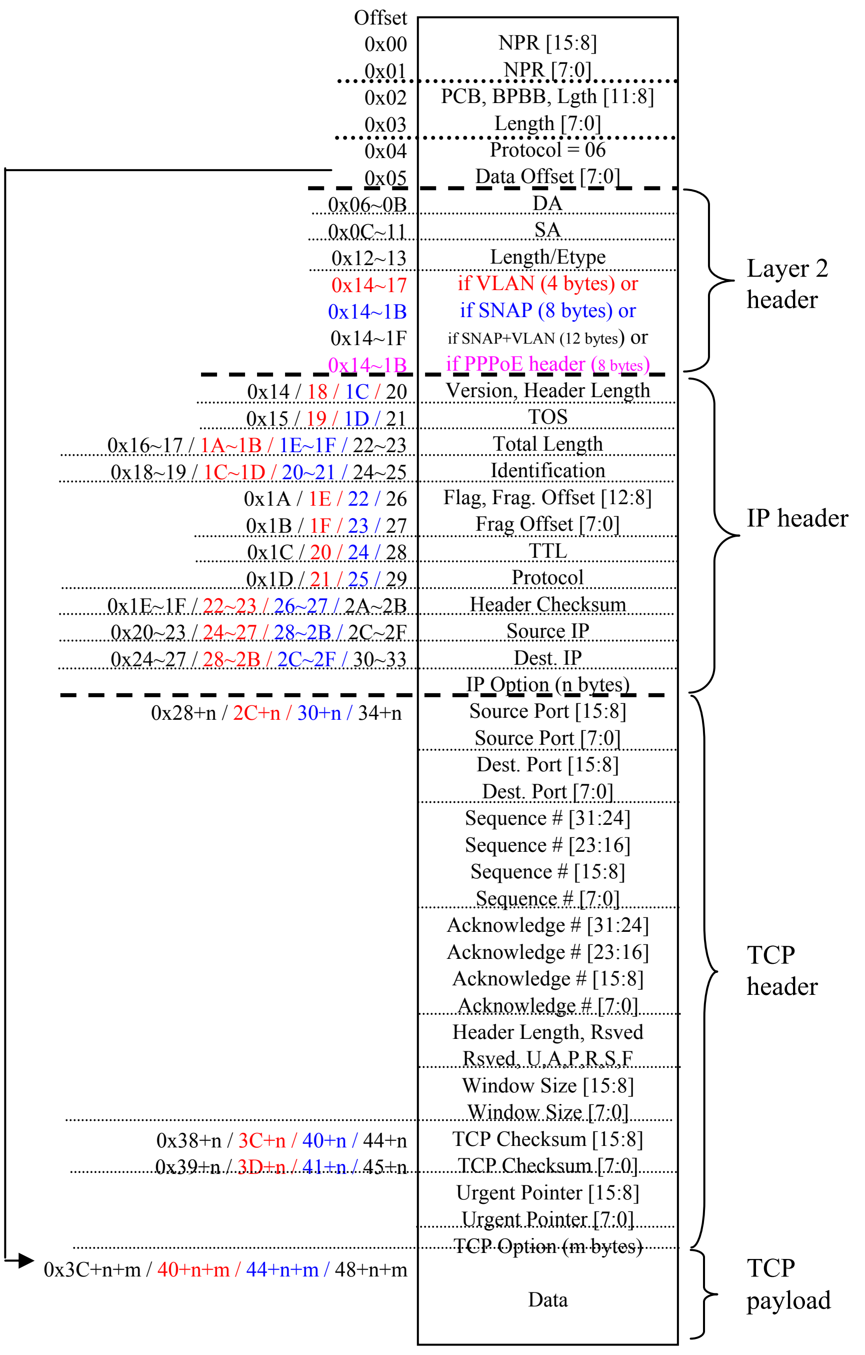
4. TCP Packet Format in RPBR

L2-Engine in Non-Transparent Mode



Offset	
0x00	NPR [15:8]
0x01	NPR [7:0]
0x02	PCB, BPBB, Lgth [11:8]
0x03	Length [7:0]
0x04	Protocol = 06
0x05	Data Offset [7:0]
0x06	Version, Header Length
0x07	TOS
0x08	Total Length [15:8]
0x09	Total Length [7:0]
0x0A	Identification [15:8]
0x0B	Identification [7:0]
0x0C	Flag, Frag. Offset [12:8]
0x0D	Frag Offset [7:0]
0x0E	TTL
0x0F	Protocol
0x10	Header Checksum [15:8]
0x11	Header Checksum [7:0]
0x12	Source IP [31:24]
0x13	Source IP [23:16]
0x14	Source IP [15:8]
0x15	Source IP [7:0]
0x16	Dest. IP [31:24]
0x17	Dest. IP [23:16]
0x18	Dest. IP [15:8]
0x19	Dest. IP [7:0]
	IP Option (n bytes)
0x1A + n	Source Port [15:8]
0x1B + n	Source Port [7:0]
0x1C + n	Dest. Port [15:8]
0x1D + n	Dest. Port [7:0]
0x1E + n	Sequence # [31:24]
0x1F + n	Sequence # [23:16]
0x20 + n	Sequence # [15:8]
0x21 + n	Sequence # [7:0]
0x22 + n	Acknowledge # [31:24]
0x23 + n	Acknowledge # [23:16]
0x24 + n	Acknowledge # [15:8]
0x25 + n	Acknowledge # [7:0]
0x26 + n	Header Length, Rsved
0x27 + n	Rsved, U,A,P,R,S,F
0x28 + n	Window Size [15:8]
0x29 + n	Window Size [7:0]
0x2A + n	TCP Checksum [15:8]
0x2B + n	TCP Checksum [7:0]
0x2C + n	Urgent Pointer [15:8]
0x2D + n	Urgent Pointer [7:0]
	TCP Option (m bytes)
0x2E+n+m	Data

L2-Engine in Transparent Mode



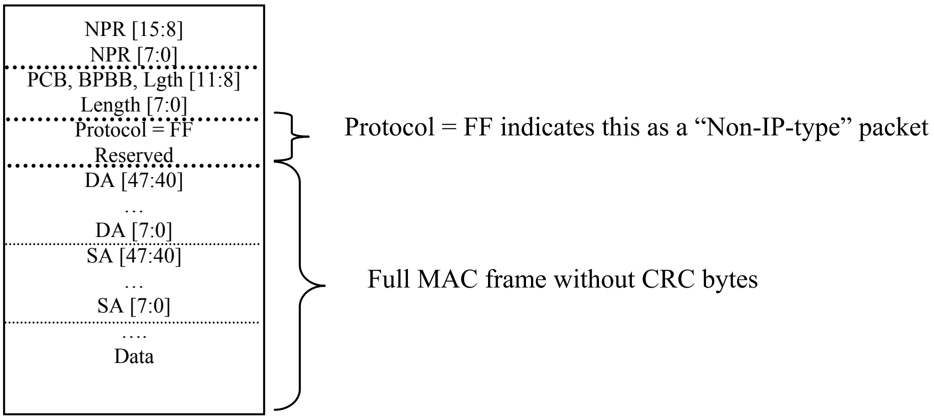
Offset	
0x00	NPR [15:8]
0x01	NPR [7:0]
0x02	PCB, BPBB, Lgth [11:8]
0x03	Length [7:0]
0x04	Protocol = 06
0x05	Data Offset [7:0]
0x06~0B	DA
0x0C~11	SA
0x12~13	Length/Etype
0x14~17	if VLAN (4 bytes) or
0x14~1B	if SNAP (8 bytes) or
0x14~1F	if SNAP+VLAN (12 bytes) or
0x14~1B	if PPPoE header (8 bytes)
0x14 / 18 / 1C / 20	Version, Header Length
0x15 / 19 / 1D / 21	TOS
0x16~17 / 1A~1B / 1E~1F / 22~23	Total Length
0x18~19 / 1C~1D / 20~21 / 24~25	Identification
0x1A / 1E / 22 / 26	Flag, Frag. Offset [12:8]
0x1B / 1F / 23 / 27	Frag Offset [7:0]
0x1C / 20 / 24 / 28	TTL
0x1D / 21 / 25 / 29	Protocol
0x1E~1F / 22~23 / 26~27 / 2A~2B	Header Checksum
0x20~23 / 24~27 / 28~2B / 2C~2F	Source IP
0x24~27 / 28~2B / 2C~2F / 30~33	Dest. IP
	IP Option (n bytes)
0x28+n / 2C+n / 30+n / 34+n	Source Port [15:8]
	Source Port [7:0]
	Dest. Port [15:8]
	Dest. Port [7:0]
	Sequence # [31:24]
	Sequence # [23:16]
	Sequence # [15:8]
	Sequence # [7:0]
	Acknowledge # [31:24]
	Acknowledge # [23:16]
	Acknowledge # [15:8]
	Acknowledge # [7:0]
	Header Length, Rsved
	Rsved, U,A,P,R,S,F
	Window Size [15:8]
	Window Size [7:0]
0x38+n / 3C+n / 40+n / 44+n	TCP Checksum [15:8]
0x39+n / 3D+n / 41+n / 45+n	TCP Checksum [7:0]
	Urgent Pointer [15:8]
	Urgent Pointer [7:0]
	TCP Option (m bytes)
0x3C+n+m / 40+n+m / 44+n+m / 48+n+m	Data

Offset	Bit	Field Name	Description
0x00	7:0	NPR [15:8]	Same as description for ICMP packet.
0x01	7:0	NPR [7:0]	
0x02	7	PCB	Same as description for ICMP packet.
0x02	6:4	BPBB	
0x02	3:0	Length [11:8]	Same as description for ICMP packet.
0x03	7:0	Length [7:0]	

0x04	7:0	Protocol	The Protocol = 0x06 indicating that the packet is an TCP packet
0x05	7:0	Data Offset	The Data Offset field indicates the address offset of where the first TCP payload byte is located. For example, if without IP Option field and TCP Option field, this field normally is 40 (dec) for Non-transparent or 54 (dec) for Transparent without VLAN/SNAP/PPPoE. Therefore, the real memory address of first TCP payload byte = the real memory address of Data Offset field + the value of Data Offset + 1.

Figure 86: TCP Packet Format in RPBR

5. Non-IP-type Packet Format in RPBR



Offset	Bit	Field Name	Description
0x00	7:0	NPR [15:8]	Same as description for ICMP packet.
0x01	7:0	NPR [7:0]	
0x02	7	PCB	Same as description for ICMP packet.
0x02	6:4	BPBB	
0x02	3:0	Length [11:8]	The Length field indicates the total length in bytes from DA field to Data field regardless it's either in Non-transparent or Transparent mode.
0x03	7:0	Length [7:0]	
0x04	7:0	Protocol	When in L2 Engine Non-Transparent mode, the following packet encapsulation will be treated as "Non-IP-type" packet by TOE RX (i.e., the Protocol field will be put with 0xFF), ● IEEE 802.2/802.3 Encapsulation (BPDU/GMRP/GVRP, NETBIOS, IPX) ● NetWare 802.3 RAW Encapsulation (IPX) ● IPv6 Packet (Etype = 0x86DD) ● PPPoE frame (if Etype = 0x8863 or if Etype = 0x8864) When in L2 Engine Transparent mode, the following packet encapsulation will be treated as "Non-IP-type" packet by TOE RX (i.e., the Protocol field will be put with 0xFF), ● IEEE 802.2/802.3 Encapsulation (BPDU/GMRP/GVRP, NETBIOS, IPX) ● NetWare 802.3 RAW Encapsulation (IPX) ● IPv6 Packet (Etype = 0x86DD) ● PPPoE frame (if Etype = 0x8863) or (if Etype = 0x8864 and Protocol !== 0021)

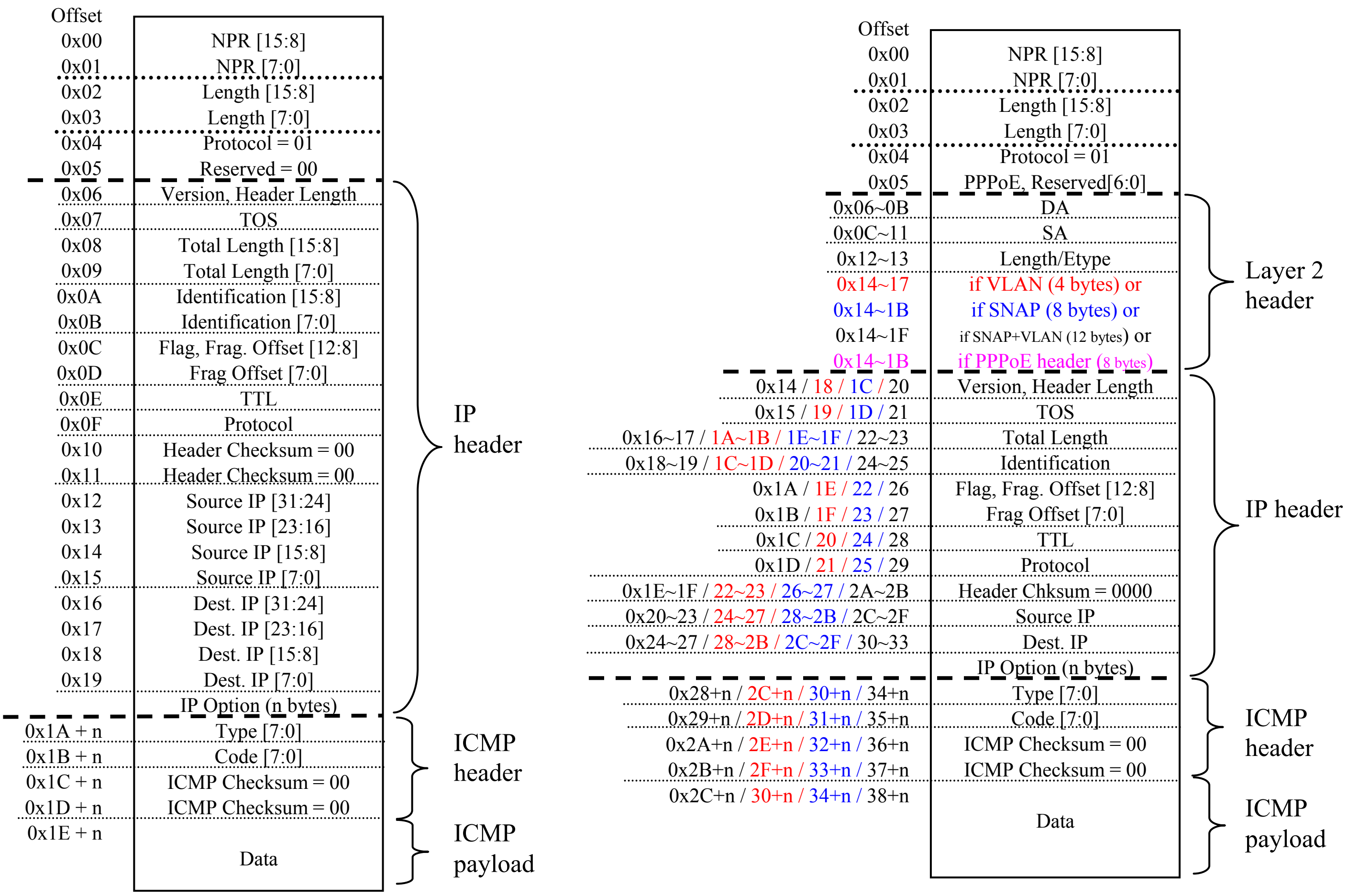
Figure 87: Non-IP-type Packet Format in RPBR

Transmit Packet Buffer Ring

In TOE TX direction, software is responsible for en-queuing the transmitted packets into TPBR while the L4_Engine is responsible for de-queuing the transmitted packets out of the TPBR. The packets in the ring are linked together via Next Pointer (NPR) field in each packet as shown below. The DMA transfer mechanism is used to move transmitted packets from TPBR through TOE to Ethernet MAC transmit buffer. Below Figure 88 to Figure 92 shows the packet format in TPBR.

1. ICMP Packet Format in TPBR

L2-Engine in Non-Transparent Mode **L2-Engine in Transparent Mode**

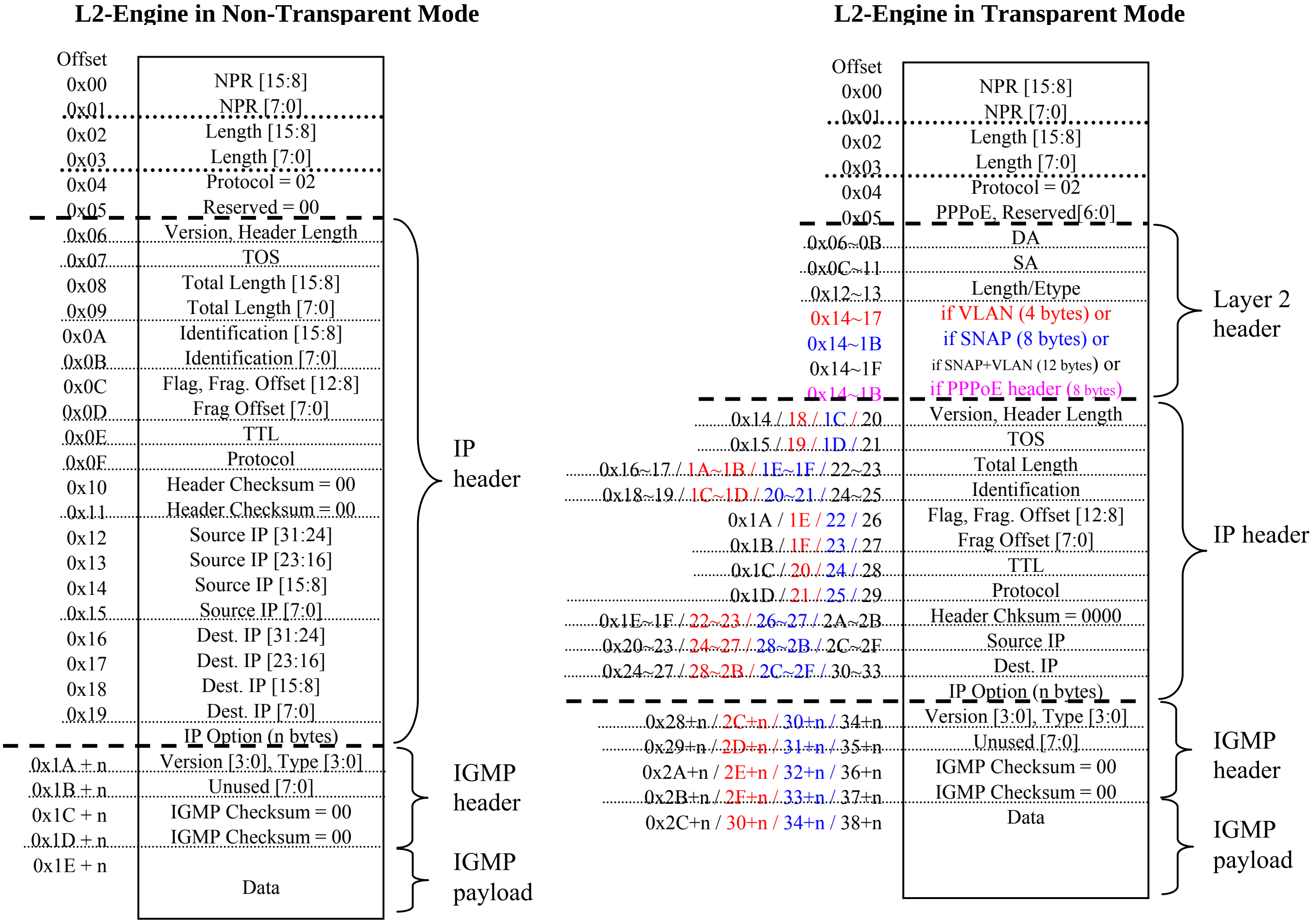

Offset
0x00 NPR [15:8]
0x01 NPR [7:0]
0x02 Length [15:8]
0x03 Length [7:0]
0x04 Protocol = 01
0x05 Reserved = 00
0x06 Version, Header Length
0x07 TOS
0x08 Total Length [15:8]
0x09 Total Length [7:0]
0x0A Identification [15:8]
0x0B Identification [7:0]
0x0C Flag, Frag. Offset [12:8]
0x0D Frag Offset [7:0]
0x0E TTL
0x0F Protocol
0x10 Header Checksum = 00
0x11 Header Checksum = 00
0x12 Source IP [31:24]
0x13 Source IP [23:16]
0x14 Source IP [15:8]
0x15 Source IP [7:0]
0x16 Dest. IP [31:24]
0x17 Dest. IP [23:16]
0x18 Dest. IP [15:8]
0x19 Dest. IP [7:0]
IP Option (n bytes)
0x1A + n Type [7:0]
0x1B + n Code [7:0]
0x1C + n ICMP Checksum = 00
0x1D + n ICMP Checksum = 00
0x1E + n Data
IP header
ICMP header
ICMP payload
Offset
0x00 NPR [15:8]
0x01 NPR [7:0]
0x02 Length [15:8]
0x03 Length [7:0]
0x04 Protocol = 01
0x05 PPPoE, Reserved[6:0]
0x06~0B DA
0x0C~11 SA
0x12~13 Length/Etype
0x14~17 if VLAN (4 bytes) or
0x14~1B if SNAP (8 bytes) or
0x14~1F if SNAP+VLAN (12 bytes) or
0x14~1B if PPPoE header (8 bytes)
0x14 / 18 / 1C / 20 Version, Header Length
0x15 / 19 / 1D / 21 TOS
0x16~17 / 1A~1B / 1E~1F / 22~23 Total Length
0x18~19 / 1C~1D / 20~21 / 24~25 Identification
0x1A / 1E / 22 / 26 Flag, Frag. Offset [12:8]
0x1B / 1F / 23 / 27 Frag Offset [7:0]
0x1C / 20 / 24 / 28 TTL
0x1D / 21 / 25 / 29 Protocol
0x1E~1F / 22~23 / 26~27 / 2A~2B Header Chksum = 0000
0x20~23 / 24~27 / 28~2B / 2C~2F Source IP
0x24~27 / 28~2B / 2C~2F / 30~33 Dest. IP
IP Option (n bytes)
0x28+n / 2C+n / 30+n / 34+n Type [7:0]
0x29+n / 2D+n / 31+n / 35+n Code [7:0]
0x2A+n / 2E+n / 32+n / 36+n ICMP Checksum = 00
0x2B+n / 2F+n / 33+n / 37+n ICMP Checksum = 00
0x2C+n / 30+n / 34+n / 38+n Data
Layer 2 header
IP header
ICMP header
ICMP payload

Offset	Bit	Field Name	Description
0x00	7:0	NPR [15:8]	The Next Pointers of TPBR: The NPR field indicates the first page of the next packet in the TPBR.
0x01	7:0	NPR [7:0]	
0x02	7:0	Length [11:8]	The Length field indicates the total length in bytes from (Version, Header Length) field to Data field in Non-Transparent mode, or from DA field to Data field in Transparent mode.
0x03	7:0	Length [7:0]	
0x04	7:0	Protocol	The Protocol = 0x01 indicating that the packet is an ICMP packet.
0x05	7	PPPoE	Set PPPoE flag to 1 when PPPoE header (8 bytes) is present in the packet format.
0x14~	7:0	VLAN or SNAP or VLAN+SNAP or PPPoE Header	If VLAN Tag (4 bytes) is present, the TL2CR[TX_VLAN_EN] should also be set to 1 to allow TOE to operate properly. If SNAP header (8 bytes) is present, the TL2CR[TX_SNAP_EN] should also be set to 1 to allow TOE to operate properly. If both VLAN Tag (4 bytes) and SNAP header (8 bytes) are present, the TL2CR[TX_VLAN_EN and TX_SNAP_EN] should also be set to 1 to allow TOE to

			operate properly. If PPPoE header (8 bytes) is present, the above PPPoE flag should also be set to 1 and TL2CR[TX_SNAP_EN] should be cleared to 0 to allow TOE to operate properly.

Figure 88: ICMP Packet Format in TPBR

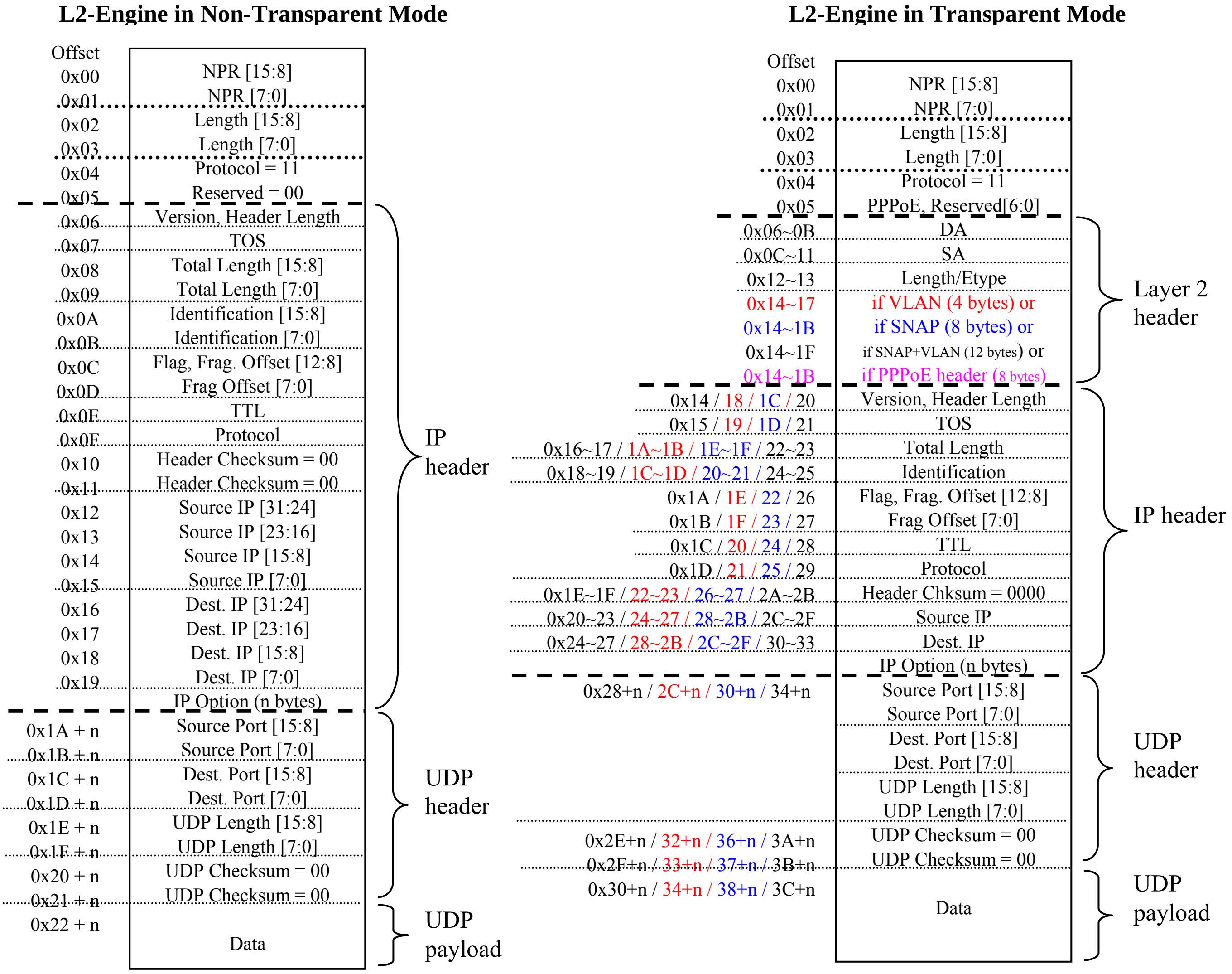
2. IGMP Packet Format in TPBR


L2-Engine in Non-Transparent Mode
Offset
0x00 NPR [15:8]
0x01 NPR [7:0]
0x02 Length [15:8]
0x03 Length [7:0]
0x04 Protocol = 02
0x05 Reserved = 00
0x06 Version, Header Length
0x07 TOS
0x08 Total Length [15:8]
0x09 Total Length [7:0]
0x0A Identification [15:8]
0x0B Identification [7:0]
0x0C Flag, Frag. Offset [12:8]
0x0D Frag Offset [7:0]
0x0E TTL
0x0F Protocol
0x10 Header Checksum = 00
0x11 Header Checksum = 00
0x12 Source IP [31:24]
0x13 Source IP [23:16]
0x14 Source IP [15:8]
0x15 Source IP [7:0]
0x16 Dest. IP [31:24]
0x17 Dest. IP [23:16]
0x18 Dest. IP [15:8]
0x19 Dest. IP [7:0]
IP Option (n bytes)
0x1A + n Version [3:0], Type [3:0]
0x1B + n Unused [7:0]
0x1C + n IGMP Checksum = 00
0x1D + n IGMP Checksum = 00
0x1E + n Data
IP header
IGMP header
IGMP payload
L2-Engine in Transparent Mode
Offset
0x00 NPR [15:8]
0x01 NPR [7:0]
0x02 Length [15:8]
0x03 Length [7:0]
0x04 Protocol = 02
0x05 PPPoE, Reserved[6:0]
0x06~0B DA
0x0C~11 SA
0x12~13 Length/Etype
0x14~17 if VLAN (4 bytes) or
0x14~1B if SNAP (8 bytes) or
0x14~1F if SNAP+VLAN (12 bytes) or
0x14~1B if PPPoE header (8 bytes)
0x14 / 18 / 1C / 20 Version, Header Length
0x15 / 19 / 1D / 21 TOS
0x16~17 / 1A~1B / 1E~1F / 22~23 Total Length
0x18~19 / 1C~1D / 20~21 / 24~25 Identification
0x1A / 1E / 22 / 26 Flag, Frag. Offset [12:8]
0x1B / 1F / 23 / 27 Frag Offset [7:0]
0x1C / 20 / 24 / 28 TTL
0x1D / 21 / 25 / 29 Protocol
0x1E~1F / 22~23 / 26~27 / 2A~2B Header Chksum = 0000
0x20~23 / 24~27 / 28~2B / 2C~2F Source IP
0x24~27 / 28~2B / 2C~2F / 30~33 Dest. IP
IP Option (n bytes)
0x28+n / 2C+n / 30+n / 34+n Version [3:0], Type [3:0]
0x29+n / 2D+n / 31+n / 35+n Unused [7:0]
0x2A+n / 2E+n / 32+n / 36+n IGMP Checksum = 00
0x2B+n / 2F+n / 33+n / 37+n IGMP Checksum = 00
0x2C+n / 30+n / 34+n / 38+n Data
Layer 2 header
IP header
IGMP header
IGMP payload

Offset	Bit	Field Name	Description
0x00	7:0	NPR [15:8]	Same as description for ICMP packet.
0x01	7:0	NPR [7:0]	
0x02	7:0	Length [11:8]	Same as description for ICMP packet.
0x03	7:0	Length [7:0]	
0x04	7:0	Protocol	The Protocol = 0x02 indicating that the packet is an IGMP packet.
0x05	7	PPPoE	Same as description for ICMP packet.
0x14~	7:0	VLAN or SNAP or VLAN+SNAP or PPPoE Header	Same as description for ICMP packet.

Figure 89: IGMP Packet Format in TPBR

3. UDP Packet Format in TPBR



L2-Engine in Non-Transparent Mode

Offset	Field	
0x00	NPR [15:8]	
0x01	NPR [7:0]	
0x02	Length [15:8]	
0x03	Length [7:0]	
0x04	Protocol = 11	
0x05	Reserved = 00	
0x06	Version, Header Length	IP header
0x07	TOS	
0x08	Total Length [15:8]	
0x09	Total Length [7:0]	
0x0A	Identification [15:8]	
0x0B	Identification [7:0]	
0x0C	Flag, Frag. Offset [12:8]	
0x0D	Frag Offset [7:0]	
0x0E	TTL	
0x0F	Protocol	
0x10	Header Checksum = 00	
0x11	Header Checksum = 00	
0x12	Source IP [31:24]	
0x13	Source IP [23:16]	
0x14	Source IP [15:8]	
0x15	Source IP [7:0]	
0x16	Dest. IP [31:24]	
0x17	Dest. IP [23:16]	
0x18	Dest. IP [15:8]	
0x19	Dest. IP [7:0]	
	IP Option (n bytes)	
0x1A + n	Source Port [15:8]	UDP header
0x1B + n	Source Port [7:0]	
0x1C + n	Dest. Port [15:8]	
0x1D + n	Dest. Port [7:0]	
0x1E + n	UDP Length [15:8]	
0x1F + n	UDP Length [7:0]	
0x20 + n	UDP Checksum = 00	
0x21 + n	UDP Checksum = 00	
0x22 + n	Data	UDP payload

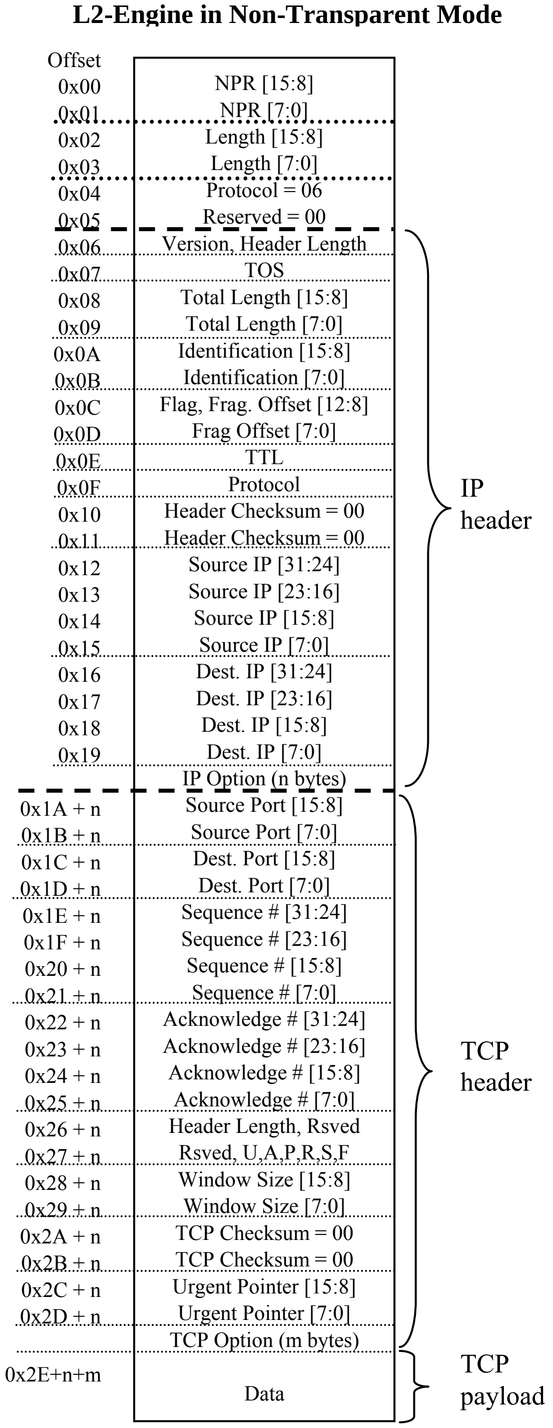
L2-Engine in Transparent Mode

Offset	Field	
0x00	NPR [15:8]	
0x01	NPR [7:0]	
0x02	Length [15:8]	
0x03	Length [7:0]	
0x04	Protocol = 11	
0x05	PPPoE, Reserved[6:0]	
0x06~0B	DA	Layer 2 header
0x0C~11	SA	
0x12~13	Length/Etype	
0x14~17	if VLAN (4 bytes) or	
0x14~1B	if SNAP (8 bytes) or	
0x14~1F	if SNAP+VLAN (12 bytes) or	
0x14~1B	if PPPoE header (8 bytes)	
0x14 / 18 / 1C / 20	Version, Header Length	IP header
0x15 / 19 / 1D / 21	TOS	
0x16~17 / 1A~1B / 1E~1F / 22~23	Total Length	
0x18~19 / 1C~1D / 20~21 / 24~25	Identification	
0x1A / 1E / 22 / 26	Flag, Frag. Offset [12:8]	
0x1B / 1F / 23 / 27	Frag Offset [7:0]	
0x1C / 20 / 24 / 28	TTL	
0x1D / 21 / 25 / 29	Protocol	
0x1E~1F / 22~23 / 26~27 / 2A~2B	Header Chksum = 0000	
0x20~23 / 24~27 / 28~2B / 2C~2F	Source IP	
0x24~27 / 28~2B / 2C~2F / 30~33	Dest. IP	
	IP Option (n bytes)	
0x28+n / 2C+n / 30+n / 34+n	Source Port [15:8]	UDP header
	Source Port [7:0]	
	Dest. Port [15:8]	
	Dest. Port [7:0]	
	UDP Length [15:8]	
	UDP Length [7:0]	
0x2E+n / 32+n / 36+n / 3A+n	UDP Checksum = 00	
0x2F+n / 33+n / 37+n / 3B+n	UDP Checksum = 00	
0x30+n / 34+n / 38+n / 3C+n	Data	UDP payload

Offset	Bit	Field Name	Description
0x00	7:0	NPR [15:8]	Same as description for ICMP packet.
0x01	7:0	NPR [7:0]	
0x02	7:0	Length [11:8]	Same as description for ICMP packet.
0x03	7:0	Length [7:0]	
0x04	7:0	Protocol	The Protocol = 0x11 indicating that the packet is an UDP packet.
0x05	7	PPPoE	Same as description for ICMP packet.
0x14~	7:0	VLAN or SNAP or VLAN+SNAP or PPPoE Header	Same as description for ICMP packet.

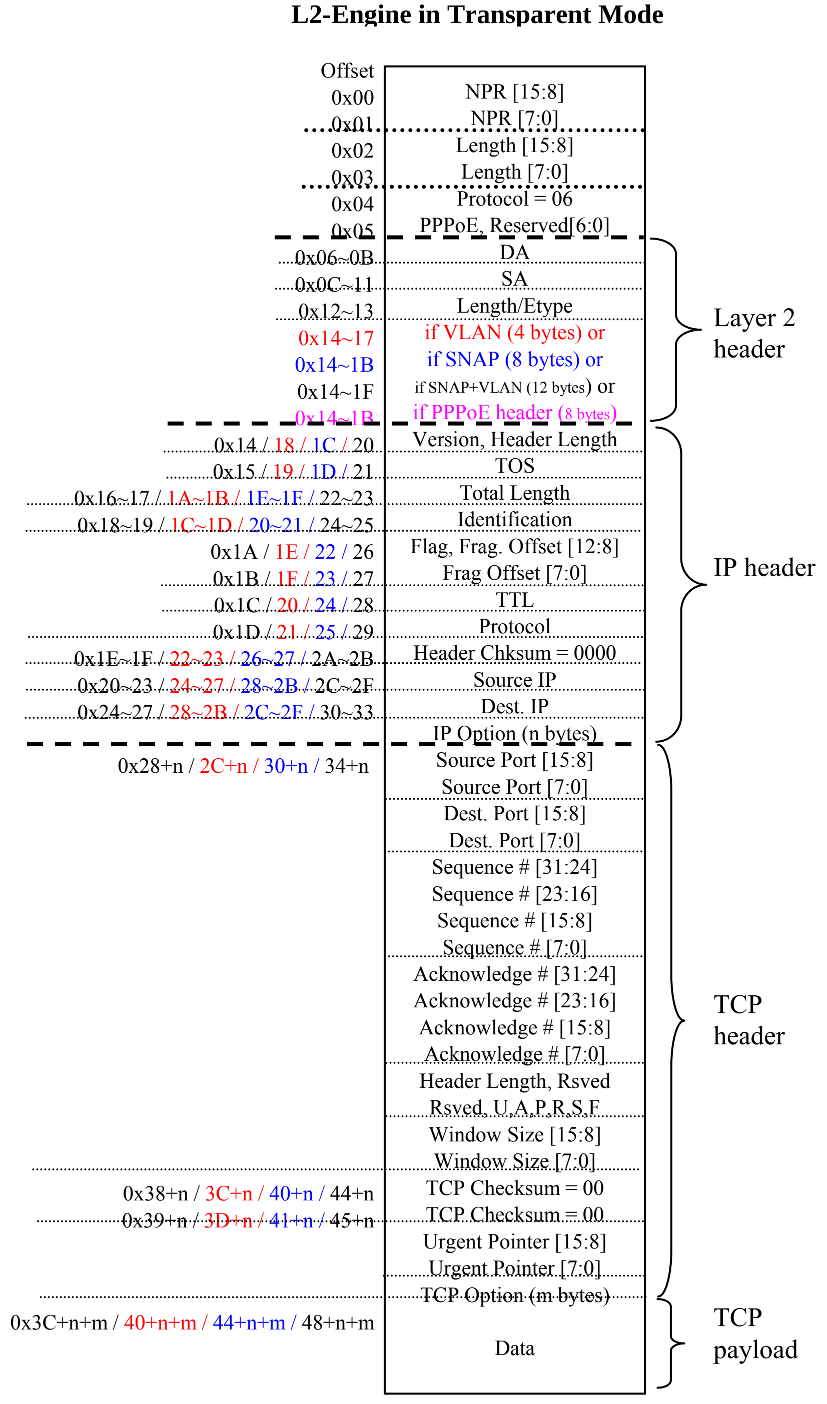
Figure 90: UDP Packet Format in TPBR

4. TCP Packet Format in TPBR



L2-Engine in Non-Transparent Mode

Offset	Field	
0x00	NPR [15:8]	
0x01	NPR [7:0]	
0x02	Length [15:8]	
0x03	Length [7:0]	
0x04	Protocol = 06	
0x05	Reserved = 00	
0x06	Version, Header Length	IP header
0x07	TOS	
0x08	Total Length [15:8]	
0x09	Total Length [7:0]	
0x0A	Identification [15:8]	
0x0B	Identification [7:0]	
0x0C	Flag, Frag. Offset [12:8]	
0x0D	Frag Offset [7:0]	
0x0E	TTL	
0x0F	Protocol	
0x10	Header Checksum = 00	
0x11	Header Checksum = 00	
0x12	Source IP [31:24]	
0x13	Source IP [23:16]	
0x14	Source IP [15:8]	
0x15	Source IP [7:0]	
0x16	Dest. IP [31:24]	
0x17	Dest. IP [23:16]	
0x18	Dest. IP [15:8]	
0x19	Dest. IP [7:0]	
	IP Option (n bytes)	
0x1A + n	Source Port [15:8]	TCP header
0x1B + n	Source Port [7:0]	
0x1C + n	Dest. Port [15:8]	
0x1D + n	Dest. Port [7:0]	
0x1E + n	Sequence # [31:24]	
0x1F + n	Sequence # [23:16]	
0x20 + n	Sequence # [15:8]	
0x21 + n	Sequence # [7:0]	
0x22 + n	Acknowledge # [31:24]	
0x23 + n	Acknowledge # [23:16]	
0x24 + n	Acknowledge # [15:8]	
0x25 + n	Acknowledge # [7:0]	
0x26 + n	Header Length, Rsved	
0x27 + n	Rsved, U,A,P,R,S,F	
0x28 + n	Window Size [15:8]	
0x29 + n	Window Size [7:0]	
0x2A + n	TCP Checksum = 00	
0x2B + n	TCP Checksum = 00	
0x2C + n	Urgent Pointer [15:8]	
0x2D + n	Urgent Pointer [7:0]	
	TCP Option (m bytes)	
0x2E+n+m	Data	TCP payload



L2-Engine in Transparent Mode

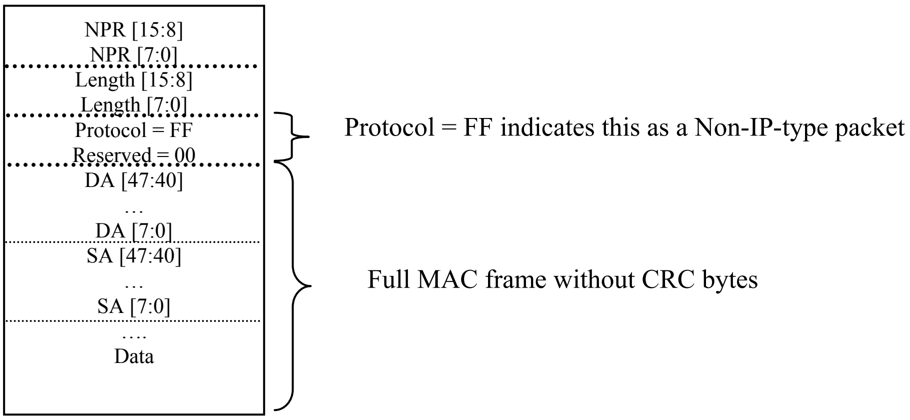
Offset	Field	
0x00	NPR [15:8]	
0x01	NPR [7:0]	
0x02	Length [15:8]	
0x03	Length [7:0]	
0x04	Protocol = 06	
0x05	PPPoE, Reserved[6:0]	
0x06~0B	DA	Layer 2 header
0x0C~11	SA	
0x12~13	Length/Etype	
0x14~17	if VLAN (4 bytes) or	
0x14~1B	if SNAP (8 bytes) or	
0x14~1F	if SNAP+VLAN (12 bytes) or	
0x14~1B	if PPPoE header (8 bytes)	
0x14 / 18 / 1C / 20	Version, Header Length	IP header
0x15 / 19 / 1D / 21	TOS	
0x16~17 / 1A~1B / 1E~1F / 22~23	Total Length	
0x18~19 / 1C~1D / 20~21 / 24~25	Identification	
0x1A / 1E / 22 / 26	Flag, Frag. Offset [12:8]	
0x1B / 1F / 23 / 27	Frag Offset [7:0]	
0x1C / 20 / 24 / 28	TTL	
0x1D / 21 / 25 / 29	Protocol	
0x1E~1F / 22~23 / 26~27 / 2A~2B	Header Chksum = 0000	
0x20~23 / 24~27 / 28~2B / 2C~2F	Source IP	
0x24~27 / 28~2B / 2C~2F / 30~33	Dest. IP	
	IP Option (n bytes)	
0x28+n / 2C+n / 30+n / 34+n	Source Port [15:8]	TCP header
	Source Port [7:0]	
	Dest. Port [15:8]	
	Dest. Port [7:0]	
	Sequence # [31:24]	
	Sequence # [23:16]	
	Sequence # [15:8]	
	Sequence # [7:0]	
	Acknowledge # [31:24]	
	Acknowledge # [23:16]	
	Acknowledge # [15:8]	
	Acknowledge # [7:0]	
	Header Length, Rsved	
	Rsved, U,A,P,R,S,F	
	Window Size [15:8]	
	Window Size [7:0]	
0x38+n / 3C+n / 40+n / 44+n	TCP Checksum = 00	
0x39+n / 3D+n / 41+n / 45+n	TCP Checksum = 00	
	Urgent Pointer [15:8]	
	Urgent Pointer [7:0]	
	TCP Option (m bytes)	
0x3C+n+m / 40+n+m / 44+n+m / 48+n+m	Data	TCP payload

Offset	Bit	Field Name	Description
0x00	7:0	NPR [15:8]	Same as description for ICMP packet.
0x01	7:0	NPR [7:0]	
0x02	7:0	Length [11:8]	Same as description for ICMP packet.
0x03	7:0	Length [7:0]	
0x04	7:0	Protocol	The Protocol = 0x06 indicating that the packet is a TCP packet.
0x05	7	PPPoE	Same as description for ICMP packet.

0x14~	7:0	VLAN or SNAP or VLAN+SNAP or PPPoE Header	Same as description for ICMP packet.

Figure 91: TCP Packet Format in TPBR

5. Non-IP-type Packet Format in TPBR



Offset	Bit	Field Name	Description
0x00	7:0	NPR [15:8]	Same as description for ICMP packet.
0x01	7:0	NPR [7:0]	
0x02	7	PCB	Same as description for ICMP packet.
0x02	6:4	BPBB	
0x02	3:0	Length [11:8]	The Length field indicates the total length in bytes from DA field to Data field regardless it's either in non-transparent or transparent mode.
0x03	7:0	Length [7:0]	
0x04	7:0	Protocol	When in L2 Engine Non-Transparent mode, the following packet encapsulation should be treated as Non-IP-type packet by software, i.e., software should put the Protocol field = 0xFF, ● IEEE 802.2/802.3 Encapsulation (BPDU/GMRP/GVRP, NETBIOS, IPX) ● NetWare 802.3 RAW Encapsulation (IPX) ● IPv6 Packet (Etype = 0x86DD) ● PPPoE frame (if Etype = 0x8863 or if Etype = 0x8864) When in L2 Engine Transparent mode, the following packet encapsulation should be treated as Non-IP-type packet by software, i.e., software should put the Protocol field = 0xFF, ● IEEE 802.2/802.3 Encapsulation (BPDU/GMRP/GVRP, NETBIOS, IPX) ● NetWare 802.3 RAW Encapsulation (IPX) ● IPv6 Packet (Etype = 0x86DD) ● PPPoE frame (if Etype = 0x8863) or (if Etype = 0x8864 and Protocol !== 0021)

Figure 92: Non-IP-type Packet Format in TPBR

4.15 10/100M Ethernet MAC

The 10/100 Ethernet MAC core block diagram is shown in Figure 93 below. It supports 802.3 and 802.3u MAC sub-layer functions as listed below,

- Ethernet MAC frame receive and transmit through MII interface
- With dedicated receive buffer of 8K bytes SRAM and transmit buffer of 4K bytes SRAM
- Flow-control support in full-duplex mode by monitoring receive buffer usage to compare with high water mark and low water mark for triggering flow control
- Received MAC frame CRC check and transmit MAC frame CRC generation
- Received packet filtering for broadcast, multicast, unicast, or CRC error MAC frames, etc. if enabled
- Support collision-detection, exponential backoff, packet retransmission, and backpressure in half-duplex mode
- Support Magic packet, predefined Wakeup frame, and Ethernet PHY linkup remote-wakeup mode. Upon detecting wakeup event, it can awake the AX11025 up from PMM or STOP mode
- Provides additional media-independent interface (MII) interface for interfacing external HomePlug PHY or HomePNA PHY functions

The 10/100M Ethernet MAC interfaces to both MII interface of the embedded 10/100M Ethernet PHY and the external MII interface I/O pins. The selection between the two MII interfaces is controlled by software via PCR (0x22) register.

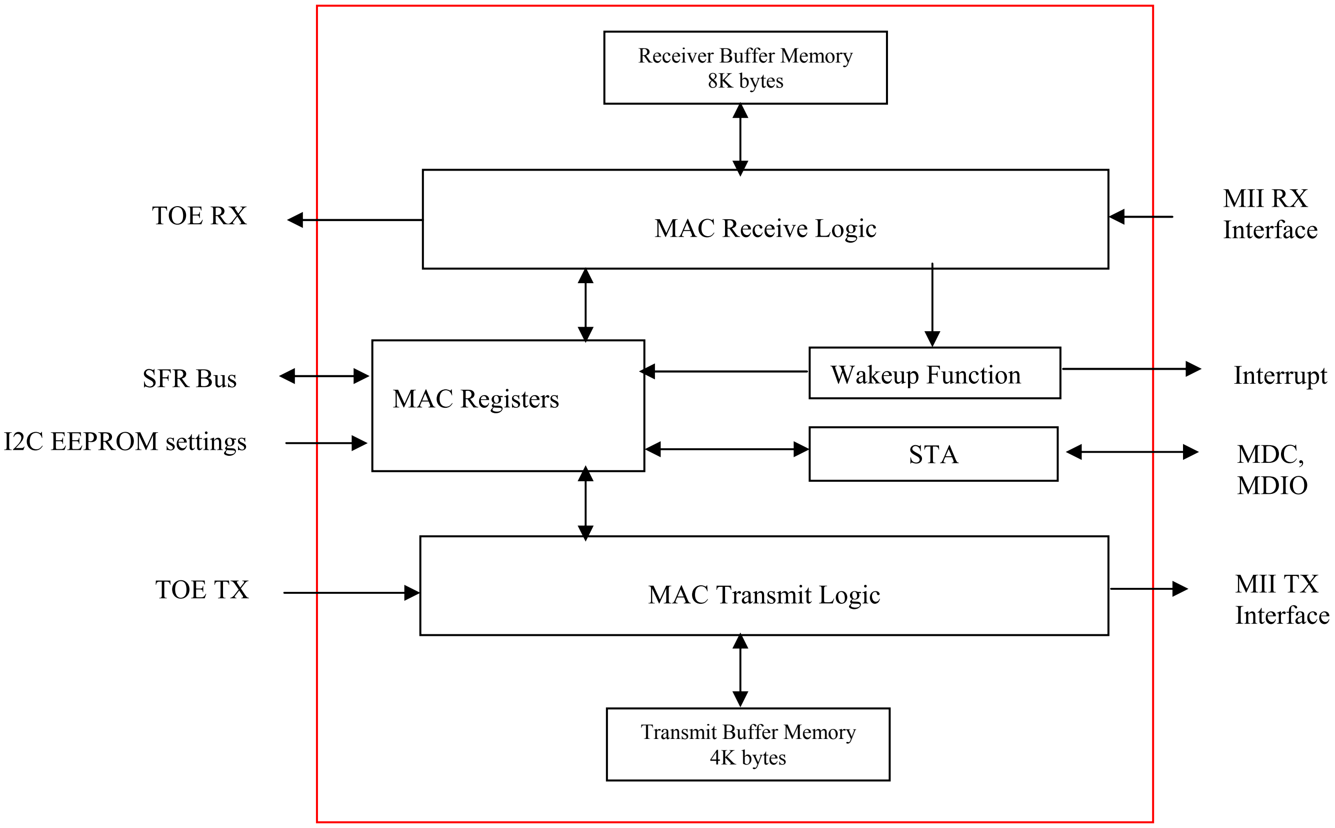


Figure 93: 10/100M Ethernet MAC Block Diagram

4.15.1 10/100M Ethernet MAC SFR Register Map

Address	Name	Description
0xB6	MCIR	MAC Command Index Register is used to indicate the address of Ethernet MAC registers.
0xB7	MDR	MAC Data Register is used to read data from or write data to the specified Ethernet MAC register.

Table 35: 10/100M Ethernet MAC SFR Register Map

MAC Command Index Register (MCIR, 0xB6)

Bit	7	6	5	4	3	2	1	0
Name	MCIR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	MCIR	WO	Indicate which of the Ethernet MAC register as listed in Table 36 is to be accessed.

MAC Data Register (MDR, 0xB7)

Bit	7	6	5	4	3	2	1	0
Name	MDR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	MDR	R/W	Data Register is used to write data to or read data from the Ethernet MAC registers.

10/100M Ethernet MAC Register Indirect Access Method

Software shall use indirect access method through MCIR and MDR registers to do read and write access to the 10/100M Ethernet MAC registers as listed in Table 36 below.

Read a register from 10/100M Ethernet MAC:

Step 1. Write MCIR: Software indicates the MAC register address to be accessed as the data and write it to the SFR register MCIR.

Step 2. Read MDR: Software then read SFR register MDR. The data read from MDR is the MAC register data indicated in step 1. Keep reading from MDR if the MAC registers have more than one byte, in that case, the first byte being read back is LSB byte.

Write a register to 10/100M Ethernet MAC:

Step 1. Write MDR: Software writes the data you want to write into MAC registers to the SFR register MDR. Keep writing to MDR if the MAC registers have more than one byte, in that case, the first byte being written should be LSB byte.

Step 2. Write MCIR: After writing MAC register data to MDR, software then indicates the target MAC register address as data and write it to MCIR.

Note: While software is reading or writing Ethernet MAC Registers during a sequence of SFR accesses, software can abort that process by writing MCIR with 0xFF.

10/100 Ethernet MAC Core Register Map

Address	Register Name	Description
0x00	RTSCR	RX/TX SRAM Command Register
0x02	RTSDR	RX/TX SRAM Data Register
0x0A	RCR	RX Control Register
0x0C	IPGCR	IPG Control Register
0x10	MACAR	MAC Address Register
0x16	MFA	Multicast Filter Array
0x1E	TR	Test Register
0x20	MSMR	Medium Status and Mode Register
0x22	PCR	PHY Control Register
0x24	SPWIE	STOP and PMM Wakeup Interrupt Enable Register
0x26	PLCIE	PHY Link Change Interrupt Enable Register
0x28	WPLS	Wakeup and PHY Link Status Register
0x30	WFCR	Wakeup Frame Command Register
0x32	WFBM0	Wakeup Frame Byte Mask 0 Register
0x36	WFCRC0	Wakeup Frame CRC 0 Register
0x38	WFOS0	Wakeup Frame Offset 0 Register
0x3A	WFLB0	Wakeup Frame Last Byte 0 Register
0x40	WFBM1	Wakeup Frame Byte Mask 1 Register
0x44	WFCRC1	Wakeup Frame CRC 1 Register
0x46	WFOS1	Wakeup Frame Offset 1 Register
0x48	WFLB1	Wakeup Frame Last Byte 1 Register
0xFF		Command Abort

Table 36: 10/100M Ethernet MAC Register Map

4.15.2 Ethernet MAC Receive Filtering

The address filtering logic compares the Destination Address field (first 6 bytes of the received packet) to the Ethernet MAC address registers (MACAR) of AX11025. If any one of the six bytes does not match the pre-programmed MACAR registers, the Ethernet MAC Receive Logic rejects the packet. This is for unicast address filtering. All multicast destination addresses are filtered using a hashing algorithm. See following description. If the multicast address indexes a bit that has been set in the filter bit array of the "Multicast Filter Array", the packet is accepted. Otherwise the Ethernet MAC rejects it. Each destination address is also checked for all 1's, which is the reserved broadcast address.

Unicast Packet Filtering

The MAC address registers (MACAR) are used to compare the destination address (DA[47:0]) of incoming packets for rejecting or accepting packets. Comparisons are performed on a byte wide basis. The bit assignment shown below relates the sequence in NODE_ADDR_0 – NODE_ADDR_5 registers to the bit sequence of the received packet.

	D7	D6	D5	D4	D3	D2	D1	D0
NODE_ADDR_0	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
NODE_ADDR_1	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8
NODE_ADDR_2	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16
NODE_ADDR_3	DA31	DA30	DA29	DA28	DA27	DA26	DA25	DA24
NODE_ADDR_4	DA39	DA38	DA37	DA36	DA35	DA34	DA33	DA32
NODE_ADDR_5	DA47	DA46	DA45	DA44	DA43	DA42	DA41	DA40

Note: The bit sequence of the received packet is DA0, DA1, …, DA7, DA8, …., DA47.

MAC Address Register (MACAR, 0x10)

Bit	7	6	5	4	3	2	1	0
NAME	NODE_ADDR 0							
	NODE_ADDR 1							
	NODE_ADDR 2							
	NODE_ADDR 3							
	NODE_ADDR 4							
	NODE_ADDR 5							
Reset Value	Reset value is determined by the I2C EEPROM							

Bit	Name	Access	Description
7:0 … 47:40	NODE_ADDR 0 … NODE_ADDR 5	R/W	The NODE_ADDR 5~0 is the MAC address of this device where NODE_ADDR 0 represents bit 7~0 of MAC address while NODE_ADDR 5 represents bit 47~40.

RX Control Register (RCR, 0x0A)

Bit	7	6	5	4	3	2	1	0
Name	SO	AC	AP	AM	AB	SEP	AMALL	PRO
Reset Value	0	1	0	1	1	0	0	0

Bit	Name	Access	Description
0	PRO	R/W	PRO: PACKET_TYPE_PROMISCUOUS. 1: All frames received by the Ethernet MAC are forwarded up toward the CPU. 0: Disabled (default).
1	AMALL	R/W	AMALL: PACKET_TYPE_ALL_MULTICAST. 1: All multicast frames received by the Ethernet MAC are forwarded up toward the CPU, not just the frames whose scrambling result of DA matching with multicast address list provided in Multicast Filter Array Register. 0: Disabled. This only allows multicast frames whose scrambling result of DA field matching with multicast address list provided in Multicast Filter Array Register to be forwarded up toward the CPU (default).
2	SEP	R/W	SEP: Save Error Packet. 1: Received packets with CRC error are saved and forwarded to the CPU anyway. 0: Received packets with CRC error are discarded automatically without forwarding to the CPU (default).
3	AB	R/W	AB: PACKET_TYPE_BROADCAST. 1: All broadcast frames received by the Ethernet MAC are forwarded up toward the CPU (default). 0: Disabled.
4	AM	R/W	AM: PACKET_TYPE_MULTICAST. 1: All multicast frames whose scrambling result of DA matching with multicast address list are forwarded up to the CPU (default). 0: Disabled.
5	AP	R/W	AP: Accept Physical Address from Multicast Filter Array. 1: Allow unicast packets to be forwarded up toward CPU if the lookup of scrambling result of DA is found within multicast address list defined in Multicast Filter Array Register. 0: Disabled, that is, unicast packets filtering are done without regarding multicast address list. This only allows unicast packets matching with MACAR register to be accepted (default).
6	AC	R/W	AC: Reserved bit. For normal operation, please always write 1 to this bit.
7	SO	R/W	SO: Start Operation of Ethernet MAC. 1: start operation. 0: stop operation and reset Ethernet MAC packet buffer (default).

Following is the truth table about unicast packet filtering condition.

DA Matching MACAR?	PRO bit	Broadcast or Multicast Packet?	Unicast Packet Filtered by Ethernet MAC?
No	0	No	Yes
No	1	No	No
Yes (see Note below)	0	No	No

Note: DA Matching MACAR including following two cases:

1. Destination Address field of incoming packets matches with MACAR.
2. When AP (RCR.5) is set to 1and the scrambling result of DA is found within multicast address list.

Multicast Packet Filtering

As shown in Figure 94 below, the Multicast Filter Array (MFA) provides filtering of multicast addresses hashed through the CRC logic. All Destination Address field are fed through the 32 bits CRC generation logic and as the last bit of the Destination Address field enters the CRC, the 6 most significant bits of the CRC generator are latched. These 6 bits are then decoded by a 1 to 64 decoder to index a unique filter bit (FB0-63) in the Multicast Filter Array. If the filter bit selected is set, the multicast packet is accepted. The system designer would use a program to determine which filter bits to set in the multicast registers. All multicast filter bits that correspond to Multicast Filter Array Registers accepted by the node are then set to one. To accept all multicast packets all of the registers are set to all ones. Note that received Pause Frames are always filtered by Ethernet MAC regardless of MFA setting.

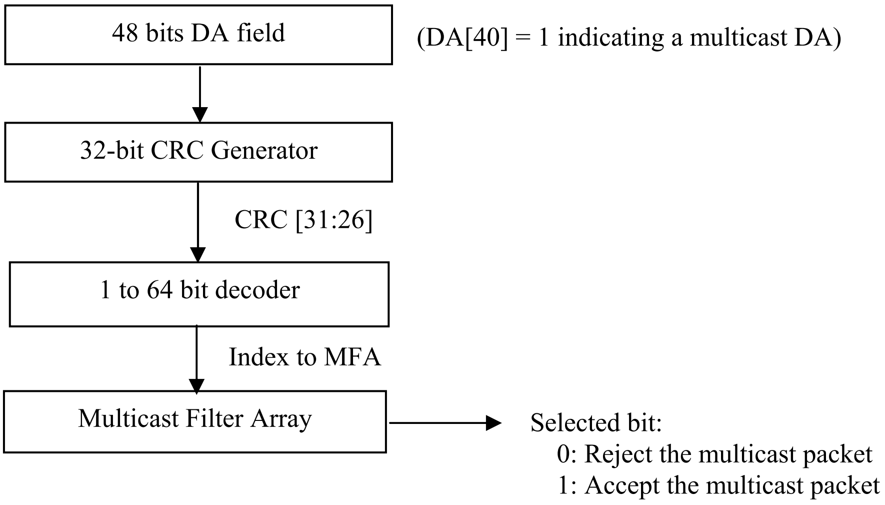


Figure 94: Multicast Filter Array Hashing Algorithm

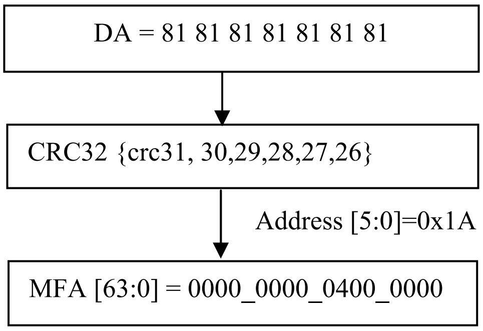
Example: If the accepted multicast packet's destination address Y is found to hash to the value 32 (0x20), then FB32 in MA4 should be initialized to "1". This will allow the Ethernet MAC to accept any multicast packet with the destination address Y. Although the hashing algorithm does not guarantee perfect filtering of multicast address, it will perfectly filter up to 64 logical address filters if these addresses are chosen to map into unique locations in the multicast filter. Note: The LSB bit of received packet's first byte being "1" signifies a Multicast Address.

	D7	D6	D5	D4	D3	D2	D1	D0
MA0	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
MA1	FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8
MA2	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
MA3	FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24
MA4	FB39	FB38	FB37	FB36	FB35	FB34	FB33	FB32
MA5	FB47	FB46	FB45	FB44	FB43	FB42	FB41	FB40
MA6	FB55	FB54	FB53	FB52	FB51	FB50	FB49	FB48
MA7	FB63	FB62	FB61	FB60	FB59	FB58	FB57	FB56

Figure 95: Multicast Filter Array Bit Mapping

Multicast Filter Array (MFA, 0x16)

Bit	7	6	5	4	3	2	1	0
NAME	MA 0							
	MA 1							
	MA 2							
	MA 3							
	MA 4							
	MA 5							
	MA 6							
	MA 7							
Reset Value	0x0000_0000_0000_0000							

Bit	Name	Access	Description
7:0 … 63:56	MA 0 … MA 7	R/W	The MA 7~0 is the multicast address bit map used by multicast frame filtering block where MA 0 represents bit 7~0 while MA 7 represents bit 63~56. For example.  DA = 81 81 81 81 81 81 81 CRC32 {crc31, 30,29,28,27,26} Address [5:0]=0x1A MFA [63:0] = 0000_0000_0400_0000

Following is the truth table about multicast packet filtering condition.

PRO bit	AMALL bit	AM bit	Pass Hashing Algorithm?	Multicast Packet Filtered by Ethernet MAC?
0	0	0	0	Yes
0	0	0	1	Yes
0	0	1	0	Yes
0	0	1	1	No
0	1	0/1	0/1	No
1	0/1	0/1	0/1	No

Note: Passing Hashing Algorithm means that the selected bit in MFA of CRC-32 result is set to “1”.

Broadcast Packet Filtering

The broadcast filtering logic compares the Destination Address field (first 6 bytes of the received packet) to all 1's, that is, the values are "FFFF_FFFF_FFFF" in Hex format. If any bit of the six bytes does not equal to 1's, the Ethernet MAC rejects the packet if both unicast and multicast receive condition are not met.

Following is a truth table about broadcast packet filtering condition.

PRO bit	AB bit	Broadcast Packet?	Broadcast Packet Filtered by Ethernet MAC?
0	1	Yes	No
0	0	Yes	Yes
1	0/1	Yes	No

CRC-Error Packet Filtering

Normally, all the packets received with CRC error will be rejected by Ethernet MAC. When SEP bit (RCR.2) is enabled the packet with CRC error will be received and forwarded to CPU.

Packet Filtering During Remote-Wakeup Enable Mode

When CPU entering STOP or PMM mode with the remote wakeup function being enabled, the packet receive function of Ethernet MAC will be disabled and all the packets received will be filtered.

The Magic Packet wakeup function is enabled by RWMP bit (SPWIE.4). The external pin, EXT_WKUP, wakeup function is enabled by EPWT bit (SPWIE.5). The Microsoft Wakeup Frame wakeup Function can be enabled by MWFE bit (SPWIE.6) and WFCR bit 0 or 2.

Following is the truth table about packet filtering condition during remote-wakeup enable mode.

RWMP bit (SPWIE.4)	EPWT bit (SPWIE.5)	EWFF0 bit (WFCR.0)	EWFF1 bit (WFCR.2)	Received Packet Type?	Packet Filtered by Ethernet MAC?
0	0	0	0	Any packets but Magic Packet	Based on unicast, multicast, and broadcast filtering rule
0	0	0	0	Magic Packet	No
1	0	0	0	Any packets but Magic Packet	CPU not in STOP/PMM mode -> No. CPU in STOP/PMM mode -> Yes.
1	0	0	0	Magic packet	CPU not in STOP/PMM mode -> No. CPU in STOP/PMM mode -> Yes and the CPU will be awaked up.
0	1	0	0	Any packets	No
0	0	10, 01, or 11		Any packets but Wakeup Frame	CPU not in STOP/PMM mode -> No. CPU in STOP/PMM mode -> Yes.
0	0	10, 01, or 11		Microsoft Wakeup Frame	CPU not in STOP/PMM mode -> No. CPU in STOP/PMM mode -> Yes and the CPU will be awaked up.
1	1	10, 01, or 11		Any packets but Magic Packet and Wakeup Frame	CPU not in STOP/PMM mode -> No. CPU in STOP/PMM mode -> Yes.
1	1	10, 01, or 11		Magic Packet or Wakeup Frame	CPU not in STOP/PMM mode -> No. CPU in STOP/PMM mode -> Yes and the CPU will be awaked up.

Table 37: Packet Filtering During Remote-Wakeup Enable Mode

Note that when the primary or secondary PHY linkup wakeup function is enabled, it normally means the Ethernet PHY link is down during PMM or STOP mode. Therefore, there will be no packets coming in to Ethernet MAC from the Ethernet PHY so the packet filtering is meaningless here.

4.15.3 Ethernet MAC Packet Transmit

Preamble, Sync, Padding and CRC

When transmitting the Ethernet packets, the Ethernet MAC will automatically append the preamble and sync byte at the beginning of the packet. It will also generate the padding bytes (if transmitted packet size is less than 60 bytes) and the 4 bytes CRC field at the end of the packet (if ACB bit in Flag byte in I2C EEPROM is enabled). The minimum size of an Ethernet packet without preamble is 64 bytes.

Preamble	7 bytes
Synch	1 byte
Destination Address	6 bytes
Source Address	6 bytes
Length/Type	2 bytes
Data + Padding (if needed)	Min 46 bytes
CRC	4 bytes

Figure 96: Ethernet Packet Format

Collision

During transmission when operating in half duplex mode, the Ethernet MAC monitors the collision signal from Ethernet PHY to determine if a collision has occurred. If a collision is detected, the Ethernet MAC will reset the FIFO and restore the transmit pointers for retransmission of the packet based on exponential backoff algorithm. If 15 retransmissions each result in a collision, the transmission will be aborted and the transmitted packet is discarded after 16 transmission attempts.

Medium Status and Mode Register (MSMR, 0x20)

Bit	7	6	5	4	3	2	1	0
Name	RFC	PS	FD	TFC	PF	RE	BPC	SM
Reset Value	1	1	1	1	0	0	0	0

Bit	Name	Access	Description
0	SM	R/W	SM: Super Mac support. 1: Enable Super Mac to shorten exponential back-off time during each transmit retries. 0: Disabled (default).
1	BPC	R/W	BPC: Backpressure Continuously. 1: When TFC bit = 1, setting this bit enables backpressure on TX direction “continuously” during RX buffer full condition in half duplex mode. 0: When TFC bit = 1, setting this bit enable backpressure on TX direction “intermittently” during RX buffer full condition in half duplex mode (default).
2	RE	R/W	RE: Receive Enable. 1: Enable RX path of the ASIC. 0: Disabled (default).
3	PF	R/W	PF: Check only “length/type” field for Pause Frame. 1: Enable, i.e., Pause frame is identified only based on L/T filed. 0: Disabled, i.e., Pause frames are identified based on both DA and L/T fields (default).
4	TFC	R/W	TFC: TX Flow Control enable. 1: Enable transmitting pause frame on TX direction in full duplex mode or enable transmitting jam pattern on TX direction in half duplex mode during RX buffer’s free buffer less than low water mark setting (default). 0: Disabled.
5	FD	R/W	FD: Full Duplex mode 1: Full Duplex mode (default). 0: Half Duplex mode.

6	PS	R/W	PS: Port Speed in Ethernet MAC. 1: 100 Mbps (default). 0: 10 Mbps.
7	RFC	R/W	RFC: RX Flow Control enable. 1: Enable receiving pause frame on RX direction during full duplex mode (default). 0: Disabled.

IPG Control Register (IPGCR, 0x0C)

Bit	7	6	5	4	3	2	1	0
NAME	CPEF	IPG 0						
	Reserved	IPG 1						
	Reserved	IPG 2						
Reset Value	0x12_0C_95							

Bit	**Name**	**Access**	**Description**
6:0	IPG 0	R/W	IPG 0 [6:0]: Inter Packet Gap for back-to-back transfer on TX direction in Ethernet MAC (default = 15h).
7	CPEF	R/W	Capture Effective setting in half duplex operation. 1: To shorten backoff time for 2nd collision retransmission in half duplex mode (default). 0: Normal exponential backoff time is used for 2nd collision retransmission in half duplex mode.
14:8	IPG 1	R/W	IPG1 [6:0]: IPG part1 value (default = 0Ch). Bit 15 is reserved.
22:16	IPG 2	R/W	IPG2 [6:0]: IPG part1 value + part2 value (default = 12h). Bit 23 is reserved.

Test Register (TR, 0x1E)

Bit	7	6	5	4	3	2	1	0
Name	Reserved				ABORT	Reserved	CRC_ER	LDRND
Reset Value	00_0000						0	0

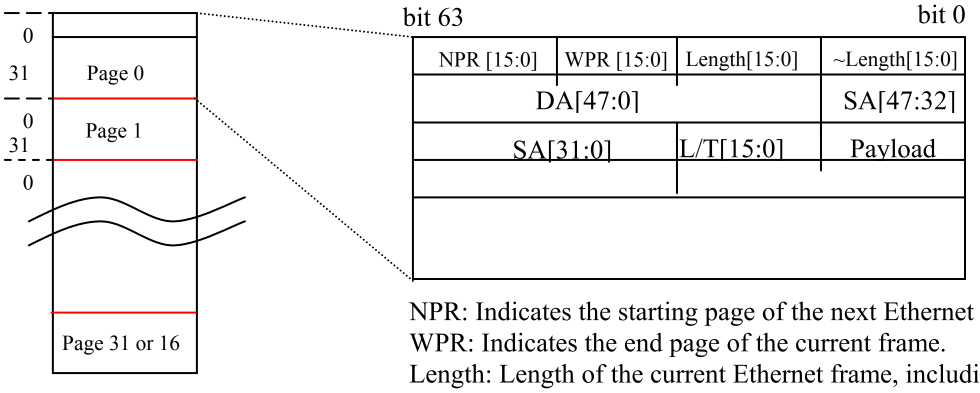
Bit	**Name**	**Access**	**Description**
0	LDRND	R/W	LDRND: Load Random number into Ethernet MAC’s exponential back-off timer. User writes a “1” to enable loading a small random number into MAC’s back-off timer to shorten the back-off duration in each retry after collision. This register is used for test purpose. Default value = 0.
1	CRC_ER	CR	This bit will be ‘1’ whenever receiving packets with CRC error. This bit will be clear after software reads it.
2	Reserved		
3	ABORT	CR	In half duplex mode, this bit will be ‘1’ whenever the transmitted frame has been aborted and dropped by Ethernet MAC after 15 retransmission attempts. This bit is not used in full duplex mode.
7:4	Reserved		

4.15.4 Ethernet MAC Buffer Management

Receive/Transmit Packet Buffer SRAM Map

The Ethernet MAC embeds a dedicated 8K bytes SRAM for its Receive Packet Buffering and a dedicated 4K bytes SRAM for its Transmit Packet Buffering. The RX Packet Buffer is divided into 32 pages while the TX Packet Buffer is divided into 16 pages. Each page has 256 bytes of storage space.

Figure 97 shows the data structure of the Packet Buffer memory. The first 8 bytes preceding each Ethernet packets are status bytes. After the 8 status bytes is the Ethernet packet DA, SA, LT fields, etc. Both Receive and Transmit Packet Buffer memory can be access via RTSCR and RTSDR registers described below. Note that these two registers are used only for debug purpose and normal packet receive/transmit should be accessed through TOE as described in section 4.14.



NPR: Indicates the starting page of the next Ethernet frame.
WPR: Indicates the end page of the current frame.
Length: Length of the current Ethernet frame, including CRC field.
~Length: Bitwise negation of Length field.

Figure 97: Ethernet Packet Buffer Data Structure in Ethernet MAC

Rx/Tx SRAM Command Register (RTSCR, 0x00)

Bit	7	6	5	4	3	2	1	0
Name	ADDR[7:0]							
	READ	GO	TX	Reserved			ADDR[9:8]	
Reset Value	0x0000							

Bit	Name	Access	Description
7:0 9:8	ADDR	R/W	The read/write address {ADDR [9:8], ADDR [7:0]} of RX Packet SRAM. The read/write address {ADDR [8], ADDR [7:0]} of TX Packet SRAM.
13	TX	R/W	RAM selection. 0: indicates to read/write to RX Packet SRAM. 1: indicates to read/write to TX Packet SRAM. [4:2] Reserved
14	GO	R/W1	Setting GO bit to "1" to initiates the SRAM read or write access request to the internal SRAM arbiter. This bit will remain "1" while the access request is still in progress and be cleared automatically by arbiter after current access request is completed. Note: software can only write"1" to this bit and can't write"0".
15	READ	R/W	Setting READ bit to "1" indicates to read from SRAM. Setting READ bit to "0" indicates to write to SRAM.

Rx/Tx SRAM Data Register (RTSDR, 0x02)

Bit	7	6	5	4	3	2	1	0
Name	BUFFER_DATA 0							
	BUFFER_DATA 1							
	BUFFER_DATA 2							
	BUFFER_DATA 3							
	BUFFER_DATA 4							
	BUFFER_DATA 5							
	BUFFER_DATA 6							
	BUFFER_DATA 7							
Reset Value	0x0000_0000_0000_0000							

Bit	**Name**	**Access**	**Description**
7:0 … 63:56	BUFFER_DATA 0 … BUFFER_DATA 7	R/W	The BUFFER_DATA 7~0 is the content of the RX or TX SRAM where BUFFER_DATA 0 represents bit 7~0 of the SRAM while BUFFER_DATA 7 represents bit 63~56. When writing to the SRAM, software needs to first write desired data into this register before issuing RTSCR register. When reading from the SRAM, software first issues RTSCR register and then retrieves the SRAM data from this register.

Flow Control in Full Duplex Mode

Flow Control on RX Direction

Flow control in RX direction is used to avoid RX packet buffer being overflowed in full-duplex mode, it's enabled by TFC and FD bit (MSMR bit 4,5). The Ethernet MAC uses a Buffer Ring structure to store the received packets (see Figure 98) and maintain a Free Buffer Counter (FBC). The FBC is counting the free memory pages. When flow control is enabled, the Ethernet MAC will send out Pause ON frame to notify the other end to stop sending packets when its FBC is less than "Low Water Mark" being defined in I2C EEPROM offset 0x11. The format of the Pause Frame is shown in Figure 99. The Ethernet MAC will send out Pause OFF frame when its FBC is over "High Water Mark" being defined in I2C EEPROM offset 0x10.

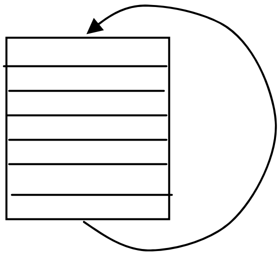


Figure 98: Transmit/Receive Buffer Ring Structure

DA =	SA=	L/T=		Pause Time	Padding
0180_C200_0001	0000_0000_0000	8808	0001	FF00	

Pause On Frame

DA =	SA=	L/T=		Pause Time	Padding
0180_C200_0001	0000_0000_0000	8808	0001	0000	

Pause OFF Frame

Figure 99: Pause Frame

Flow Control on TX Direction

When enabled RFC bit (MSMR.7) in full duplex mode, the Ethernet MAC will stop sending packets out towards the other end, upon receiving the Pause ON Frame. It will resume transmitting packets after the pause timer times out. The duration of this pause timer is specified in the received packet's Pause Time Field. The Ethernet MAC can recognize either one of the following conditions as a Pause Frame:

- Destination Address field equals to the multicast address, 01-80-C2-00-00-01, and Length/Type Field = 0x8808
- Length/Type Field = 0x8808

Above condition is determined by PF bit (MSMR.3).

Backpressure in Half Duplex Mode

The backpressure mechanism is used to avoid RX packet buffer being overflowed in half-duplex mode, it's enabled by TFC bit (MSMR.4). When backpressure mechanism is enabled, the Ethernet MAC will send out Jam pattern to force collision and force the other end to backoff from current transmission when its FBC is less than "Low Water Mark". The format of this Jam pattern consists of 16 bytes of 0x55.

4.15.5 Magic Packet and Wakeup Frame

When the PMM or STOP mode is invoked, the CPU can be awaked up by two types of Ethernet frame - Magic Packet or Microsoft Wakeup Frame. Note that during these two types of packet remote wakeup mode, the Ethernet PHY shall not be set in reset or power down mode in BMCR register to allow Ethernet packet reception.

Magic Packet Wakeup Function

Magic Packet is an easy and simple MAC layer Ethernet frame used to awake up the AX11025. Setting RWMP bit (SPWIE.4) prior to entering STOP or PMM mode can enable Magic Packet Wakeup Function. Once the Ethernet MAC has been put into the Magic Packet wakeup enable mode, it monitors all incoming frames for a specific data sequence. This sequence can be located anywhere after the Length/Type field but is preceded by a synchronization stream (6 bytes of 0xFF). The data sequence is 16 iterations of the MAC address of this chip. See Figure 100 below. When Ethernet MAC detects this type of packet, it will generate interrupt on INT6 to awake up the CPU and report this wakeup event in WPLS register.

Assume the MAC address of this chip is "00123456789A".

DA	SA	L/T	Payload
00123456789a	xxxxxxxxxxxx	0800	xxxx_FFFFFFFFFFFF_00123456789a_00123456789a_00123456789a_00123456789a_00123456789a_00123456789a_00123456789a_00123456789a_00123456789a_00123456789a_00123456789a_00123456789a_00123456789a_00123456789a_00123456789a_00123456789a_xxxxxxxx

Figure 100: Example Magic Packet Format

Wakeup Frame Wakeup Function.

Wakeup Frame is used to awake up the AX11025 from receiving a more complex Ethernet frame, which can be defined with specific TCP/IP header and payload pattern. User can define the pattern of this wakeup frame and set MWFE bit (SPWIE.6) and either EWFF0 or WSFF1 bit in WFCR register to enable this Wakeup Frame wakeup function prior to entering STOP or PMM mode. Once enabled, the Ethernet MAC monitors all incoming frames for this user-defined pattern. If matched, the Ethernet MAC will generate interrupt on INT6 to awake up the CPU and report this wakeup event in WPLS register.

There are two filter sets supported in the Ethernet MAC, which can define two different patterns of Wakeup Frame. The filter set consists of Wakeup Frame Command Register, Wakeup Frame Byte Mask Register, Wakeup Frame CRC Register, Wakeup Frame Offset Register, and Wakeup Frame Last Byte Register. Also, if a more complex pattern of Wakeup Frame is needed, user can choose to cascade the two filter sets into one and specify a longer pattern for Wakeup Frame matching.

STOP and PMM Wakeup Interrupt Enable Register (SPWIE, 0x24)

Bit	7	6	5	4	3	2	1	0
Name	Reserved	MWFE	EPWT	RWMP	Reserved	SPLWE	Reserved	PPLWE
Reset Value	0	0	0	0	0	0	0	0

Bit	**Name**	**Access**	**Description**
0	PPLWE	R/W	PPLWE: Primary PHY Linkup Wakeup Enable Register 1: Enable Interrupt 0: Disable Interrupt (Default)
1	Reserved		
2	SPLWE	R/W	SPLWE: Secondary PHY Linkup Wakeup Enable Register 1: Enable Interrupt 0: Disable Interrupt (Default)
3	Reserved		
4	RWMP	R/W	RWMP: Remote Wakeup trigger by Magic Packet. 1: Enable. 0: Disabled (default).
5	EPWT	R/W	EPWT: External Pin Wakeup trigger 1: Enable 0: Disabled (default).
6	MWFE	R/W	MWFE: Microsoft Wakeup Frame Enable Register 1: Enable 0: Disabled (default).
7	Reserved		

Note:

1. The CPU can be awaked up by various wakeup events if software enables this register. Upon enabled, the wakeup events will trigger the INT 6 of the CPU.
2. After the CPU awakes up, software can read WPLS register to identify the source of wakeup events.

PHY Link Change Interrupt Enable Register (PLCIE, 0x26)

Bit	7	6	5	4	3	2	1	0
Name	Reserved					SLCIE	Reserved	PLCIE
Reset Value	0					0	0	0

Bit	**Name**	**Access**	**Description**
0	PLCIE	R/W	PLCIE: Primary PHY Link Change Interrupt Enable Register. 1:Enable Primary PHY Interrupt. 0:Disable Primary PHY Interrupt (Default).
1	Reserved		
2	SLCIE	R/W	SLCIE: Second PHY Link Change Interrupt Enable Register. 1:Enable Second PHY Interrupt. 0:Disable Second PHY Interrupt (Default).
7:3	Reserved		

Note:

1. The Ethernet MAC will check the link status of both primary and secondary Ethernet PHY for every 200ms, if the link status is changed and above PLCIE is also enabled, an interrupt on INT 4 is generated to CPU (both link-up or link-down transition can cause an interrupt to be generated)
2. After the CPU is interrupted, the software can read WPLS register on PPLSCR, PPLSR, SPLSCR, and SPLSR bits to learn about the latest link status.

Wakeup and PHY Link Status Register (WPLS, 0x28)

Bit	7	6	5	4	3	2	1	0
Name	Reserved	MWFSR	EWPSR	MPSR	SPLSR	SPLSCR	PPLSR	PPLSCR
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	PPLSCR	CR	PPLSR: Primarily PHY Link Status Changed Register. 1: Whenever Primarily PHY's Link status changed, this bit will be set to 1. 0: This bit will be cleared after software reads it.
1	PPLSR	R	PPLSR: Primarily PHY Link Status Register. 1: The link status of Primarily PHY is link-up. 0: The link status of Primarily PHY is link-down.
2	SPLSCR	CR	SPLSR: Secondary PHY Link Status Changed Register. 1: Whenever Secondary PHY's Link status changed, this bit will be set to 1. 0: This bit will be cleared after software reads it.
3	SPLSR	R	SPLSR: Secondary PHY Link Status Register. 1: The link status of Secondary PHY is link-up. 0: The link status of Secondary PHY is link-down.
4	MPSR	CR	MPSR: Magic Packet Status Register. 1: Whenever Ethernet MAC receives Magic Packet and awakes up the CPU. 0: This bit will be cleared after software reads it.
5	EWPSR	CR	EWPSR: External Wakeup Pin Status Register. 1: Whenever user triggers the external wakeup pin, EXT_WKUP, and awakes up the CPU. 0: This bit will be cleared after software reads it.
6	MWFSR	CR	MWFSR: Microsoft Wakeup Frame Status Register. 1: Whenever Ethernet MAC receives Microsoft Wakeup Frame and awakes up the CPU. 0: This bit will be cleared after software reads it.
7	Reserved		

Note:

1. Both SPWIE and PLCIE use WPLS register to report status, the software should read this register after receiving INT 6 or INT 4.

Wakeup Frame Command Register (WFCR, 0x30)

Bit	7	6	5	4	3	2	1	0
Name	Reserved			ECFF	EUM1	EWFF1	EUM0	EWFF0
Reset Value	000			0	0	0	0	0

Bit	Name	Access	Description
0	EWFF0	RW	Enable Wakeup Frame Filter 0 (WFF0), which consists of WFBM0, WFCRC0, WFOS0, and WFLB0 registers. Please program WFBM0, WFCRC0, WFOS0, and WFLB0 registers before setting this bit. 1: Enabled wakeup frame detection mode for WFF0. 0: Disabled wakeup frame detection mode for WFF0.
1	EUM0	RW	Enable Unicast Match mode for Wakeup Frame Filter 0. 1: When receiving frame with DA equal to MACAR and WFF0 is matched, then the packet is considered as valid wakeup frame. 0: When receiving frame with any DA but the WFF0 is matched, then the packet is considered as valid wakeup frame.
2	EWFF1	RW	Enable Wakeup Frame Filter 1 (WFF1), which consists of WFBM1, WFCRC1, WFOS1, and WFLB1 registers. Please program WFBM1, WFCRC1, WFOS1, and WFLB1 registers before setting this bit. 1: Enabled wakeup frame detection mode for WFF1. 0: Disabled wakeup frame detection mode for WFF1.

3	EUM1	RW	Enable Unicast Match mode for Wakeup Frame Filter 1. 1: When receiving frame with DA equal to MACAR and WFF1 is matched, then the packet is considered as valid wakeup frame. 0: When receiving frame with any DA but the WFF1 is matched, then the packet is considered as valid wakeup frame.
4	ECFF	RW	Enable Cascading wakeup Frame Filter 1: Enable cascading the WFF0 and WFF1 into one filter. When enabled, the WFBM0, WFBM1, WFOS0, WFOS1, WFLB1, WFCRC1 will be used, and the WFCRC0 and WFLB0 registers are not used. When enabled, the EWFF0, EWFF1 should both be set to 1 at the same time, and the EUM0, EUM1 should both be set to 1 or 0 at the same time. The value of WFOS1 indicates the offset from the last byte of first filter, WFF0. For example, if WFOS0 = 0x08, and WFOS1 = 0x06, then the first bit of WFBM1 is used as byte mask for the ((8*2) + 32 + (6*2) + 1)th byte in the wakeup frame. In other words, the first bit of WFBM1 is the byte mask of ((WFOS0*2) + 32 + (WFOS1*2) +1)th byte in the wakeup frame. 0: The WFF0 and WFF1 functions as two independent wakeup frame filters.
7:5	Reserved		

Wakeup Frame Byte Mask 0 Register (WFBM0, 0x32)

Bit	7	6	5	4	3	2	1	0
Name	BM_0_0							
	BM_0_1							
	BM_0_2							
	BM_0_3							
Reset Value	0x0000_0000							

Bit	Name	Access	Description
7:0 … 31:24	BM_0_0 … BM_0_3	RW	BM_0_3 ~ 0_0 is the 32 bit for byte mask. The byte mask defines which bytes in the incoming frame will be examined to determine whether or not this is a wake-up frame. The BM_0_0 represents bit 7~0 and BM_0_3 represents bit 31~24.

Wakeup Frame CRC 0 Register (WFCRC0, 0x36)

Bit	7	6	5	4	3	2	1	0
Name	CRC_0_0							
	CRC_0_1							
Reset Value	0x0000							

Bit	Name	Access	Description
7:0 15:8	CRC_0_0 CRC_0_1	RW	This register defines the 16-bit CRC value of the valid wakeup frames. Software should calculate this based on the valid wakeup frame patterns, WFOS0 and WFBM0 settings. This value is used to compare with the CRC calculated on the incoming frame, when matched and the WFLB0 is also matched, then the frame is considered as valid wakeup frame. CRC_0_0 represents bit 7~0 and CRC_0_1 represents bit 15~8. CRC-16 Polynomials = X^16 + X^15 + X^2 + 1

Wakeup Frame Offset 0 Register (WFOS0, 0x38)

Bit	7	6	5	4	3	2	1	0
Name	OFFSET_0							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	OFFSET_0	RW	This register defines the offset of the first byte in the incoming frame from which the CRC is calculated for the wakeup frame recognition. Each value represents two bytes in the frame. For example: The offset value of 0 is the first byte of the incoming frame's destination address. The offset value of 1 is the 3rd byte of the incoming frame, etc.

Wakeup Frame Last Byte 0 Register (WFLB0, 0x3A)

Bit	7	6	5	4	3	2	1	0
Name	LB_0							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	LB_0	RW	This 1-byte pattern is used to compare with the last masked byte in the incoming frame. The last masked byte is the byte of the last bit mask being 1 in WFBM0. A valid wakeup frame shall have both WFCRC0 and WFLB0 match conditions.

Wakeup Frame Byte Mask 1 Register (WFBM1, 0x40)

Bit	7	6	5	4	3	2	1	0
Name	BM_1_0							
	BM_1_1							
	BM_1_2							
	BM_1_3							
Reset Value	0x0000_0000							

Bit	Name	Access	Description
7:0 … 31:24	BM_1_0 … BM_1_3	RW	BM_1_3 ~ 1_0 is the 32 bit for byte mask. The byte mask defines which bytes in the incoming frame will be examined to determine whether or not this is a wake-up frame. The BM_1_0 represents bit 7~0 and BM_1_3 represents bit 31~24.

Wakeup Frame CRC 1 Register (WFCRC1, 0x44)

Bit	7	6	5	4	3	2	1	0
Name	CRC_1_0							
	CRC_1_1							
Reset Value	0x0000							

Bit	Name	Access	Description
7:0 15:8	CRC 1_0 CRC 1_1	RW	This register defines the 16-bit CRC value of the valid wakeup frames. Software should calculate this based on the valid wakeup frame patterns, WFOS0 and WFBM0 settings. This value is used to compare with the CRC calculated on the incoming frame, when matched and the WFLB0 is also matched, then the frame is considered as valid wakeup frame. CRC_1_0 represents bit 7~0 and CRC_1_1 represents bit 15~8. CRC-16 Polynomials = X^16 + X^15 + X^2 + 1.

Wakeup Frame Offset 1 Register (WFOS1, 0x46)

Bit	7	6	5	4	3	2	1	0
Name	OFFSET_1							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	OFFSET_1	RW	This register defines the offset of the first byte in the incoming frame from which the CRC is calculated for the wakeup frame recognition. Each value represents two bytes in the frame. For example: The offset value of 0 is the first byte of the incoming frame's destination address. The offset value of 1 is the 3rd byte of the incoming frame, etc.

Wakeup Frame Last Byte 1 Register (WFLB1, 0x48)

Bit	7	6	5	4	3	2	1	0
Name	LB_1							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	LB_1	RW	This 1-byte pattern is used to compare with the last masked byte in the incoming frame. The last masked byte is the byte of the last bit mask being 1 in WFBM0. A valid wakeup frame shall have both WFCRC1 and WFLB1 match conditions.

Wakeup Frame Example

If following packet pattern is defined as a Wakeup Frame and the Ethernet MAC will monitor packet with DA = 1234_5678_9abc and L/T=0800.

Field	DA						SA						L/T		Payload																	
Byte Mask	1	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mask Data	12	34	56	78	9a	bc							08	00																		

Software should set following registers,

1. SPWIE: Enable MWFE bit.
2. WFBM0: Set to 0x0000_303f, where BM_0_0 = 0x3f (indicating byte 0 to 5 are used to compare and byte 6~7 are not used), BM_0_1 = 0x30, BM_0_2 = 0x00, and BM_0_3 = 0x00.
3. WFCRC0: The pattern is "12, 34, 56, 78, 9a, bc, 08, 00", so the CRC-16 result is 0x8fbb.
4. WFOS0: The pattern is compared at the first byte of packet, so offset is 0x00.
5. WFLB0: The last byte of the pattern is 00, so set this register to 0x00.
6. WFCR: Set this register to 0x01 to enable the filter set 0. This register should be set at the end of the procedure.

PHY Control Register (PCR, 0x22)

Bit	7	6	5	4	3	2	1	0
Name	Reserved					IPRL	Reserved	PSEL
Reset Value	0					1	1	1

Bit	**Name**	**Access**	**Description**
0	PSEL	R/W	PSEL: PHY Select. 1: Select embedded 10/100 Ethernet PHY (default). 0: Select external PHY, which is attached to the MII interface.
1	Reserved		
2	IPRL	R/W	IPRL: Internal Ethernet PHY Reset control. This bit controls the reset signal of internal Ethernet PHY. 1: Internal Ethernet PHY is in operating state (default). 0: Internal Ethernet PHY in reset state.
7:3	Reserved		

4.16 10/100M Ethernet PHY

The 10/100 Ethernet PHY of AX11025 is compliant with IEEE 802.3 and IEEE 802.3u standards. It contains an on-chip crystal oscillator, PLL-based clock multiplier, and digital phase-locked loop for data/timing recovery. It provides over-sampling mixed-signal transmit drivers complying with 10/100BASE-TX transmit wave-shaping / slew rate control requirements. It has robust mixed-signal loop adaptive equalizer for receiving signal recovery.

- Support full-duplex mode, half-duplex mode, and auto-negotiation
- Support twisted pair crossover detection and auto-correction (Auto-MDIX)
- DSP-based adaptive line equalizer, providing superior immunity to near end crosstalk and inter-symbol interference
- Fully compliant with 100BASE-TX, and 10BASE-T PMD level standards (IEEE 802.3u and IEEE 802.3)
- DSP-controlled symbol timing recovery circuit
- Baseline wander corrective circuits to compensate data dependent offset due to AC coupling transformers
- Over-sampling mixed-signal transmit driver complies with 10/100BASE-TX transmit wave-shaping/slew-control requirements

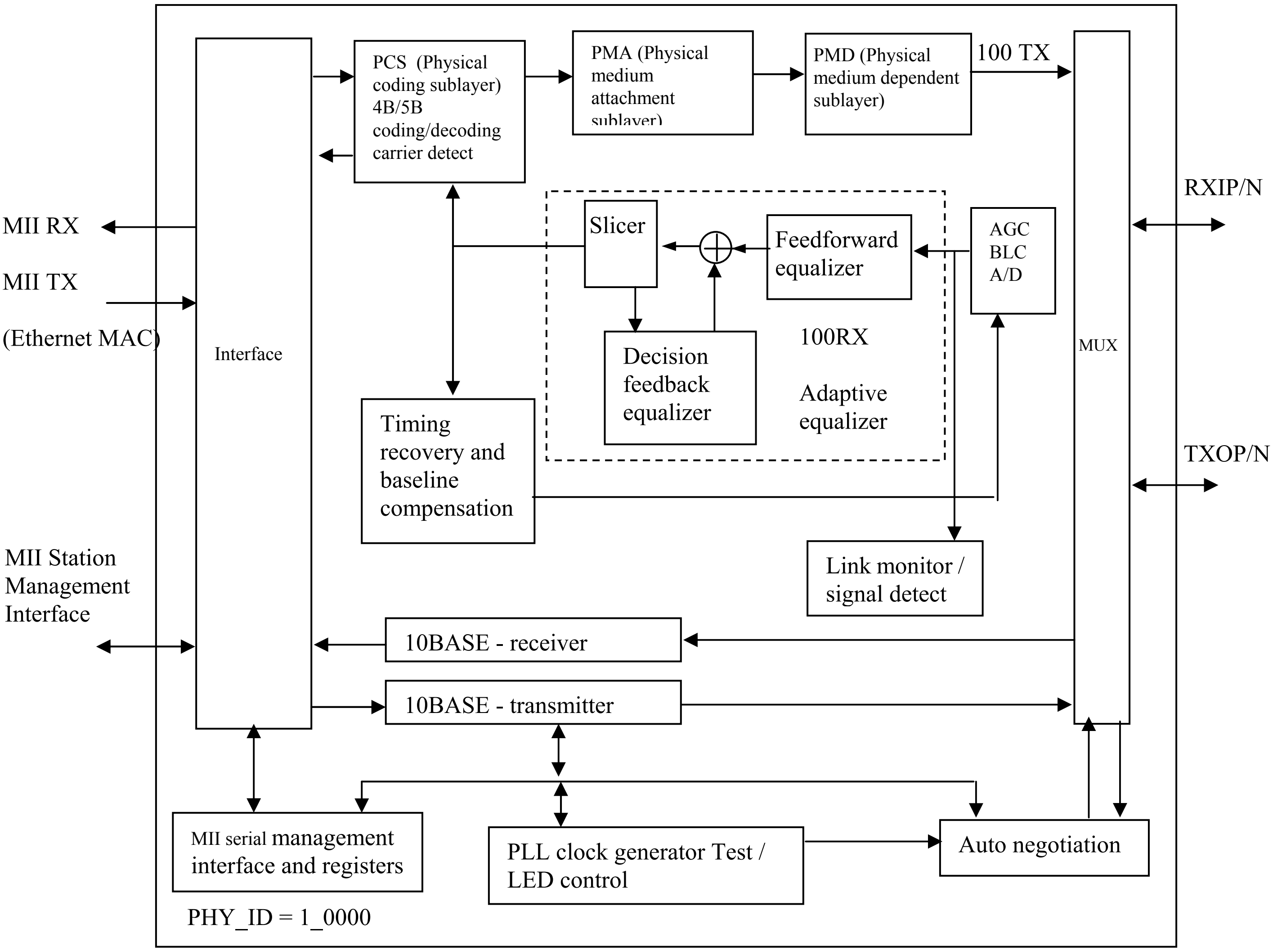

MII RX
MII TX
(Ethernet MAC)
MII Station Management Interface
Interface
PCS (Physical coding sublayer) 4B/5B coding/decoding carrier detect
PMA (Physical medium attachment sublayer)
PMD (Physical medium dependent sublayer)
100 TX
RXIP/N
TXOP/N
MUX
AGC BLC A/D
Slicer
Feedforward equalizer
Decision feedback equalizer
100RX
Adaptive equalizer
Timing recovery and baseline compensation
Link monitor / signal detect
10BASE - receiver
10BASE - transmitter
MII serial management interface and registers
PHY_ID = 1_0000
PLL clock generator Test / LED control
Auto negotiation

Figure 101: 10/100M Ethernet PHY Block Diagram

4.16.1 MII Station Management Function

The primary function of station management is to transfer control and status information of the Ethernet PHY to a management entity. This function is accomplished by the MDC clock input from Ethernet MAC (frequency is about 1.5 MHz) along with the MDIO signal to/from the Ethernet PHY. The embedded Ethernet PHY's ID address is fixed to "1_0000". Frames transmitted on the MII management interface will have the frame structure shown in Figure 102. The order of bit transmission is from left to right. Note that reading and writing the management register must be completed without interruption.

Read/Write	Preamble	ST	OP	PHY ID	REGAD	TA	DATA	IDLE
Read	1. . .1	01	10	AAAAA	RRRRR	Z0	DDDDDDDDDDDDDDDD	Z
Write	1. . .1	01	01	AAAAA	RRRRR	10	DDDDDDDDDDDDDDDD	Z

Field	Description
Preamble	**Preamble** of MII station management frame, which consists of 32 bits of 1.
ST	**Start of Frame.** The start of frame is indicated by a 01 pattern.
OP	**Operation Code**. The operation code for a read transaction is 10. The operation code for a write transaction is a 01.
PHY ID	**PHY Address**. The PHY address is 5 bits, allowing for 32 unique addresses. The first PHY address bit transmitted and received is the MSB of the address. A station management entity that is attached to multiple PHY entities must have prior knowledge of the appropriate PHY address for each entity. The embedded Ethernet PHY's address is fixed to "1_0000".
REGAD	**Register Address.** The register address is 5 bits, allowing for 32 unique registers within each PHY. The first register address bit transmitted and received is the MSB of the address.
TA	**Turnaround**. The turnaround time is a two-bits time spacing between the register address field, and the data field of a frame, to avoid drive contention on MDIO during a read transaction. During a write to the PHY, these bits are driven to 10 by the station. During a read, the MDIO is not driven during the first bit time and is driven to a 0 by the PHY during the second bit time.
DATA	**Data**. The data field is 16 bits. The first bit transmitted and received will be bit 15 of the register being addressed.
IDLE	**Idle Condition.** The IDLE condition on MDIO is a high-impedance state. All three state drivers will be disabled and the PHY's pull-up resistor will pull the MDIO line to logic 1.

Figure 102: MII Station Management Frame Format

4.16.2 10/100M Ethernet PHY SFR Register Map

Address	Name	Description
0xBE	EPCR	Ethernet PHY Command Register
0xBF	EPDR	Ethernet PHY Data Register

Table 38: 10/100 Ethernet PHY SFR Register Map

Ethernet PHY Command Register (EPCR, 0xBE)

Bit	7	6	5	4	3	2	1	0
Name	Reserved			REG_ADDR				
	READ	GO	Reserved	PHYID				
Reset Value	0x0000							

Bit	**Name**	**Access**	**Description**
4:0	REG_ADDR	R/W	The PHY register address to be accessed as listed in Table 39.
7:5	Reserved		These bits must be 0, except for "command abort" command.
12:8	PHYID	R/W	The PHY ID value. When accessing the embedded Ethernet PHY, write "1_0000".
13	Reserved		This bit must be 0, except for "command abort" command.
14	GO	R/W1	Setting GO bit to "1" to initiate read or write access request to the Ethernet PHY registers. This bit will remain "1" while the access request is still in progress and be cleared to 0 automatically after current access request is completed. Note: software can only write"1" to this bit and can't write"0".
15	READ	R/W	Setting READ bit to "1" indicates to **read** data from Ethernet PHY through EPDR. Setting READ bit to "0" indicates to **write** data to Ethernet PHY through EPDR.

Note: Command Abort operation: While software is reading/writing EPCR, the command can be aborted by writing EPCR register with data = 0xFF. After generating the "Command Abort operation", the following EPCR read/write command will start with accessing bit 7~0 of EPCR.

Ethernet PHY Data Register (EPDR, 0xBF)

Bit	7	6	5	4	3	2	1	0
NAME	PHY_DATA 0							
	PHY_DATA 1							
Reset Value	0x0000							

Bit	**Name**	**Access**	**Description**
7:0 15:8	PHY_DATA 0 PHY_DATA 1	R/W	The PHY_DATA 1~0 is the register data being written to or read back from the Ethernet PHY. When writing to Ethernet PHY, software needs to first write desired data into this register before issuing EPCR register. When reading from the Ethernet PHY, software first issues EPCR register and then retrieves the read data from this register.

Note: Command Abort operation: While software is reading/writing EPDR, the command can be aborted by writing EPCR register with data = 0xFF. After generating the "Command Abort operation", the following EPDR read/write command will start with accessing bit 7~0 of EPDR.

10/100M Ethernet PHY Register Indirect Access Method

Software shall use indirect access method through EPCR and EPDR registers to do read and write access to the Ethernet PHY registers as listed in Table 39.

Read a register from 10/100M Ethernet PHY:

Step 1. Write EPCR two times: Software indicates REG_ADDR in first write and indicates PHYID and sets READ=1 and GO=1 in second write to SFR register EPCR.

Step 2. Read EPCR: Software should keep reading EPCR until GO bit become 0. When GO bit is clear, the Ethernet PHY register data is presented on SFR register EPDR. Note that each EPCR read cycle consists of two SFR bus read accesses to retrieve the 16 bits wide data in EPCR.

Step 3. Read EPDR two times: Software then read SFR register EPDR which now stores the read data of Ethernet PHY register provided in step 1. Note that the first read returns the 7:0 bits of Ethernet PHY register data.

Write a register to 10/100M Ethernet PHY:

Step 1. Write EPDR two times: Software writes the data you want to write into Ethernet PHY register to SFR register EPDR. The first write is the LSB byte that maps to Ethernet PHY register's 7:0 bits.

Step 2: Write EPCR two times: Software indicates REG_ADDR in first write and indicates PHYID and sets READ=0 and GO=1 in second write to SFR register EPCR.

Step 3: Read EPCR: Software should keep reading EPCR until GO bit become 0. When GO bit is clear, the requested write to Ethernet PHY register is completed. Note that each EPCR read cycle consists of two SFR bus read accesses to retrieve the 16 bits wide data in EPCR.

Embedded 10/100M Ethernet PHY Register Map

Address	Register Name	Description
0h	BMCR	Basic mode control register, basic register.
1h	BMSR	Basic mode status register, basic register.
2h	PHYIDR1	PHY identifier register 1, extended register.
3h	PHYIDR2	PHY identifier register 2, extended register.
4h	ANAR	Auto negotiation advertisement register, extended register.
5h	ANLPAR	Auto negotiation link partner ability register, extended register.
6h	ANER	Auto negotiation expansion register, extended register.
7h	Reserved	Reserved and currently not supported.
8h-Fh	IEEE reserved	IEEE 802.3u reserved.

Table 39: Embedded 10/100M Ethernet PHY Register Map

Basic Mode Control Register (BMCR, 0x00)

Bit	Bit Name	Reset Value	Access	Description
15	Reset	0	R/W/SC	Reset. 1: Software reset. 0: Normal operation.
14	Loopback	0	R/W	Loopback. 1: Loopback enabled. 0: Normal operation.
13	Speed selection	1	R/W	Speed selection. 1: 100 Mb/s. 0: 10 Mb/s.
12	Auto-negotiation enable	1	R/W	Auto-negotiation enable. 1: Auto-negotiation enabled. Bits 8 and 13 of this register are ignored when this bit is set. 0: Auto-negotiation disabled. Bits 8 and 13 of this register determine the link speed and mode.
11	Power down	0	R/W	Power down. 1: Power down. 0: Normal operation.
10	Isolate	,	R/W	Isolate. (PHYAD = 00000) 1: Isolate. 0: Normal operation.
9	Restart auto-negotiation	0	R/W/SC	Restart auto-negotiation. 1: Restart auto-negotiation. 0: Normal operation.
8	Duplex mode	1	R/W	Duplex mode. 1: Full duplex operation. 0: Normal operation.
7	Collision test	0	R/W	Collision test. 1: Collision test enabled. 0: Normal operation.
6:0	Reserved		RO	

Basic Mode Status Register (BMSR, 0x01)

Bit	Bit Name	Reset Value	Access	Description
15	100BASE-T4	0	RO/PS	100BASE-T4 capable. 0: This PHY is not able to perform in 100BASE-T4 mode.
14	100BASE-TX full duplex	1	RO/PS	100BASE-TX full-duplex capable. 1: This PHY is able to perform in 100BASE-TX full-duplex mode.
13	100BASE-TX half duplex	1	RO/PS	100BASE-TX half-duplex capable. 1: This PHY is able to perform in 100BASE-TX half-duplex mode.
12	10BASE-T full duplex	1	RO/PS	10BASE-T full-duplex capable. 1: This PHY is able to perform in 10BASE-T full-duplex mode.
11	10BASE-T half duplex	1	RO/PS	10BASE-T half-duplex capable. 1: This PHY is able to perform in 10BASE-T half-duplex mode.
10:7	Reserved	0	RO	Reserved. Write as 0, read as “don’t care”.
6	MF preamble suppression	0	RO/PS	Management frame preamble suppression. 0: This PHY will not accept management frames with preamble suppressed.
5	Auto-negotiation complete	0	RO	Auto-negotiation completion. 1: Auto-negotiation process completed. 0: Auto-negotiation process not completed.
4	Remote fault	0	RO/LH	Remote fault. 1: Remote fault condition detected (cleared on read or by a chip reset). 0: No remote fault condition detected.
3	Auto-negotiation ability	1	RO/PS	Auto configuration ability. 1: This PHY is able to perform auto-negotiation.
2	Link status	0	RO/LL	Link status. 1: Valid link established (100Mb/s or 10Mb/s operation) 0: Link not established.
1	Jabber detect	0	RO/LH	Jabber detection. 1: Jabber condition detected. 0: No Jabber condition detected.
0	Extended capability	1	RO/PS	Extended capability. 1: Extended register capable. 0: Basic register capable only.

PHY Identifier Register 1 (PHYIDR1, 0x02)

Bit	Bit Name	Reset Value	Access	Description
15:0	OUI_MSB	0x003B	RO/PS	OUI most significant bits. Bits 3 to 18 of the OUI are mapped to bits 15 to 0 of this register respectively. The most significant two bits of the OUI are ignored.

PHY Identifier Register 2 (PHYIDR2, 0x03)

Bit	Bit Name	Reset Value	Access	Description
15:10	OUI_LSB	00_0110	RO/PS	OUI least significant bits: Bits 19 to 24 of the OUI are mapped to bits 15 to 10 of this register respectively.
9:4	VNDR_MDL	00_0101	RO/PS	Vendor model number.
3:0	MDL_REV	0001	RO/PS	Model revision number.

Auto Negotiation Advertisement Register (ANAR, 0x04)

Bit	Bit Name	Reset Value	Access	Description
15	NP	0	RO/PS	Next page indication. 0: No next page available. The PHY does not support the next page function.
14	ACK	0	RO	Acknowledgement. 1: Link partner ability data reception acknowledged 0: Not acknowledged
13	RF	0	R/W	Remote fault. 1: Fault condition detected and advertised 0: No fault detected
12:11	Reserved	X	R/W	Reserved. Write as 0, read as “don’t care”.
10	Pause	0	R/W	Pause. 1: Pause operation enabled for full-duplex links 0: Pause operation not enabled
9	T4	0	RO/PS	100BASE-T4 support. 0: 100BASE-T4 not supported
8	TX_FD	1	R/W	100BASE-TX full-duplex support. 1: 100BASE-TX full-duplex supported by this PHY. 0: 100BASE-TX full-duplex not supported by this PHY.
7	TX_HD	1	R/W	100BASE-TX half-duplex support: 1: 100BASE-TX half-duplex supported by this PHY. 0: 100BASE-TX half-duplex not supported by this PHY.
6	10_FD	1	R/W	10BASE-T full-duplex support. 1: 10BASE-T full-duplex supported by this PHY. 0: 10BASE-T full-duplex not supported by this PHY.
5	10_HD	1	R/W	10BASE-T half-duplex support. 1: 10BASE-T half-duplex supported by this PHY. 0: 10BASE-T half-duplex not supported by this PHY.
4:0	Selector	0_0001	R/W	Protocol selection bits. These bits contain the binary encoded protocol selector supported by this PHY. “0 0001” indicates that this PHY supports IEEE 802.3u CSMA/CD.

Auto Negotiation Link Partner Ability Register (ANLPAR, 0x05)

Bit	Bit Name	Reset Value	Access	Description
15	NP	0	RO	Next page indication. 1: Link partner next page enabled 0: Link partner not next page enabled
14	ACK	0	RO	Acknowledgement. 1: Link partner ability for reception of data word acknowledged 0: Not acknowledged
13	RF	0	RO	Remote fault. 1: Remote fault indicated by link partner 0: No remote fault indicated by link partner
12:11	Reserved	X	RO	Reserved. Write as 0, read as “don’t care”.
10	Pause	0	RO	Pause. 1: Pause operation supported by link partner 0: Pause operation not supported by link partner
9	T4	0	RO	100BASE-T4 support. 1: 100BASE-T4 supported by link partner 0: 100BASE-T4 not supported by link partner
8	TX_FD	0	RO	100BASE-TX full-duplex support. 1: 100BASE-TX full-duplex supported by link partner 0: 100BASE-TX full-duplex not supported by link partner

7	TX_HD	0	RO	100BASE-TX half-duplex support. 1: 100BASE-TX half-duplex supported by link partner 0: 100BASE-TX half-duplex not supported by link partner
6	10_FD	0	RO	10BASE-T full-duplex support. 1: 10BASE-T full-duplex supported by link partner 0: 10BASE-T full-duplex not supported by link partner
5	10_HD	0	RO	10BASE-T half-duplex support. 1: 10BASE-T half-duplex supported by link partner 0: 10BASE-T half-duplex not supported by link partner
4:0	Selector	0_0000	RO	Protocol selection bits. Link partner's binary encoded protocol selector.

Auto Negotiation Expansion Register (ANER, 0x06)

Bit	Bit Name	Reset Value	Access	Description
15:5	Reserved	0	0, RO	Reserved. Write as 0, read as "don't care".
4	PDF	0	0, RO / LH	Parallel detection fault. 1: Fault detected via the parallel detection function 0: No fault detected
3	LP_NP_AB	0	0, RO	Link partner next page enable. 1: Link partner next page enabled 0: Link partner not next page enabled
2	NP_AB	0	0, RO / PS	PHY next page enable. 0: PHY is not next page able.
1	Page_RX	0	0, RO / LH	New page reception. 1: New page received 0: New page not received
0	LP_AN_AB	0	0, RO	Link partner auto-negotiation enable. 1: Link partner auto-negotiation supported.

4.17 Programmable Counter Array

The Programming Counter Array (PCA) provides more timing capabilities with less CPU intervention than the standard timer/counter. Its advantages include reduced software overhead and improved accuracy. As shown in Figure 103, the PCA consists of a dedicated timer/counter, which serves as the time base for an array of 5 compare/capture modules. The PCA uses 6 I/O pins, one external clock input pin, ECI, and five bi-directional capture/compare signal pins, CEX [4:0]. Each of the five modules can be programmed in any of the following modes:

- Rising and/or falling edge capture
- Software timer
- High speed output
- Pulse Width Modulator (PWM)

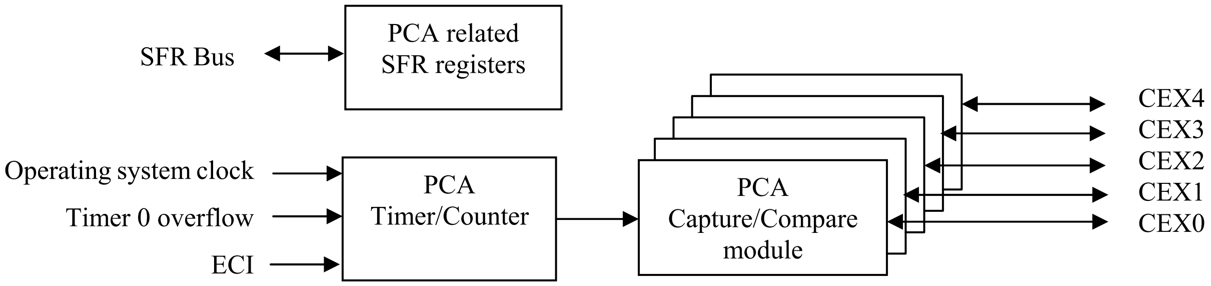


Figure 103: Programmable Counter Array Block Diagram

4.17.1 Programmable Counter Array SFR Register Map

Address	Register Name	Description
0xC2	CMOD	PCA Timer/Counter Mode Register.
0xC3	CCON	PCA Timer/Counter Control Register.
0xC4	CL	PCA Timer/Counter.
0xC5	CH	
0xD1	CCAPM0	PCA Compare/Capture Module Mode Register 0.
0xB1	CCAP0L	PCA Module 0 Compare/Capture Registers.
0xB9	CCAP0H	
0xD2	CCAPM1	PCA Compare/Capture Module Mode Register 1.
0xB2	CCAP1L	PCA Module 1 Compare/Capture Registers.
0xBA	CCAP1H	
0xD3	CCAPM2	PCA Compare/Capture Module Mode Register 2.
0xB3	CCAP2L	PCA Module 2 Compare/Capture Registers.
0xBB	CCAP2H	
0xD4	CCAPM3	PCA Compare/Capture Module Mode Register 3.
0xB4	CCAP3L	PCA Module 3 Compare/Capture Registers.
0xBC	CCAP3H	
0xD5	CCAPM4	PCA Compare/Capture Module Mode Register 4.
0xB5	CCAP4L	PCA Module 4 Compare/Capture Registers.
0xBD	CCAP4H	

Table 40: Programmable Counter Array SFR Register Map

4.17.2 PCA Timer/Counter

The PCA timer is a free-running 16-bit timer consisting of registers CH and CL (the high and low bytes of the count values). As shown in Figure 104 and Table 41, the PCA timer/counter's reference timing tick can be selected from operating system clock with 4 different divider ratios, Timer 0 overflow, and the ECI pin input. The PCA timer is the common time base for all five modules. The timer/counter source is determined from the CPS[2:0] bits in the CMOD register.

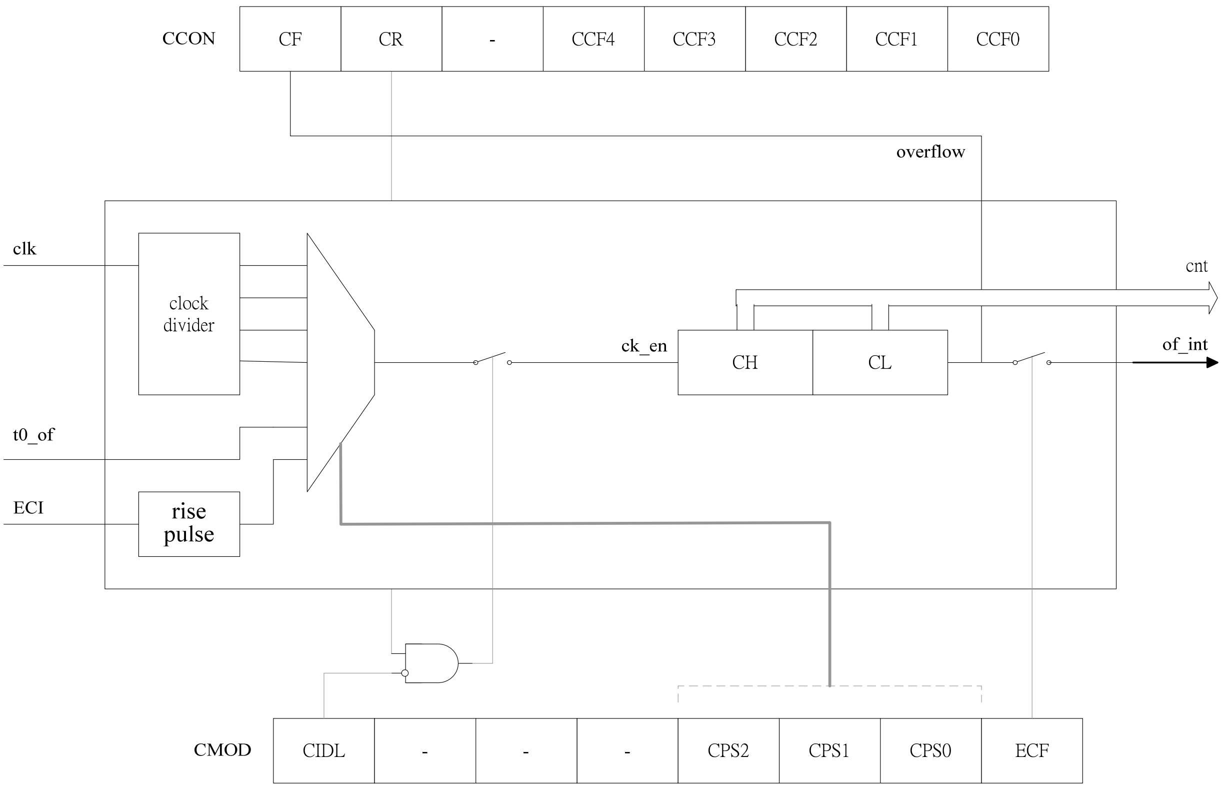


Figure 104: PCA Timer/Counter

CPS[2:0]	PCA Timer/Counter Mode		PCA Timer/Counter Reference Timing Tick		
			Operating System Clock Frequency (Fclk)		
			25MHz	50MHz	100MHz
000	Mode 0: (Fclk / 25)		1μsec	0.5μsec	0.25μsec
001	Mode 1: (Fclk / 19)		0.76μsec	0.38μsec	0.19μsec
010	Mode 2: (Fclk / 8)		0.32μsec	0.16μsec	0.08μsec
011	Mode 3: (Fclk / 6)		0.24μsec	0.12μsec	0.06μsec
100	Mode 4: Timer 0 Overflows. Timer 0 programmed to:	8-bit mode	Note 1	Note 1	Note 1
		16-bit mode	Note 1	Note 1	Note 1
		8-bit auto-reload	Note 1	Note 1	Note 1
111	Mode 5: External input pin ECI (The max input frequency should be less than Fclk / 2)		> 80ns	> 40ns	> 20ns

Note:

1. The reference timing tick is determined by the Timer 0 overflow rate programmed by software. In Mode 4, the overflow interrupt for Timer 0 does not need to be enabled.

Table 41: PCA Timer/Counter Input Sources and Reference Timing Tick

4.17.3 Compare/Capture Modules

Each PCA module has a mode register with it. These registers are: CCAPM0 for module 0, CCAPM1 for module 1, etc. Each register contains 7 bits that are used to control the mode in which each module will operate.

Capture Mode

Capture mode is used to capture the PCA timer/counter value into a module's capture registers (CCAPnH and CCAPnL). As shown in Figure 105 below, the capture will occur on a positive edge, negative edge, or both on the corresponding module's CEX[n] pin. To use one of the PCA modules in the capture mode, either one or both the CCAPM register bits CAPN and CAPP for that module must be set. When a valid transition occurs on the CEX[n] pin corresponding to the module used, the PCA hardware loads the 16-bit value of the PCA counter register (CH and CL) into the module's capture registers (CCAPnL and CCAPnH).

The CCFn bit for the module in the CCON register is set by hardware. The value of trigger event CEXn after last trigger is reflected is CEXn bit of CCAPMn register. If the ECCFn bit in the CCAPMn register is set, then an interrupt on INT3 will be generated. In the interrupt service routine, the 16-bit capture value must be saved in memory before the next event capture occurs. If a subsequent capture occurred, the original capture values would be lost. After the event flag (CCFn) has been set by hardware, the user must clear the flag in software.

A common use for the PCA capture mode is to measure the properties of a waveform. Properties such as the period, pulse width or the phase difference of two waveforms are measured by determining the difference in capture values between two edges of the waveform. The hardware support of the PCA capture mode allows accurate measurement of these properties with low software overhead.

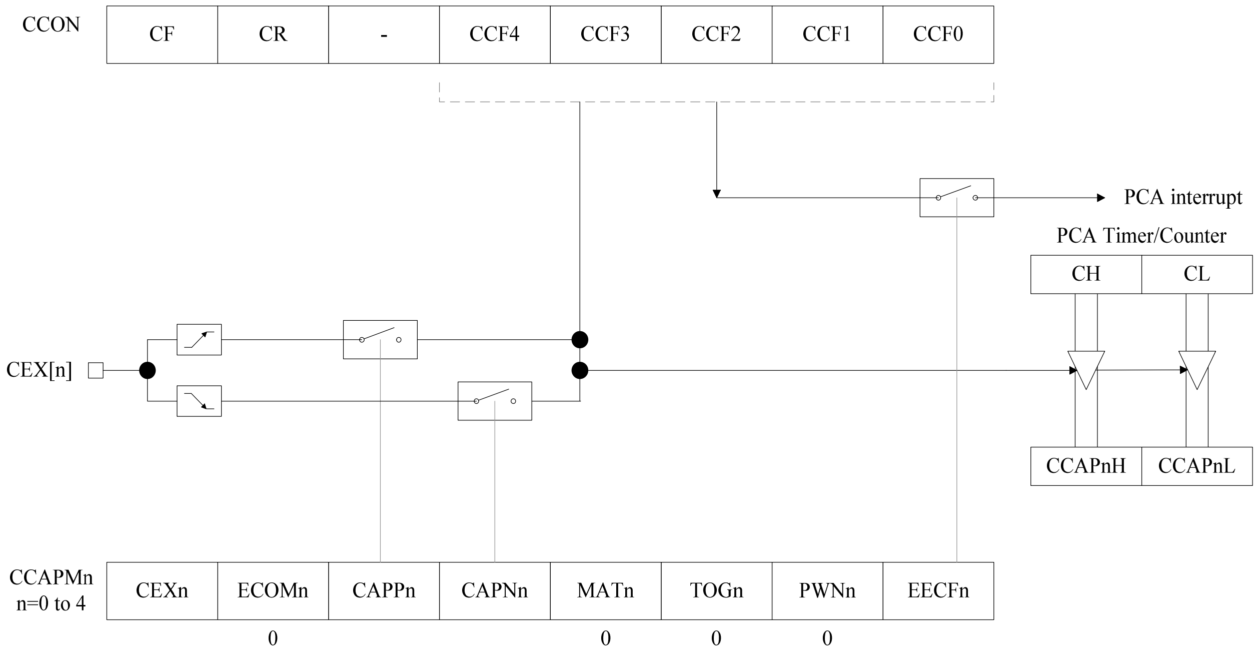


Figure 105: PCA Capture Mode

16-bit Software Timer Mode

The 16-bit software timer mode is used to trigger interrupt routines, which must occur at periodic intervals. As shown in Figure 106, it is setup by setting both the ECOM and MAT bits in the module's CCAPMn register. The PCA timer will be compared to the module's capture registers (CCAPnL and CCAPnH) and when a match occurs, an interrupt on INT3 will occur, if the ECCFn bit in CCAPMn register for the module is set.

If necessary, a new 16-bit compare value can be loaded into CCAPnH and CCAPnL during the interrupt routine. The user should be aware that the hardware temporarily disables the comparator function while these registers are being updated so that an invalid match will not occur. Thus, it is recommended that the user write to the low byte first (CCAPnL) to disable the comparator, then write to the high byte (CCAPnH) to re-enable it. If any updates to the registers are done, the user may want to hold off any interrupts from occurring by clearing the ECCFn bit or the EA bit.

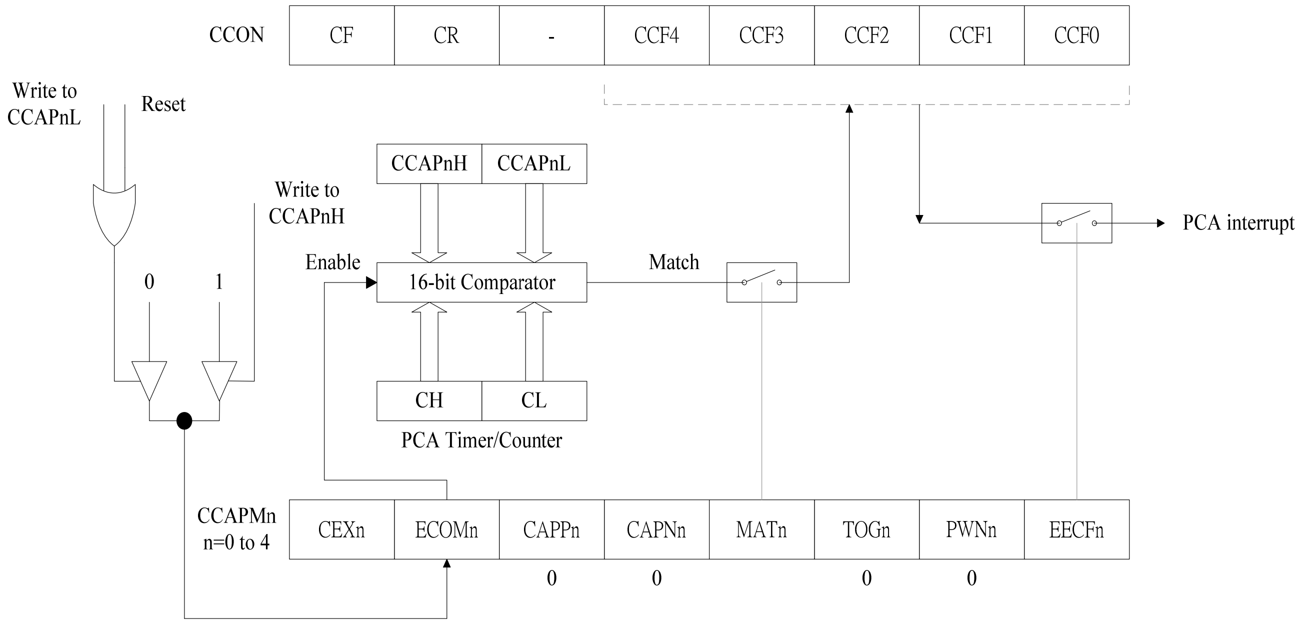


Figure 106: PCA Software Timer Mode (Compare Mode)

High Speed Output Mode

In this mode, the CEX[n] output pin associated with the PCA module will toggle every time there is a match between the PCA counter (CH and CL) and the capture registers (CCAPnH and CCAPnL). As shown in

Figure 107 below, to activate this mode, the user must set TOG, MAT, and ECOM bits in the module's CCAPMn register. High Speed Output mode is much more accurate than software pin toggling since the toggle occurs before branching to an interrupt. In this case, interrupt latency will not affect the accuracy of the output.

When using High Speed Output mode, using an interrupt is optional. Only if the user wishes to change the time for the next toggle is it necessary to update the compare registers. Otherwise, the next toggle will occur when the PCA timer rolls over and matches the last compare value.

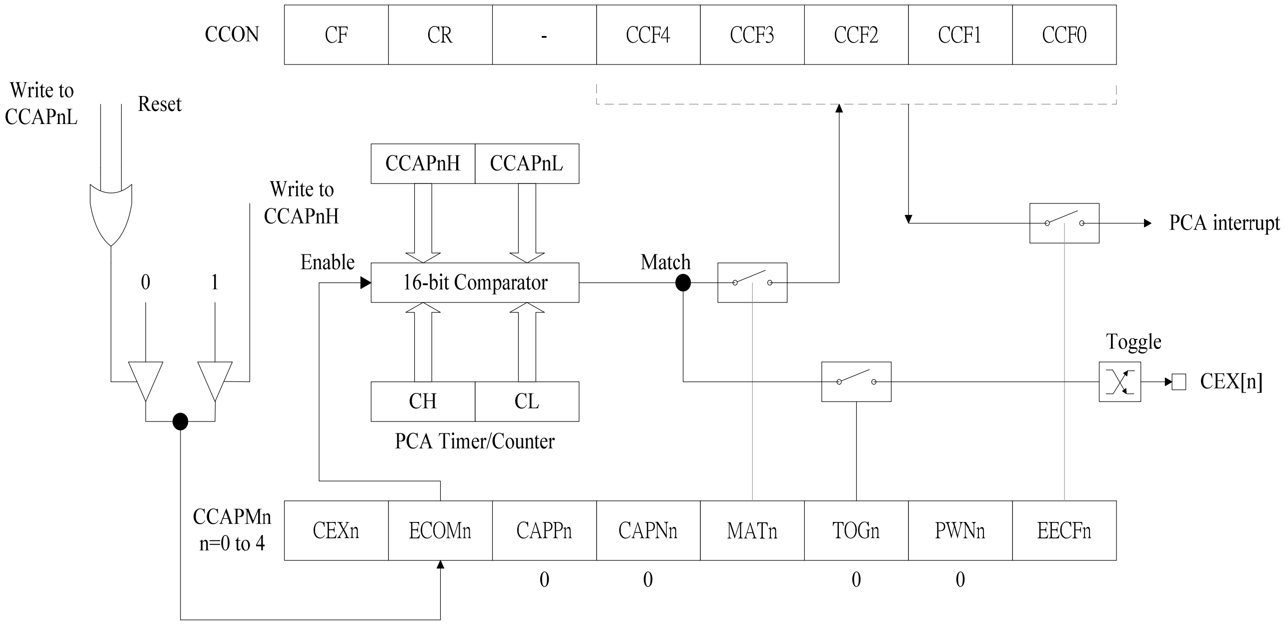


Figure 107: PCA High-Speed Output Mode

Pulse Width Modulator

The Pulse Width Modulator (PWM) mode is used to generate a continuous square-wave with an arbitrary duty cycle. As shown in Figure 108 below, it generates 8-bit PWM by comparing the low byte of the PCA timer (CL) with the low byte of the compare register (CCAPnL). When CL < CCAPnL the output of CEX[n] is low. When CL >= CCAPnL the output CEX[n] is high. To activate this mode, the user must set the PWM and ECOM bits in the module's CCAPMn register.

In PWM mode, the frequency of the output depends on the source for the PCA timer. See Table 42 below. Since there is only one set of CH and CL registers, all modules share the PCA timer and frequency. The frequency is fixed to 256 counts of the PCA timer. Duty cycle of the output is controlled by the value loaded into the high byte (CCAPnH). Since writes to the CCAPnH register are asynchronous, a new value written to the high byte will not be shifted into CCAPnL for comparison until the next period of the output (when CL rolls over from 255 to 00).

To calculate values for CCAPnH for any duty cycle, use the following equation:

CCAPnH = 256 * (1 - Duty Cycle)

Where CCAPnH is an 8-bit integer and duty cycle is a fraction.

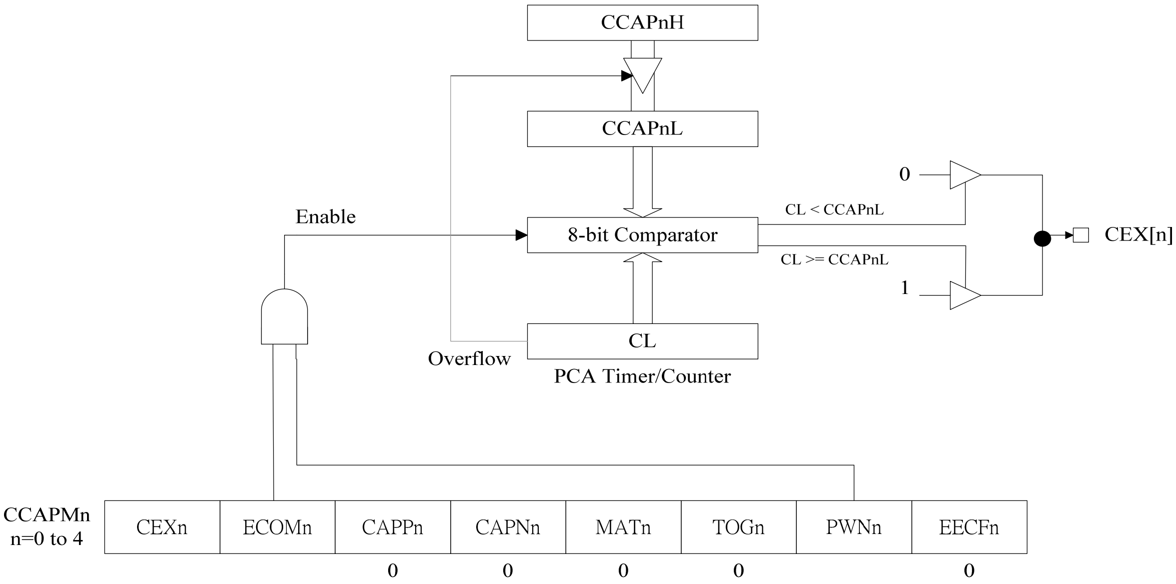


Figure 108: PCA Pulse Width Modulator Mode

PCA Timer Mode		Max. PWM Frequency Operating System Clock (Fclk)		
		25MHz	50MHz	100MHz
Mode 0: (Fclk / 25), Note 1		3.91 KHz	7.81 KHz	15.63 KHz
Mode 1: (Fclk / 19), Note 1		5.14 KHz	10.28 KHz	20.56 KHz
Mode 2: (Fclk / 8), Note 1		12.22 KHz	24.41 KHz	48.83 KHz
Mode 3: (Fclk / 6), Note 1		16.28 KHz	32.55 KHz	65.1 KHz
Mode 4: Timer 0 Overflow,	8-bit	Note 2	Note 2	Note 2
	16-bit	Note 2	Note 2	Note 2
	8-bit Auto-Reload	Note 2	Note 2	Note 2
Mode 5: External input pin ECI (The max input frequency should be less than Fclk / 2)		< 48.8 KHz, Note 3	< 97.7 KHz, Note 3	< 195.3 KHz, Note 3

Note:
1. Max. PWM frequency = (Fclk / Divider) / 256, where Divider is 25, 19, 8, 6 for Mode 0, 1, 2, 3, respectively.
2. Max. PWM frequency = Timer 0 overflow rate / 256.
3. Max. PWM frequency = ECI frequency / 256.

Table 42: Pulse Width Modulator Frequency

PCA Timer/Counter Mode Register (CMOD, 0xC2)

Bit	7	6	5	4	3	2	1	0
Name	CIDL	Reserved			CPS			ECF
Reset Value	0	00			00			0

<table>
<tr><th>Bit</th><th>Name</th><th>Access</th><th>Description</th></tr>
<tr><td>0</td><td>ECF</td><td>R/W</td><td>PCA Enable Counter Overflow interrupt.
1: Enables CF bit in CCON to generate an interrupt.
0: Disables the CF bit in CCON.</td></tr>
<tr><td>3:1</td><td>CPS</td><td>R/W</td><td>PCA Count Pulse Select.
<table>
<tr><th>CPS</th><th>Description</th></tr>
<tr><td>000</td><td>Reference timing tick = operating system clock divided by 25.</td></tr>
<tr><td>001</td><td>Reference timing tick = operating system clock divided by 19.</td></tr>
<tr><td>010</td><td>Reference timing tick = operating system clock divided by 8.</td></tr>
<tr><td>011</td><td>Reference timing tick = operating system clock divided by 6.</td></tr>
<tr><td>100</td><td>Reference timing tick = Timer 0 overflow rate.</td></tr>
<tr><td>111</td><td>Reference timing tick = external clock at ECI pin (the max input frequency should be less than operating system clock divided by 2)</td></tr>
</table>
For more details, please refer to Table 41.</td></tr>
<tr><td>6:4</td><td>Reserved</td><td>RO</td><td></td></tr>
<tr><td>7</td><td>CIDL</td><td>R/W</td><td>Counter Idle Control.
1: Program the PCA Counter to be gated off during idle.
0: Program the PCA Counter to continue functioning during idle mode.</td></tr>
</table>

PCA Timer/Counter Control Register (CCON, 0xC3)

Bit	7	6	5	4	3	2	1	0
Name	CF	CR	Res.	CCF4	CCF3	CCF2	CCF1	CCF0
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	CCF0	CR	PCA Module 0 interrupt flag. Set by hardware when a match or capture occurs.
1	CCF1	CR	PCA Module 1 interrupt flag. Set by hardware when a match or capture occurs.
2	CCF2	CR	PCA Module 2 interrupt flag. Set by hardware when a match or capture occurs.
3	CCF3	CR	PCA Module 3 interrupt flag. Set by hardware when a match or capture occurs.
4	CCF4	CR	PCA Module 4 interrupt flag. Set by hardware when a match or capture occurs.
5	Reserved	RO	
6	CR	R/W	PCA Counter Run control bit. Set by software to turn the PCA counter on. Must be cleared by software to turn the PCA counter off.
7	CF	CR	PCA Counter Overflow Flag. Set by hardware when the counter rolls over. CF flags an interrupt to INT3 if bit ECF in CMOD is set.

The CCON register shown above is associated with all PCA timer functions. It contains the run control bit (CR) and overflow flags for the PCA timer (CF) and all modules (CCFx). To run the PCA the CR bit (CCON.6) must be set by software. Clearing the bit will turn off PCA.

When the PCA counter overflows, the CF (CCON.7) will be set, and an interrupt will be generated if the ECF bit (CMOD.0) is set. The CF bit can only be cleared by software.

Each module has its own timer overflow flag or capture flag (CCF0 for module 0, CCF4 for module 4, etc.). They are set when either a match or capture occurs. These flags can be cleared by software read.

PCA Timer/Counter (CL/CH)

CL, 0xC4

Bit	7	6	5	4	3	2	1	0
Name	PTC[7:0]							
Reset Value	00							

CH, 0xC5

Bit	7	6	5	4	3	2	1	0
Name	PTC[15:8]							
Reset Value	00							

Bit	**Name**	**Access**	**Description**
15:0	PTC	RO	PCA Timer/Counter.

PCA Compare/Capture Module Mode Register (CCAPMn)

CCAPM0, 0xD1

Bit	7	6	5	4	3	2	1	0
Name	CEX0	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0
Reset Value	0	0	0	0	0	0	0	0

CCAPM1, 0xD2

Bit	7	6	5	4	3	2	1	0
Name	CEX1	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1
Reset Value	0	0	0	0	0	0	0	0

CCAPM2, 0xD3

Bit	7	6	5	4	3	2	1	0
Name	CEX2	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2
Reset Value	0	0	0	0	0	0	0	0

CCAPM3, 0xD4

Bit	7	6	5	4	3	2	1	0
Name	CEX3	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3
Reset Value	0	0	0	0	0	0	0	0

CCAPM4, 0xD5

Bit	7	6	5	4	3	2	1	0
Name	CEX4	ECOM4	CAPP4	CAPN4	MAT4	TOG4	PWM4	ECCF4
Reset Value	0	0	0	0	0	0	0	0

Bit	**Name**	**Access**	**Description**
0	ECCFn	R/W	Enable CCF Interrupt. 1: Enable compare/capture flag CCF[4:0] in the CCON register to generate an interrupt. 0: Disable compare/capture flag CCF[4:0] in the CCON register to generate an interrupt.
1	PWMn	R/W	Pulse Width Modulation mode. 1: Enable CEX[n] pin to be used as a pulse width modulated output.

			0: Disable PWM mode.
2	TOGn	R/W	Toggle. 1: A match of the PCA counter with this module's compare/capture register causes the CEX[n] pin to toggle. 0: Disable toggle function.
3	MATn	R/W	Match: Set ECOM[4:0] and MAT[4:0] to implement the software timer mode. 1: A match of the PCA counter with this module's compare/capture register causes the CCFn bit in CCON to be set, flagging an interrupt. 0: Disable software timer mode.
4	CAPNn	R/W	Capture Negative. 1: Enable negative edge capture on CEX[4:0]. 0: Disable negative edge capture on CEX[4:0].
5	CAPPn	R/W	Capture Positive. 1: Enable positive edge capture on CEX[4:0]. 0: Disable positive edge capture on CEX[4:0].
6	ECOMn	R/W	Enable Comparator. 1: Enable the comparator function. 0: Disable the comparator function.
7	CEXn	RO	Capture/Compare External In for PCA Module n. 1: After last trigger event the CEX[n] is 1. 0: After last trigger event the CEX[n] is 0.

The ECCFn bit (CCAPMn bit 0 where n = 0, 1, 2, 3, or 4 depending on module) will enable the CCFn flag in the CCON register to generate an interrupt when a match or compare occurs. PWM bit (CCAPMn.1) enables the pulse width modulation mode. The MATn bit (CCAPMn.3) when set will cause the CCFn bit in the CCON register to be set when there is a match between the PCA counter and the module's capture/compare registers. Additionally, the TOG bit (CCAPMn.2) when set, causes the CEX[n] output pin associated with that module to toggle when there is a match between the PCA counter and the module's capture/compare registers.

The CAPNn bit (CCAPMn.4) and CAPPn (CCAPMn.5) determine whether the capture input will be active on a positive edge or negative edge. The CAPNn bit enables a capture at the negative edge, and the CAPPn bit enables a capture at the positive edge. When both bits are set, both edges will be enabled and a capture will occur for either transition to measure pulse width. The ECOMn bit (CCAPMn.6) when set enables the comparator function.

CEXn bit (CCAPMn.7) shows after last trigger event the CEX[n] value in capture mode to determine whether it's positive or negative edge.

Table 43 and Table 44 show the CCAPMn settings for the various PCA functions.

Module Mode	**ECOMn**	**CAPPn**	**CAPNn**	**MATn**	**TOGn**	**PWMn**	**ECCFn**
No Operation	0	0	0	0	0	0	0
16-bit capture on positive-edge trigger at CEX[4:0]	0	1	0	0	0	0	0
16-bit capture on negative-edge trigger at CEX[4:0]	0	0	1	0	0	0	0
16-bit capture on positive/negative-edge trigger at CEX[4:0]	0	1	1	0	0	0	0
Compare: software timer	1	0	0	1	0	0	0
Compare: high-speed output	1	0	0	1	1	0	0
Compare: 8-bit PWM	1	0	0	0	0	1	0

Note: n = 0, 1, 2, 3, 4

Table 43: PCA Module Modes Without Interrupt Enabled

Module Mode	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn
16-bit capture on positive-edge trigger at CEX[4:0]	0	1	0	0	0	0	1
16-bit capture on negative-edge trigger at CEX[4:0]	0	0	1	0	0	0	1
16-bit capture on positive/negative-edge trigger at CEX[4:0]	0	1	1	0	0	0	1
Compare: software timer	1	0	0	1	0	0	1
Compare: high-speed output	1	0	0	1	1	0	1
Compare: 8-bit PWM	1	0	0	0	0	1	X

Note:

1. n = 0, 1, 2, 3, 4
2. No PCA interrupt is needed to generate the PWM.

Table 44: PCA Module Modes With Interrupt Enabled

PCA Module n Compare/Capture Registers (CCAPnL/CCAPnH)

There are two additional registers associated with each of the PCA modules: CCAPnH and CCAPnL. They are registers that hold the 16-bit count value when a capture occurs or a comparison occurs. When a module is used in PWM mode, these registers are used to control the duty cycle of the output.

CCAP0L, 0xB1

Bit	7	6	5	4	3	2	1	0
Name	CCAP0[7:0]							
Reset Value	00							

CCAP0H, 0xB9

Bit	7	6	5	4	3	2	1	0
Name	CCAP0[15:8]							
Reset Value	00							

CCAP1L, 0xB2

Bit	7	6	5	4	3	2	1	0
Name	CCAP1[7:0]							
Reset Value	00							

CCAP1H, 0xBA

Bit	7	6	5	4	3	2	1	0
Name	CCAP1[15:8]							
Reset Value	00							

CCAP2L, 0xB3

Bit	7	6	5	4	3	2	1	0
Name	CCAP2[7:0]							
Reset Value	00							

CCAP2H, 0xBB

Bit	7	6	5	4	3	2	1	0
Name	CCAP2[15:8]							
Reset Value	00							

CCAP3L, 0xB4

Bit	7	6	5	4	3	2	1	0
Name	CCAP3[7:0]							
Reset Value	00							

CCAP3H, 0xBC

Bit	7	6	5	4	3	2	1	0
Name	CCAP3[15:8]							
Reset Value	00							

CCAP4L, 0xB5

Bit	7	6	5	4	3	2	1	0
Name	CCAP4[7:0]							
Reset Value	00							

CCAP4H, 0xBD

Bit	7	6	5	4	3	2	1	0
Name	CCAP4[15:8]							
Reset Value	00							

Bit	Name	Access	Description
15:0	CCAPn	RO	PCA Module n Compare/Capture Registers

4.18 I2C Controller

The I2C controller of AX11025 supports Standard-mode (100K bps) and Fast-mode (400K bps), but not High-speed mode (3.4M bps) of the standard I2C bus spec. As shown in Figure 109, the I2C controller consists of an I2C master controller to support communication to external I2C devices, an I2C slave controller to support communication to external micro-controller with I2C master, and an I2C boot loader to support communication to external I2C EEPROM being used for storing chip configuration data.

The I2C master controller is compatible with I2C bus protocol. It provides eight registers to fully control and monitor I2C bus transaction, and it has separate receive and transmit registers to hold data for transactions between AX11025 and the external I2C devices. The I2C master controller also provides arbitration for multi-master operation scenario and reports the arbitration status. Also, the I2C master controller accepts the SCL being extended low by the slow I2C slave devices as additional wait state indication during data or acknowledge cycles. The I2C clock frequency is software programmable.

The I2C slave controller allows an external micro-controller with I2C master to communicate with AX11025. It provides an I2C device ID register to allow flexible assignment of AX11025 with any I2C device address for either 7-bit or 10-bit address mode, and can automatically filter I2C bus transactions not belonging to AX11025 in hardware. The I2C slave controller can extend the SCL line low when it needs additional wait state to respond to the external I2C master's bus transaction. The I2C slave controller supports 6 flexible command instructions for the external micro-controller to access the internal registers and memory resources of AX11025.

The I2C boot loader is used to load chip configuration data from external I2C EEPROM. It is activated after hardware reset (either power-on-reset or RST_N input) or via the software reload command (via I2CCTR register). The detailed memory map of I2C EEPROM is described in section 3.1. The use of external I2C EEPROM is optional, when not used, the I2C_BOOT_DIS pin should be pulled up during chip hardware reset, in that case, the reset value listed in I2C EEPROM memory map shall be used by this chip by default.

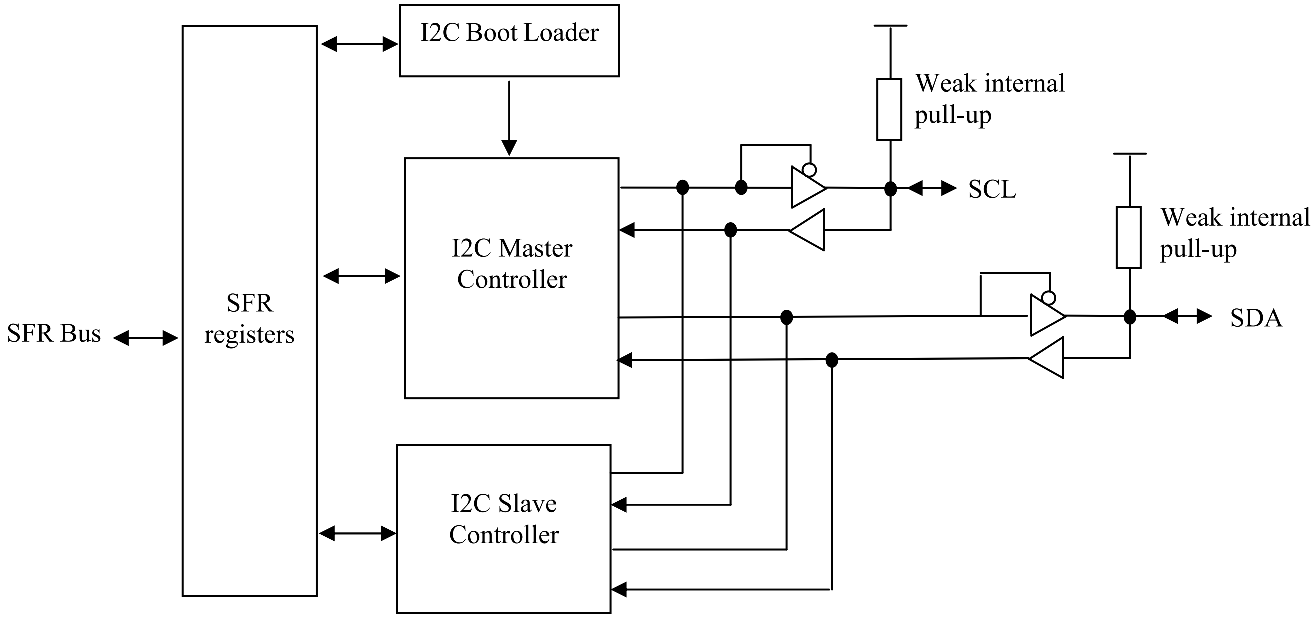


Figure 109: I2C Controller Block Diagram

4.18.1 I2C Controller SFR Register Map

Address	Name	Description
0x96	I2CCIR	I2C Command Index Register is used to indicate the address of I2C controller register.
0x97	I2CDR	I2C Data Register is used to read data from or write data to the specified I2C controller register.

Table 45: I2C Controller SFR Register Map

I2C Command Index Register (I2CCIR, 0x96)

Bit	7	6	5	4	3	2	1	0
Name	I2CCIR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	I2CCIR	WO	Indicate which of the I2C controller register as listed in Table 46 is to be accessed.

I2C Data Register (I2CDR, 0x97)

Bit	7	6	5	4	3	2	1	0
Name	I2CDR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	I2CDR	R/W	Data Register is used to write data to or read data from the I2C controller registers.

I2C Controller Register Indirect Access Method

Software shall use indirect access method through I2CCIR and I2CDR registers to do read and write access to the I2C controller registers as listed in Table 46 below.

Read a register from I2C controller:

Step 1. Write I2CCIR: Software indicates the I2C controller register address to be accessed as the data and write it to the SFR register I2CCIR.

Step 2. Read I2CDR: Software then read SFR register I2CDR. The data read from I2CDR is the I2C controller register data indicated in step 1. Keep reading from I2CDR if the I2C controller registers have more than one byte, in that case, the first byte being read back is LSB byte.

Write a register to I2C controller:

Step 1. Write I2CDR: Software writes the data you want to write into I2C controller registers to the SFR register I2CDR. Keep writing to I2CDR if the I2C controller registers have more than one byte, in that case, the first byte being written should be LSB byte.

Step 2. Write I2CCIR: After writing I2C controller register data to I2CDR, software then indicates the target I2C controller register address as data and write it to I2CCIR.

Note: While software is reading or writing I2C controller registers during a sequence of SFR accesses, software can abort that process by writing I2CCIR with 0xFF.

I2C Controller Register Map

Address	Name	Description
0x00	I2CCPR	I2C Clock Pre-scale Register.
0x02	I2CTR	I2C Transmit Register is used to transmit data to device.
0x03	I2CRR	I2C Receive Register is used to hold data that receive from device.
0x04	I2CCTR	I2C Control Register is used to configure operation mode.
0x05	I2CCR	I2C Command Register is used to configure operation mode.
0x06	I2CMSR	I2C Master Status Register is used to report status of the I2C master mode.
0x07	Reserved	
0x08	I2CSDA	I2C Slave Device Address Register
0x0A	I2CSSR	I2C Slave Status Register is used to report status of the I2C slave mode.

Table 46: I2C Controller Register Map

I2C Clock Pre-scale Register (I2CCPR, 0x00)

Bit	7	6	5	4	3	2	1	0
Name	PRER0							
	PRER1							
Reset Value	0xFFFF							

Bit	Name	Access	Description
7:0 15:8	PRER0 PRER1	R/W	This register is used to pre-scale the SCL clock line. I2C SCL clock frequency = Operating system clock frequency / (5* (PRER + 1))

Operating System Clock	PRER	
25 Mhz	0x00_0c	Fast mode
50 Mhz	0x00_18	
75 Mhz	0x00_25	
100 Mhz	0x00_31	
25 Mhz	0x00_3a	Standard mode
50 Mhz	0x00_76	
75 Mhz	0x00_B1	
100 Mhz	0x00_C7	

I2C Transmit Register (I2CTR, 0x02)

Bit	7	6	5	4	3	2	1	0
Name	TXD							RW_TXD
Reset Value	000_0000							0

Bit	Name	Access	Description
0	RW_TXD	R/W	In case of a slave address transfer this bit represents the RW bit, where 1: Reading from slave. 0: Writing to slave. In case of a data transfer this bit represents the data's LSB bit.
7:1	TXD	R/W	Next byte to transmit on I^2C bus in either master or slave mode.

I2C Receive Register (I2CRR, 0x03)

Bit	7	6	5	4	3	2	1	0
Name	RXD							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	RXD	RO	Last byte received from I^2C bus in either master or slave mode.

I2C Control Register (I2CCTR, 0x04)

Bit	7	6	5	4	3	2	1	0
Name	MSS	SIE	Reserved	RLE	TE	SD	EN	MIE
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	MIE	R/W	I2C Master mode Interrupt Enable. This bit is only valid when operating in I2C master mode. 1: Interrupt enable. 0: Interrupt disable.
1	EN	R/W	I2C controller Enable. 1: Enable I2C controller. 0: Disable I2C controller.
2	SD	R/W	I2C Speed in slave mode. This bit is only valid when operating in I2C slave mode. 1: I2C slave mode operating in STANDARD mode. 0: I2C slave mode operating in FAST mode.
3	TE	R/W	Ten address Enable. This bit is only valid when operating in I2C slave mode. 1: 10-bit address enable. 0: 7-bit address enable.
4	RLE	R/W1	Re-load I2C Configuration EEPROM. 1: To reload the external I2C EEPROM's content. The I2C Boot Loader will automatically re-load the content of I2C EEPROM into the chip. This bit is cleared by hardware. The status of reload progress is reported in I2CMSR. 0: Normal operation mode.
5	Reserved	-	
6	SIE	R/W	I2C Slave mode Interrupt Enable. This bit is only valid when operating in I2C slave mode. 1: Interrupt enable. 0: Interrupt disable.
7	MSS	R/W	I2C Master/Slave mode Select. 1: Set I2C controller to operate as I2C master. 0: Set I2C controller to operate as I2C slave.

I2C Command Register (I2CCR, 0x05)

Bit	7	6	5	4	3	2	1	0
Name	STA	STO	RD	WR	MG	Reserved	SG	RLS
Reset Value	0	0	0	0	0	0	0	0

Bit	**Name**	**Access**	**Description**
0	RLS	R/W1	Release the SCL pin. This bit is only valid when operating in I2C slave mode. When the external I2C master controller is reading data from this chip, after each byte being transferred, normally the SCL line will be held low by the I2C slave controller of this chip so that the software can prepare for the next 8-bit data byte in I2CTR. This forces the external I2C master controller to wait for this chip. However, in the case of finishing the transfer of the last bit of the last byte during the external I2C master read command, software shall write 1 to this bit to release SCL pin of this chip to allow the STOP condition on I2C bus.
1	SG	R/W1	Slave Go. This bit is only valid when operating in I2C slave mode. Writing 1 to this bit starts the transfer in I2C slave mode. This bit remains set during the transfer and is automatically cleared after the transfer finished.
2	Reserved	-	
3	MG	R/W1	Master Go. This bit is only valid when operating in I2C master mode. Writing 1 to this bit starts the transfer in I2C master mode. This bit remains set during the transfer and is automatically cleared after the transfer finished.
4	WR	R/W1	When operating in I2C master mode, setting ‘1’ to request to send the 8-bit data in I2CTR to the slave. This bit is only valid when operating in I2C master mode.
5	RD	R/W1	When operating in I2C master mode, setting ‘1’ to request to receive data from slave. The received data is stored in I2CRR. Setting RD bit and STO bit at the same time will cause the transfer to end with a NACK condition to the addressed slave. Setting RD bit without setting STO bit will cause the transfer to end with an ACK condition to the addressed slave. This bit is only valid when operating in I2C master mode.
6	STO	R/W1	When operating in I2C master mode, setting ‘1’ to request to generate the STOP condition on I2C bus. This bit is only valid when operating in I2C master mode.
7	STA	R/W1	When operating in I2C master mode, setting ‘1’ to request to generate the START or ReSTART condition on I2C bus. This bit is only valid when operating in I2C master mode.

I2C Master Status Register (I2CMSR, 0x06)

Bit	7	6	5	4	3	2	1	0
Name	ACK	BUSY	AL	BLD	RLES	Reserved	TIP	IF
Reset Value	0	0	0	1	0	0	0	0

Bit	**Name**	**Access**	**Description**
0	IF	CR	Interrupt Flag. This bit is set when following events occur, ● One byte transfer has been completed ● Arbitration is lost This will cause an interrupt request on INT4 if the MIE bit (I2CCTR.0) is set.
1	TIP	RO	Transfer in progress. 1: When transferring data is in progress. 0: When transfer is completed.
2	Reserved	-	
3	RLES	RO	Reload EEPROM Status. 1: After software sets the RLE bit (I2CCTR.4) to ‘1’, the I2C Boot Loader will reload the I2C EEPROM content and keeps this bit to ‘1’ until the reload is completed. 0: The chip hardware will clear this bit after it completes the reload process.
4	BLD	RO	Boot Loader Done. 1: I2C Boot Loader has done with loading.

			0: I2C Boot Loader is still loading configuration data from I2C Configuration EEPROM.
5	AL	CR	Arbitration Lost. This bit is set when the I2C master lose arbitration during multi-master scenario. Arbitration is lost when: ● A STOP signal is detected, but non requested ● The master drives SDA high, but SDA is low.
6	BUSY	RO	I2C bus Busy. 1: After the START signal is detected on I2C bus, the I2C bus is busy. 0: After the STOP signal is detected on the I2C bus, the I2C bus is not busy.
7	ACK	CR	This flag represents the Acknowledgement received from I2C slave after a transmit transfer (8-bit data being sent to the external I2C device). This bit is only meaningful after the TIP bit changes from '1' to '0' for a transfer. 1: NACK is received from the slave. 0: ACK is received from the slave.

I2C Slave Device Address (I2CSDA, 0x08)

Bit	7	6	5	4	3	2	1	0
Name	DA0							
	DA1							
Reset Value	0x00							

Bit	Name	Access	Description
7:0 9:8	DA0 DA1	R/W	Device Address of this chip operating in I2C slave mode. The {DA1, DA0} is the I2C device address of this chip operating in I2C slave mode. If the device is configured as 7-bits address mode then only bit [6:0] are valid. The 6th bit is MSB. If the device is configured as 10-bits address mode then bit [9:0] are valid. The 9th bit is MSB.
15:10	Reserved	-	

I2C Slave Status Register (I2CSSR, 0x0a)

Bit	7	6	5	4	3	2	1	0
Name	ERR	STOP	START	RE-START	RD	WR	NACK	STC
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	STC	CR	Slave Transfer Complete. Reading '1' indicates that the external I2C master has just completed one transfer on I2C bus.
1	NACK	CR	NACK condition. Reading '1' indicates that the external I2C master returns a NACK condition during current transfer.
2	WR	RO	Write command. Reading '1' indicates that the external I2C master needs to transmit data to this chip. The data is held in I2CRR register.
3	RD	RO	Read command. Reading '1' indicates that the external I2C master needs to receive data from this chip. After knowing this, the software shall put the requested data in I2CTR register.
4	RE-START	CR	ReSTART condition detected. Reading '1' indicates that the ReSTART condition is detected on the I2C bus.
5	START	CR	START condition detected. Reading '1' indicates that the SART condition is detected on the I2C bus.
6	STOP	CR	STOP condition detected. Reading '1' indicates that the STOP condition is detected on the I2C bus.
7	ERR	CR	Error. Reading '1' indicates that the I2C slave controller of this chip has detected an error on SCL and aborted the current transfer.

Example Programming Procedure in I2C Master Mode

Example 1: Write 1 byte of data = 0xAC to an external slave device with slave address = 0x51 (101_0001).

1. Write 0xA2 (slave address) to I2CTR. Set STA, WR, and MG bits to I2CCR.
2. Read TIP and ACK bits from I2CMSR until both read as ‘0’ (polling mode or wait for interrupt in interrupt mode).
3. Write 0xAC to I2CTR. Set STO, WR, and MG bits to I2CCR.
4. Read TIP and ACK bits from I2CMSR until both read as ‘0’.

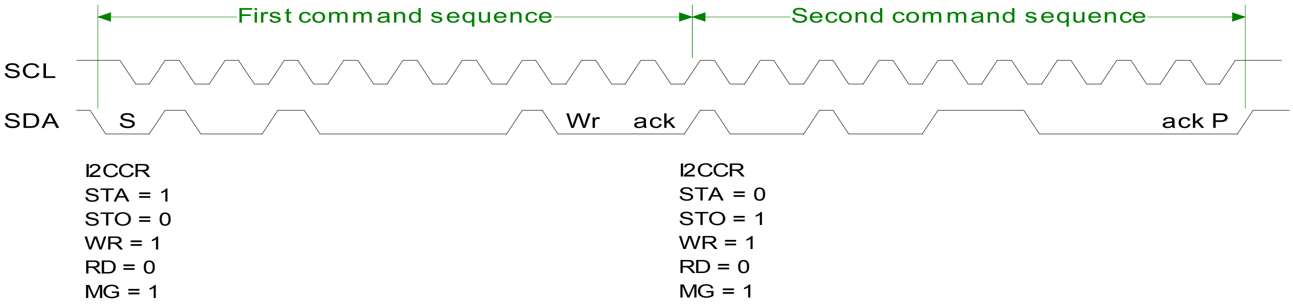


Figure 110: Transmitting Data to an I2C Slave Device

Example 2: Read a byte of data from location 0x20 of an I2C memory device with slave address = 0x4E (100_1110)

1. Write 0x9C (slave address) to I2CTR. Set STA, WR, and MG bits to I2CCR. Read TIP and ACK bits from I2CMSR until both read as ‘0’.
2. Write 0x20 to I2CTR. Set WR and MG bits to I2CCR. Read TIP and ACK bits from I2CMSR until both read as ‘0’.
3. Write 0x9D (slave address) to I2CTR. Set STA, WR, and MG bits to I2CCR. Read TIP and ACK bits from I2CMSR until both read as ‘0’.
4. Set RD, STO and MG bits to I2CCR. Read TIP and IF bits from I2CMSR until both read as ‘0’.

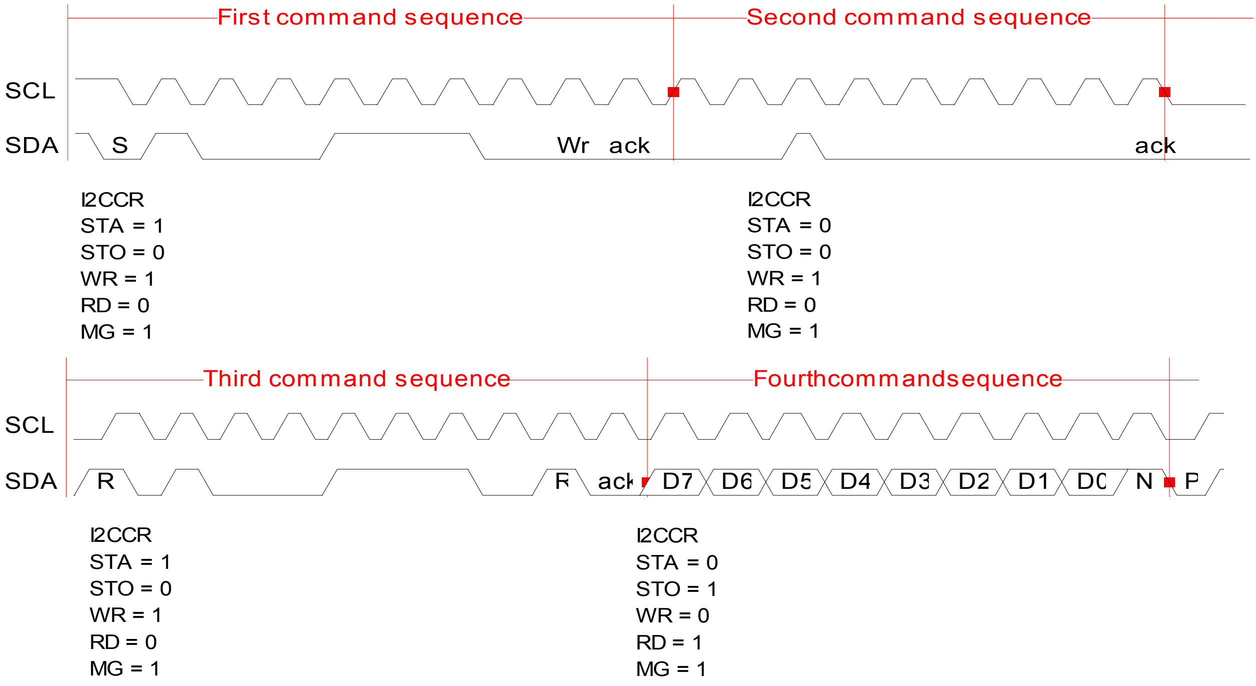


Figure 111: I2C Read Data

4.18.2 I2C Slave Mode Function Description

The I2C slave controller of this chip provides 6 reference command instructions, namely, SW_SFR, SR_SFR, IW_SFR, IR_SFR, BWDM, and BRDM commands, for the external I2C master to communicate with this chip. The external I2C master may use these commands to read or write the SFR registers or xDATA memory of this chip. Except for the I2C device address transfer that the I2C slave controller of this chip will be parsing, the remaining byte transfers are pretty much parsing by the software of AX11025, therefore, these commands are allowed to make changes to meet user's applications.

The SW_SFR, SR_SFR, IW_SFR, and IR_SFR are single read or write commands, while BWDM and BRDM are burst read or write commands in one transfer with address automatically incremented.

Command Name	Op-code	Operation Description
SW_SFR	1010 0xxx (0xA0~0xA7)	Single Write SFR register. This command requests to write various bytes of data to the specified SFR register in this chip. The xxx indicates the number of bytes to be written to target register. For example, xxx = 000 for 1 byte, and xxx = 111 for 8 bytes.
SR_SFR	0010 0xxx (0x20~0x27)	Single Read SFR register. This command requests to read various bytes of data from the specified SFR register from this chip. The xxx indicates the number of bytes to be read from the target register. For example xxx = 000 for 1 byte, and xxx = 111 for 8 bytes.
IW_SFR	1011 xxxx (0xB0~0xBF)	Indirect Write SFR register. This command requests to indirectly write various bytes of data through the specified command index register in SFR to the given indirect register in this chip. The xxxx indicates the number of bytes to be written to target indirect register. For example, xxxx = 0000 for 1 byte, and xxxx = 1111 for 16 bytes. Typical indirect access registers uses following SFR register-pair to access through: DCIR/DDR, MCIR/MDR, EPCR/EPDR, TCIR/TDR, SPICIR/SPIDR, OWCIR/OWDR, etc.
IR_SFR	0011 xxxx (0x30~0x3F)	Indirect Read SFR register. This command requests to indirectly read various bytes of data through the specified command index register in SFR from the given indirect register in this chip. The xxxx indicates the number of bytes to be read from the target indirect register. For example, xxxx = 0000 for 1 byte, and xxxx = 1111 for 16 bytes. Typical indirect access registers uses following SFR register-pair to access through: DCIR/DDR, MCIR/MDR, EPCR/EPDR, TCIR/TDR, SPICIR/SPIDR, OWCIR/OWDR, etc.
BWDM	1100 0000	Burst write data to xDATA memory of this chip. The command requests to do burst or single write to xDATA memory. See Note 1.
BRDM	0100 0000	Burst read data from xDATA memory of this chip. This command requests to do burst or single read from the xDATA memory. See Note 1.

Note:

1. The memory burst write and memory burst read commands have no pre-defined limit on the number of burst bytes, user can define your own burst command in the lower nibble of the Op-Code like in first 4 commands to assist software decoding and handshaking between the external I2C master controller and internal software of AX11025.

Table 47: Reference Command Instructions in I2C Slave Mode

Reference Transfer Format in I2C Slave Mode

SW_SFR

1010 0000

S	Device Addr	W	A	Instruction	A	SFR Addr	A	Data	A	P

1010 0001

S	Device Addr	W	A	Instruction	A	SFR Addr	A	Data$_0$	A	Data$_1$	A	P

SR_SFR

0010 0000

S	Device Addr	W	A	Instruction	A	SFR Addr	A	RS	Device Addr	R	A	Data	NA	P

0010 0001

S	Device Addr	W	A	Instruction	A	SFR Addr	A	RS	Device Addr	R	A	Data$_0$	A	Data$_1$	NA	P

IW_SFR

1011 0000

S	Device Addr	W	A	Instruction	A	Cmd Idx Reg.	A	Ind Reg.	A	Data$_0$	A	P

1011 0001

S	Device Addr	W	A	Instruction	A	Cmd Idx Reg.	A	Ind Reg.	A	Data$_0$	A	Data$_1$	A	P

IR_SFR

0011 xxxx

S	Device Addr	W	A	Instruction	A	Cmd Idx Reg.	A	Ind Reg.	A	RS	Device Addr	R	A		(next)

(cont)		Data$_0$	A	≈		Data$_N$	NA	P

Where DataN = the (xxxx+1)th byte.

BWDM

1100 0000

S	Device Addr	W	A	Instruction	A	Mem Addr B0	A	≈	Mem Addr B2	A	Data$_0$	A		(next)

(cont)		≈	Data$_N$	A	P

BRDM

0100 0000

S	Device Addr	W	A	Instruction	A	Mem Addr B0	A	≈	Mem Addr B2	A	RS	Device Addr	R	A		(next)

(cont)		Data$_0$	A	≈		Data$_N$	NA	P

4.19 1-Wire Controller

The 1-Wire controller of AX11025 is a master-mode controller that controls the communication with multiple external 1-Wire devices. The data transmissions on 1-Wire bus are bit-asynchronous and half-duplex mode only. The 1-Wire controller provides some registers for software to easily perform reading/writing data from/to the 1-Wire devices without having to deal with time-consuming bus timing and control sequences on 1-Wire bus. It supports Standard mode, Standard – Long line mode, and Overdrive mode to work with various 1-Wire devices. For detailed 1-Wire interface timing, please refer to section 5.4.5.

The 1-Wire controller also supports Search ROM Accelerator, which relieves CPU from any single bit operations on the 1-Wire Bus. As shown in figure below, it also provides a strong pull-up control pin, STPZ, for the case of large loading or long line conditions. The DQ is an open-drain pin, which needs an external pulled-up resistor or a strong pull-up through a PMOS transistor.

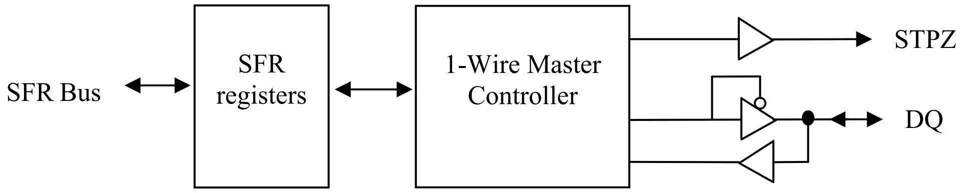


Figure 112: 1-Wire Controller Block Diagram

4.19.1 1-Wire Controller SFR Register Map

Address	Name	Description
0xD6	OWCIR	1-Wire Command Index Register is used to indicate the address of 1-Wire controller register.
0xD7	OWDR	1-Wire Data Register is used to read data from or write data to the specified 1-Wire controller register.

Table 48: 1-Wire Controller SFR Register Map

1-Wire Command Index Register (OWCIR, 0xD6)

Bit	7	6	5	4	3	2	1	0
Name	OWCIR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	OWCIR	WO	Indicate which of the 1-Wire controller register as listed in Table 49 is to be accessed.

1-Wire Data Register (OWDR, 0xD7)

Bit	7	6	5	4	3	2	1	0
Name	OWDR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	OWDR	R/W	Data Register is used to write data to or read data from the 1-Wire controller registers.

1-Wire Controller Register Indirect Access Method

Software shall use indirect access method through OWCIR and OWDR registers to do read and write access to the 1-Wire controller registers as listed in Table 49 below.

Read a register from 1-Wire controller:

Step 1. Write OWCIR: Software indicates the 1-Wire controller register address to be accessed as the data and write it to the SFR register OWCIR.

Step 2. Read OWDR: Software then read SFR register OWDR. The data read from OWDR is the 1-Wire controller register data indicated in step 1.

Write a register to 1-Wire controller:

Step 1. Write OWDR: Software writes the data you want to write into 1-Wire controller registers to the SFR register OWDR.

Step 2. Write OWCIR: After writing 1-Wire controller register data to OWDR, software then indicates the target 1-Wire controller register address as data and write it to OWCIR.

Note: While software is reading or writing 1-Wire controller registers during a sequence of SFR accesses, software can abort that process by writing OWCIR with 0xFF.

1-Wire Controller Register Map

Address	Name	Description
0x00	OWCR	1-Wire Command Register is used to configure 1-Wire operation mode.
0x01	OWTRR	1-Wire Transmit/Receive buffer Register is used to hold data to be transmitted or data being received from 1-Wire devices.
0x02	OWISR	1-Wire Interrupt Status Register.
0x03	OWIER	1-Wire Interrupt Enable Register.
0x04	OWCTR	1-Wire Control Register.
0x05	OWCD	1-Wire Clock Divider Register.

Table 49: 1-Wire Controller Register Map

1-Wire Command Register (OWCR, 0x00)

Bit	7	6	5	4	3	2	1	0
Name	Reserved					FOW	SRA	1WR
Reset Value	0_0000					0	0	0

Bit	Name	Access	Description
0	1WR	W1	1-Wire Reset. 1: If this bit is set a reset will be generated on the 1-Wire bus. Setting this bit automatically clears the SRA bit. 0: This bit will be automatically cleared as soon as the 1-Wire reset completes. The 1-Wire Master will set the Presence Detect interrupt flag (PD) when the reset is complete and sufficient time for a presence detect to occur has passed. The result of the presence detect will be reported in the PDR bit (OWISR.1). If a presence detect pulse was received PDR bit will be cleared, otherwise it will be set.
1	SRA	R/W	Search ROM Accelerator. 1: When this bit is set, the 1-Wire Master will switch to Search ROM Accelerator mode 0: When this bit is set to 0, the master will function in its normal mode. This bit is cleared to 0 on a power-up or master reset.
2	FOW	R/W	Force One Wire. 1: This bit can be used to bypass 1-Wire Master operations and drive the bus directly if needed. Setting this bit high will drive the bus low until it is cleared or the 1-Wire Master reset. While the 1-Wire bus is held low no other 1-Wire Master operations will function. By controlling the length of time this bit is set and the point when the

			line is sampled, any 1-Wire communication can be generated by the software. To prevent accidental writes to the bus, the EN_FOW bit (OWCTR.2) must be set to a 1 before this bit will function. 0: software is not forcing the 1-Wire bus. This bit is cleared to a 0 on power-up or master reset.
7:3	Reserved	-	

1-Wire Transmit/Receive buffer Register (OWTRR, 0x01)

Bit	7	6	5	4	3	2	1	0
Name	DATA							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	DATA	R/W	When the BIT_CTL bit (OWCTR.5) is 0, this holds the 8-bit data to be sent to or being received from the 1-Wire device. The LSB bit (bit 0) is serially shifted out or received in first. When the BIT_CTL bit is 1 (in “Bit Banging” mode), the LSB bit holds the 1-bit data to be sent to or being received from the 1-Wire device.

1-Wire Interrupt Status Register (OWISR, 0x02)

Bit	7	6	5	4	3	2	1	0
Name	OW_LOW	OW_SHORT	RSRF	RBF	TSRE	TBE	PDR	PD
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	PD	CR	Presence Detect. 1: After a 1-Wire Reset has been issued, this flag will be set to 1 after the appropriate amount of time for a presence detect pulse to have occurred. 0: This flag will be 0 when the master has not issued a 1-Wire Reset since the previous read of the OWISR.
1	PDR	RO	Presence Detect Result. When a presence detect interrupt occurs, this bit will reflect the result of the presence detect read - 1: if no part was found. 0: if a slave was found.
2	TBE	RO	Transmit Buffer Empty. 1: This flag will be set to 1 when there is nothing in the Transmit Buffer and it is ready to receive the next byte of data. 0: When it is 0, it indicates that the Transmit Buffer is waiting for the Transmit Shift Register to finish sending its current data before updating it. This bit is cleared when data is written to the Transmit Buffer.
3	TSRE	RO	Transmit Shift Register Empty. 1: This flag will be set to 1 when there is nothing in the Transmit Shift Register and it is ready to receive the next byte of data to be transmitted from the Transmit Buffer. 0: When this bit is 0, it indicates that the Transmit Shift Register is busy sending out data. This bit is cleared when data is transferred from the Transmit Buffer to the Transmit Shift Register.
4	RBF	RO	Receive Buffer Full. 1: This flag will be set to 1 when there is a byte of data waiting to be read in the Receive Buffer. 0: When this bit is 0, it indicates that the Receive Buffer has no new data to be read. This bit will be cleared when the byte is read from the Receive Buffer. Following a read of the OWISR, while Enable Receive Buffer Full Interrupt (ERBF) is set to 1, if the ERBF is not cleared and the value is not read from the Receive Buffer, the

			interrupt will fire again.
5	RSRF	RO	Receive Shift Register Full. 1: This flag will be set to 1 when there is a byte of data waiting in the Receive Shift Register. 0: When this bit is 0, it indicates that the Receive Shift Register is either empty or currently receiving data. This bit will be cleared by the hardware when data in the Receive Shift Register is transferred to the Receive Buffer.
6	OW_SHORT	CR	One Wire Short. 1: This flag will be set to a 1 when the 1-Wire line was low before the master was able to send out the beginning of a reset or a time slot. 0: When this flag is 0, it indicates that the 1-Wire line was high as expected prior to all resets and time slots.
7	OW_LOW	CR	One Wire Low. 1: This flag will be set to 1 when the 1-Wire line is low while the master is in idle signaling that a slave device has issued a presence pulse on the 1-Wire (DQ) line.

1-Wire Interrupt Enable Register (OWIER, 0x03)

Bit	7	6	5	4	3	2	1	0
Name	EOWL	EOWSH	ERSRF	ERBF	ETSRE	ETBE	Reserved	EPD
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	EPD	R/W	Enable Presence Detect Interrupt. 1: Setting this bit to a 1 enables the Presence Detect Interrupt. If set, interrupt will be asserted on INT4 when the PD flag is set. 0: Clearing this bit disables PD as an active interrupt source.
1	Reserved	-	
2	ETBE	R/W	Enable Transmit Buffer Empty Interrupt. 1: Setting this bit to a 1 enables the Transmit Buffer Empty Interrupt. If set, interrupt will be asserted on INT4 when the TBE flag is set. 0: Clearing this bit disables TBE as an active interrupt source.
3	ETSRE	R/W	Enable Transmit Shift Register Empty Interrupt. 1: Setting this bit to a 1 enables the Transmit Shift Register Empty Interrupt. If set, interrupt will be asserted on INT4 when the TSRE flag is set. 0: Clearing this bit disables TSRE as an active interrupt source.
4	ERBF	R/W	Enable Receive Buffer Full Interrupt. 1: Setting this bit to a 1 enables the Receive Buffer Full Interrupt. If set, interrupt will be asserted on INT4 when the RBF flag is set. 0: Clearing this bit disables RBF as an active interrupt source.
5	ERSRF	R/W	Enable Receive Shift Register Full Interrupt. 1: Setting this bit to a 1 enables the Receive Shift Register Full Interrupt. If set, interrupt will be asserted on INT4 when the RSRF flag is set. 0: Clearing this bit disables RSRF as an active interrupt source.
6	EOWSH	R/W	Enable One Wire Short Interrupt. 1: Setting this bit to a 1 enables the One Wire Short Interrupt. If set, interrupt will be asserted on INT4 when the OW_SHORT flag is set. 0: Clearing this bit disables OW_SHORT as an active interrupt source.
7	EOWL	R/W	Enable One Wire Low Interrupt. 1: Setting this bit to a 1 enables the One Wire Low Interrupt. If set, interrupt will be asserted on INT4 when the OW_LOW flag is set. 0: Clearing this bit disables OW_LOW as an active interrupt source.

1-Wire Control Register (OWCTR, 0x04)

Bit	7	6	5	4	3	2	1	0
Name	Reserved	OD	BIT_CTL	STP_SPLY	STPEN	EN_FOW	PPM	LLM
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	LLM	R/W	Long Line Mode. 1: Setting this bit to a 1 will enable Long Line Mode timings on the 1-Wire line during standard mode communications. This mode effectively moves the write one release, the data sampling, and the time slot recovery times out to roughly 8us, 22us, and 14us respectively. This provides a less strict environment for long line transmissions. 0: Clearing this bit to 0 leaves the write one release, the data sampling, and the time slot recovery times at roughly 5us, 15us, and 7us respectively.
1	PPM	R/W	Presence Pulse Masking Mode. 1: Setting this bit to a 1 will enable Presence Pulse Masking Mode. This mode causes the master to initiate the falling edge of a presence pulse during a 1-Wire Reset before the fastest slave would initiate one. This enables the master to prevent the larger amount of ringing caused by the slave devices when initiating a low on the DQ line. If the PPM bit is set, the PDR result bit in the Interrupt Register will always be set to a 0 showing that a slave device was on the line even if there were none. 0: Clearing this bit to a 0 disables the Presence Pulse Masking Mode. This mode only support standard mode.
2	EN_FOW	R/W	Enable Force One Wire. 1: Setting this bit to a 1 will enable the functionality of the Force One Wire (FOW) bit (OWCR .2). 0: Clearing this bit will disable the functionality of the FOW bit.
3	STPEN	R/W	Strong Pull-up Enable. 1: Setting this bit to a 1 enables the strong pull-up output enable (STPZ) pin's functionality which allows this output pin to enable an external strong pull-up any time the master is not pulling the line low or waiting to read a value from a slave device. This functionality is used for meeting the recovery time requirement in Overdrive mode and long-line standard communications. 0: Clearing this bit to a 0 will disable the STPZ output pin.
4	STP_SPLY	R/W	Strong Pull-up Supply. 1: Setting this bit to a 1 while STPEN is also set to a 1 will enable the STPZ output while the master is in an IDLE state. This will provide a stiff supply to devices requiring high current during operations. 0: Clearing this bit to a 0 disables the STPZ output while the master is in an IDLE state. The STP_SPLY bit is a don't-care if STPEN is set to a 0.
5	BIT_CTL	R/W	Bit Control. 1: Setting this bit to a 1 will place the master into its "Bit Banging" mode of operation. In this mode, only the least significant bit of the Transmit/Receive register would be sent/received before enabling the interrupt flags that signal the end of the transmission. 0: Clearing this bit to 0 leaves the master operating in full byte boundaries.
6	OD	R/W	Overdrive. 1: Setting this bit to a 1 will place the master into Overdrive mode that effectively changes the master's 1-Wire timings to match those outlined for Overdrive in the Book of iButton Standards. 0: Clearing this bit to a 0 leaves the master operating in Standard mode speed.
7	Reserved	-	

1-Wire Clock Divider Register (OWCD, 0x05)

Bit	7	6	5	4	3	2	1	0
Name	CLK_EN	Reserved		DIV			PRE	
Reset Value	0	00		000			00	

<table>
<tr><th>Bit</th><th>Name</th><th>Access</th><th>Description</th></tr>
<tr><td>1:0</td><td>PRE</td><td>R/W</td><td rowspan="2"><table><tr><th>Operating System Clock Frequency (MHz)</th><th>Divider Ratio</th><th>DIV[2:0]</th><th>PRE[1:0]</th></tr><tr><td>25</td><td>24</td><td>011</td><td>01</td></tr><tr><td>50</td><td>48</td><td>100</td><td>01</td></tr><tr><td>100</td><td>96</td><td>101</td><td>01</td></tr></table></td></tr>
<tr><td>4:2</td><td>DIV</td><td>R/W</td></tr>
<tr><td>6:5</td><td>Reserved</td><td>-</td><td></td></tr>
<tr><td>7</td><td>CLK_EN</td><td>R/W</td><td>Clock Enable for 1-Wire controller and its bus timing control logic.
1: Enable 1-Wire controller and its bus timing control logic.
0: Disable 1-Wire controller and its bus timing control logic.</td></tr>
</table>

4.20 SPI Controller

The serial peripheral interface (SPI) controller of AX11025 provides a full-duplex, synchronous serial communication interface (4 wires) to flexibly work with numerous peripheral devices or micro-controller with SPI. As shown in Figure 113 below, the SPI controller consists of a SPI master controller with 3 slave select pins, SS0, SS1, SS2, to connect up to 3 SPI devices, and a SPI slave controller to support communication with external micro-controller with SPI master.

The SPI master controller supports 4 types of interface timing mode, namely, Mode 0, Mode 1, Mode 2, and Mode 3 to allow working with most SPI devices available. Please see Figure 114 for the timing diagram of these timing modes. It supports variable length of transfer word up to 32 bits per software command or even extended length of transfer word for a long burst transfer by keeping slave select pins active. It supports either MSB or LSB first data transfer, and the operating SPI clock, SCLK, is programmable by software and can be run up to 25 Mhz when operating system clock is at 100MHz.

The SPI slave controller allows an external micro-controller with SPI master to communicate with AX11025. It supports 2 types of interface timing mode, namely, Mode 0 and Mode 3. In slave mode, only MSB first data transfer is supported and only the slave select pin, SS0, is used. The SPI slave controller supports 8 flexible command instructions for the external micro-controller to access the internal registers and memory resources of AX11025. It contains a 16-bytes FIFO to hold receive/transmit data on SPI interface and the SPI clock can be run up to 6 Mhz when operating system clock is at 100MHz.

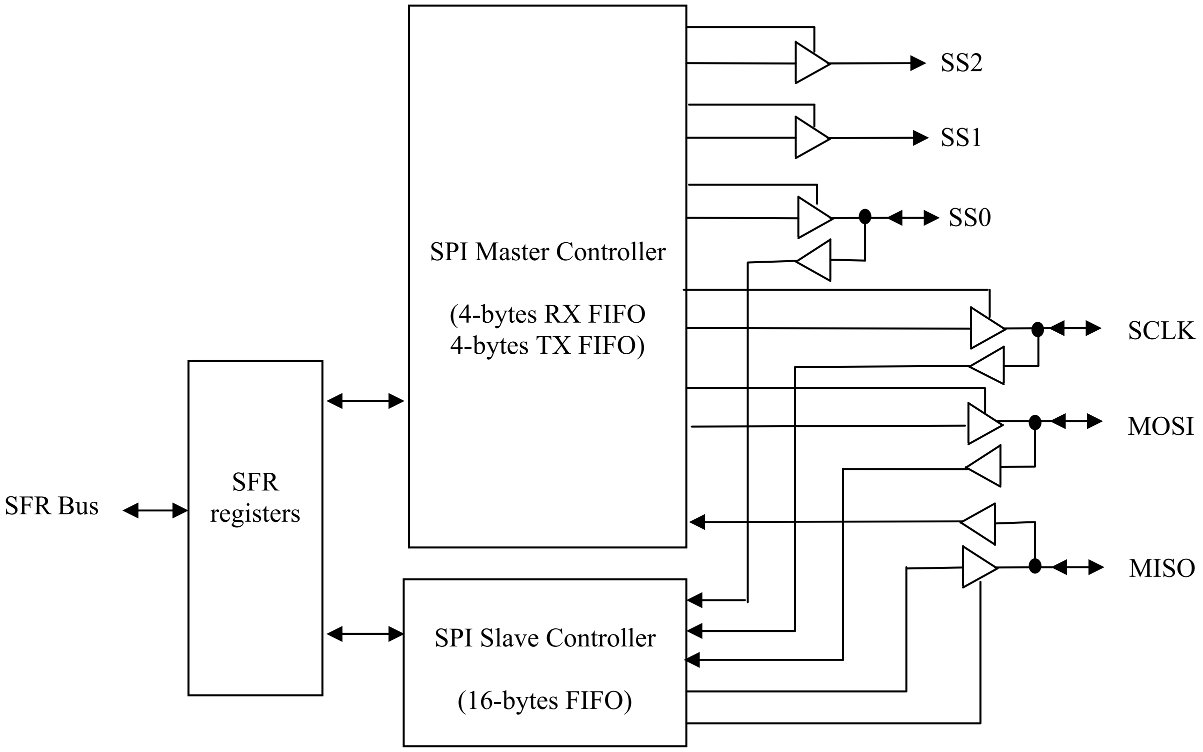
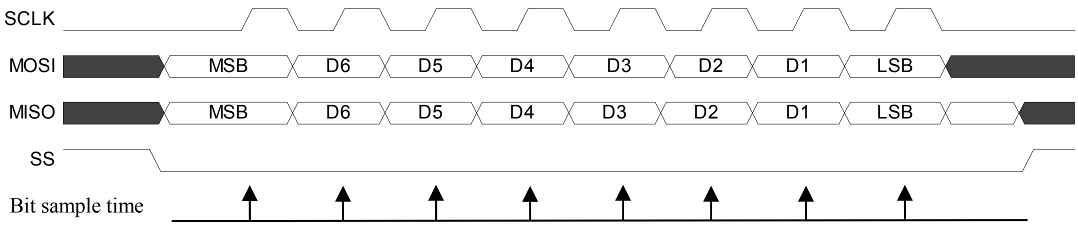


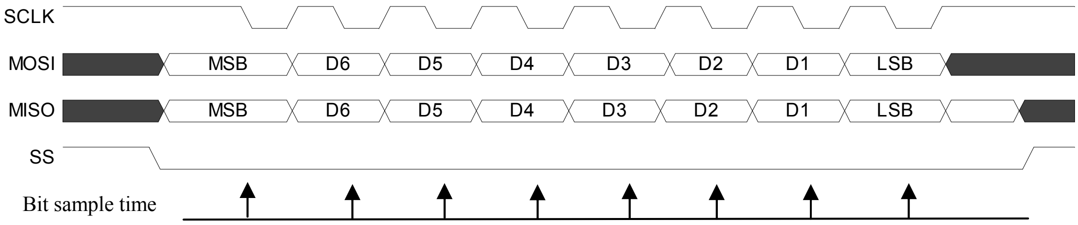
Figure 113: SPI Controller Block Diagram

Mode 0: CPHA (SPICR.1) = 0, CPOL (SPICR.2) = 0, LSB (SPICR.3) = 0, SPIMCR[CHAR_LEN] = 00111.



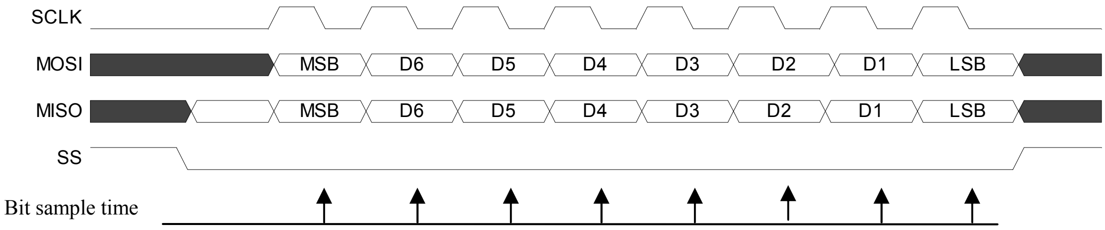
Note: SCLK pin needs external pull-down resistor and SSx pins need external pull-up resistor in Mode 0, SPI master mode.

Mode 1: CPHA (SPICR.1) = 0, CPOL (SPICR.2) = 1, LSB (SPICR.3) = 0, SPIMCR [CHAR_LEN] = 00111.



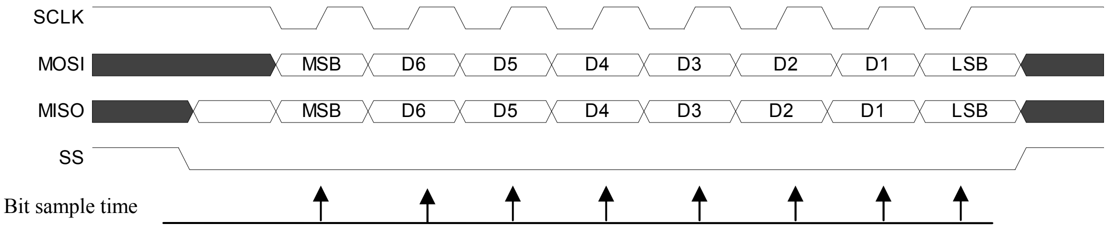
Note: SCLK pin needs external pull-up resistor and SSx pins need external pull-up resistor in Mode 1, SPI master mode.

Mode 2: CPHA (SPICR.1) = 1, CPOL (SPICR.2) = 0, LSB (SPICR.3) = 0, SPIMCR[CHAR_LEN] = 00111.



Note: SCLK pin needs external pull-down resistor and SSx pins need external pull-up resistor in Mode 2, SPI master mode.

Mode 3: CPHA (SPICR.1) = 1, CPOL (SPICR.2) = 1, LSB (SPICR.3) = 0, SPIMCR[CHAR_LEN] = 00111.



Note: SCLK pin needs external pull-up resistor and SSx pins need external pull-up resistor in Mode 3, SPI master mode.

Figure 114: SPI Timing Diagram

4.20.1 SPI SFR Register Map

Address	Name	Description
0xCE	SPICIR	SPI Command Index Register is used to indicate the address of SPI controller register.
0xCF	SPIDR	SPI Data Register is used to read data from or write data to the specified SPI controller register.

Table 50: SPI Controller SFR Register Map

SPI Command Index Register (SPICIR, 0xCE)

Bit	7	6	5	4	3	2	1	0
Name	SPICIR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	SPICIR	WO	Indicate which of the SPI controller register as listed in Table 51 is to be accessed.

SPI Data Register (SPIDR, 0xCF)

Bit	7	6	5	4	3	2	1	0
Name	SPIDR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	SPIDR	R/W	Data Register is used to write data to or read data from the SPI controller registers.

SPI Controller Register Indirect Access Method

Software shall use indirect access method through SPICIR and SPIDR registers to do read and write access to the SPI controller registers as listed in Table 51 below.

Read a register from SPI controller:

Step 1. Write SPICIR: Software indicates the SPI controller register address to be accessed as the data and write it to the SFR register SPICIR.

Step 2. Read SPIDR: Software then read SFR register SPIDR. The data read from SPIDR is the SPI controller register data indicated in step 1. Keep reading from SPIDR if the SPI controller registers have more than one byte, in that case, the first byte being read back is LSB byte.

Write a register to SPI controller:

Step 1. Write SPIDR: Software writes the data you want to write into SPI controller registers to the SFR register SPIDR. Keep writing to SPIDR if the SPI controller registers have more than one byte, in that case, the first byte being written should be LSB byte.

Step 2. Write SPICIR: After writing SPI controller register data to SPIDR, software then indicates the target SPI controller register address as data and write it to SPICIR.

Note: While software is reading or writing SPI controller registers during a sequence of SFR accesses, software can abort that process by writing SPICIR with 0xFF.

SPI Controller Register Map

Address	Name	Description
0x00	SPIRBR	SPI RX Buffer Register
0x04	SPITBR	SPI TX Buffer Register
0x08	SPICR	SPI Control Register
0x09	SPIMCR	SPI Master Command Register
0x0A	SPIBRR	SPI Baud Rate Register
0x0B	SPISSR	SPI Slave Select Register
0x0C	SPIISR	SPI Interrupt Status Register
0x0D	SPIIER	SPI Interrupt Enable Register
0x0E	SPISCR	SPI Slave Command Register
0x0F	Reserved	
0x10	SPISB	SPI Slave Buffer

Table 51: SPI Controller Register Map

SPI RX Buffer Register (SPIRBR, 0x00)

Bit	7	6	5	4	3	2	1	0
Name	SPIRBR0							
	SPIRBR1							
	SPIRBR2							
	SPIRBR3							
Reset Value	0x0000_0000							

Bit	Name	Access	Description
7:0 … 31:24	SPIRBR0 … SPIRBR3	RO	When in SPI master mode, the SPIRBR registers hold the value of received data of the last executed transfer. Valid bits depend on the CHAR_LEN bits of SPIMCR register. For example, if CHAR_LEN is less or equal to 0_0111, the value of SPIRBR1, SPIRBR2 and SPIRBR3 are undefined; if CHAR_LEN is less than 0_1111, the value of SPITBR2 and SPITBR3 are undefined, and so on.

SPI TX Buffer Register (SPITBR, 0x04)

Bit	7	6	5	4	3	2	1	0
Name	SPITBR0							
	SPITBR1							
	SPITBR2							
	SPITBR3							
Reset Value	0x0000_0000							

Bit	Name	Access	Description
7:0 … 31:24	SPITBR0 … SPITBR3	R/W	When in SPI master mode, the SPITBR registers hold the data to be transmitted in the next transfer. Valid bits depend on the CHAR_LEN bits of SPIMCR. For example, if CHAR_LEN is less or equal to 0_0111, the value of SPITBR1, SPITBR2 and SPITBR3 are undefined, if CHAR_LEN is less than 0_1111, the value of SPITBR2 and SPITBR3 are undefined, and so on.

SPI Control Register (SPICR, 0x08)

Bit	7	6	5	4	3	2	1	0
Name	SSOE	MSS	ASS	SPIEN	LSB	CPOL	CPHA	SSP
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	SSP	R/W	Slave Select pins (SS0, SS1, SS2) active Polarity. This bit is only valid in SPI master mode. 1: The slave select signals are active-high. 0: The slave select signals are active-low. When in SPI slave mode, this chip always uses SS0 for the SPI slave controller and it is always active-low.
1	CPHA	R/W	SPI Clock Phase Bit. This bit is used to control the SCLK pin, serial clock phase vs. serial data. This bit applies to both SPI master and SPI slave mode. 1: The first SCLK edge is issued at the beginning of the 8-cycle transfer operation. 0: The first SCLK edge is issued one-half cycle into the 8-cycle transfer operation.
2	CPOL	R/W	SPI Clock Polarity Bit. 1: Active-low clock selected. 0: Active-high clock selected.
3	LSB	R/W	When in SPI master mode, this bit indicates that the LSB bit is transmitted/received first. 1: The LSB of SPITBR is sent first on the line, and the first bit received from the line will be put in the LSB position in the SPIRBR register. 0: The MSB of SPITBR is transmitted first and the first bit received is put in MSB position of SPIRBR. Note that in SPI slave mode, it is always the MSB bit of each 8-bit data being transmitted or received first.
4	SPIEN	R/W	SPI Enable. 1: SPI controller is enabled. 0: SPI controller is disabled.
5	ASS	R/W	When in SPI master mode, Automatically generate Slave Select signals (SS0, SS1, SS2). 1: The slave select signal is generated automatically. This means that when setting GO_BSY bit of SPIMCR to start the transfer, the slave select signal that is indicated in SPISSR is asserted by the SPI controller automatically and is de-asserted after the transfer is finished. 0: SS0/1/2 signals are asserted and de-asserted by writing and clearing bits in SPISSR register. When this bit is setting to 0, the SSP of SPICR will not effect, and the SS0/1/2 signals is controlled directly by SPISSR register. This bit is only available in SPI master mode.
6	MSS	R/W	Master/Slave mode Select. 1: The SPI controller is set to operate in SPI mater mode. 0: The SPI controller is set to operate in SPI slave mode.
7	SSOE	R/W	Slave Select pins (SS0, SS1, SS2) Output Enable. 1: Enable driving slave select signals. 0: Put slave select signals to tri-state.

SPI Master Command Register (SPIMCR, 0x09)

Bit	7	6	5	4	3	2	1	0
Name	GO_BSY	LL	LCSR	CHAR_LEN				
Reset Value	0	0	0	0_0111				

Bit	Name	Access	Description
4:0	CHAR_LEN	R/W	When in SPI master mode, this field specifies how many bits in SPIRBR and SPITBR are transmitted on each transfer. Up to 32 bits can be transmitted. For example, the value of "0_0111" indicates 8 bits to be transferred.
5	LCSR	R/W	When in SPI master mode, setting '1' to suppress the last SCLK in the current transfer (used in some SPI EEPROM case).
6	LL	R/W	Long Length. 1: The desired transfer data length in one transfer is more than the value of CHAR_LEN. Setting '1' to keep the SS0/1/2 pins asserted after the transfer. This can be used in the case where more than 32 bits of data need to be transferred in one transfer. 0: The desired transfer data length is equal to CHAR_LEN. Setting '0' makes SS0/1/2 pins de-asserted automatically after the transfer.
7	GO_BSY	W1/R	Writing 1 to this bit starts the transfer. This bit remains set during the transfer and is automatically cleared after the transfer finished. Writing 0 to this bit has no effect. This bit is only valid in SPI master mode.

SPI Baud Rate Register (SPIBRR, 0x0A)

Bit	7	6	5	4	3	2	1	0
Name	Divider							
Reset Value	0xFF							

Bit	Name	Access	Description
7:0	Divider	R/W	The value in this field determines the frequency divider of the operating system clock to generate the serial clock SCLK output. The desired frequency is obtained according to the following equation: $\text{SCLK Frequency} = \dfrac{\text{Operating System Clock Frequency}}{(\text{Divider}+1)*2}$

SPI Slave Select Register (SPISSR, 0x0B)

Bit	7	6	5	4	3	2	1	0
Name	Reserved					SS		
Reset Value	11111					111		

Bit	Name	Access	Description
2:0	SS	R/W	When in SPI master mode, this is used to select the desired slave device to communicate to. For example, set SS = 110 to activate the SS0 pin. When in SPI slave mode, this chip always uses SS0 for the SPI slave controller and it is always active-low.
7:3	Reserved	-	

SPI Interrupt Status Register (SPIISR, 0x0C)

Bit	7	6	5	4	3	2	1	0
Name	Reserved			SRCF	Reserved			STCF
Reset Value	000			0	000			0

Bit	Name	Access	Description
0	STCF	CR	SPI Transceiver Complete Flag in SPI master mode. 1: This flag is asserted after the requested transfer (via setting GO_BUSY bit in SPIMCR) is completed. 0: The SPI bus is idle or the transfer is in progress.
3:1	Reserved	-	
4	SRCF	CR	SPI Receive Complete Flag in SPI slave mode. 1: This flag is asserted every time when the SPISB contain valid data received from the external SPI master after one transfer. Note that in Table 52, all the received instructions, except for the RSR and RDR, will cause this bit to be set after transfer completed. 0: The SPI bus is idle or the transfer is in progress.
7:5	Reserved	-	

SPI Interrupt Enable Register (SPIIER, 0x0D)

Bit	7	6	5	4	3	2	1	0
Name	Reserved			SRCFIE	Reserved			STCFIE
Reset Value	000			0	000			0

Bit	Name	Access	Description
0	STCFIE	R/W	SPI Transmit Complete Flag Interrupt Enable. 1: Enable interrupt on INT4 whenever STCF flag (SPIISR.0) is asserted. 0: Disable interrupt.
3:1	Reserved	-	
4	SRCFIE	R/W	SPI Receive Complete Flag Interrupt Enable. 1: Enable interrupt on INT4 whenever SRCF flag (SPIISR.4) is asserted. 0: Disable interrupt.
7:5	Reserved	-	

SPI Slave Command Register (SPISCR, 0x0E)

Bit	7	6	5	4	3	2	1	0
Name	Reserved							RDY
Reset Value	000_0000							1

Bit	Name	Access	Description
0	RDY	W1/R	During initialization, software shall set this bit to '1' to indicate to the external SPI master that this chip is ready to receive any commands. This bit will be reflected in the RSR instruction as listed in Table 52. This is only valid in SPI slave mode. When external SPI master needs to read data from this chip, software first prepares the data in SPISB register and then set this bit '1' to indicate to the external SPI master that the data is ready to be retrieved by the external SPI master. When external SPI master needs to write data to this chip, it initiates the SPI bus access and then checks for completion indication from this chip. Software of AX11025 shall retrieve the data from SPISB register and then sets this bit '1' to indicate that the requested write operation has been completed by this chip.
7:1	Reserved	-	

SPI Slave Buffer (SPISB, 0x10)

Bit	7	6	5	4	3	2	1	0
Name	SB0							
	SB1							
	SB2							
	SB3							
	SB4							
	SB5							
	SB6							
	SB7							
	SB8							
	SB9							
	SB10							
	SB11							
	SB12							
	SB13							
	SB14							
	SB15							
Reset Value	0x0000_0000_0000_0000_0000_0000_0000_0000							

Bit	Name	Access	Description
7:0 … 127:120	SB0 … SB15	R/W	Slave Buffer. This is only valid in SPI slave mode. When in SPI slave mode, this holds the data received from the external SPI master. The SB0 holds the first 8-bits received, and SB1 holds the second 8-bits received, and so on. Note that the transfer of each 8-bit serial data is always MSB first. When external SPI master issues the read command, software can put requested read data in SPISB here. Again SB0 holds the first 8-bits transmitted data, and SB1 holds the second 8-bits transmitted data, and so on.

Example Programming Procedure in SPI Master Mode

Example 1: Configure to SPI Mode 0, SPI frequency is 1.6MHz, and enable interrupt mode. Write 2 bytes of data = 0x0500 to slave device.

1. Write 0xFE to SPISSR register.
2. Write 0x1D to SPIBRR register.
3. Write 0x01 to SPIIER register.
4. Write 0xF0 to SPICR register.
5. Write 0x00, 0x05, 0x00, and 0x00 to SPITBR.
6. Write 0x8F to SPICMR register.
7. Wait interrupt.
8. Read SPIISR register to clear STCF.
9. Read SPIRBR register if needed.

Example 2: Read 1 byte of data from slave device.

1. Write 0xFE to SPISSR register.
2. Write 0x1D to SPIBRR register.
3. Write 0x01 to SPIIER register.
4. Write 0xF0 to SPICR register.
5. Write 0x87 to SPICMR register.
6. Wait interrupt.
7. Read SPIISR register to clear STCF.
8. Read SPIRBR register.

4.20.2 SPI Slave Mode Function Description

The SPI slave controller of this chip provides 8 different commands for the external SPI master controller to access it. These commands are shown in Table 52 and the command frame format is shown in Figure 115. Note that the serial data is always the MSB bit of each 8-bit data being transmitted or received first.

The external SPI master may use these commands to read or write the SFR registers or xDATA memory of this chip. The SPI slave controller hardware shall parse the Op-Code byte and user should follow the definition here, all other bytes in the transfer are parsed by AX11025 software. Therefore, these commands are allowed to make certain changes to meet user's applications.

Command Name	**Op-Code**	**Operation Description**
RSR	0000_0000 (0x00)	Read Status Register. When external SPI master needs to send some data to this chip, the returned status of 0x01 indicates that this chip is ready to receive new data. To avoid the internal 16-bytes FIFO overflow, the external SPI master shall check this status before sending next data to this chip. Returning 0x00 indicates the internal FIFO is still being used and this chip is not ready. When external SPI master needs to receive some data from this chip, returning 0x01 indicates that the requested data is ready in SPISB and the external SPI master can issue RDR command to receive the requested data. Returning 0x00 indicates the data is not ready.
RDR	0001_0000 (0x10)	Read Data Register. This is the data port for the external SPI master to receive the requested read data after it has issued SR_SFR, IR_SFR, and BR_MEM commands.
SW_SFR	1010_0xxx (0xA0~0xA7)	Single Write SFR register. This command requests to write various bytes of data to the specified SFR register in this chip with the given data. The xxx indicates the number of bytes to be written to the target registers. For example, xxx = 000 for 1 byte, and xxx = 111 for 8 bytes. After sending this command, to avoid internal 16-byte FIFO being overflowed, the external SPI master should use RSR command to learn that this chip has finished processing the command before it can send next command.
SR_SFR	0010_0xxx (0x20~0x27)	Single Read SFR register. This command requests to read various bytes of data from the specified SFR register in this chip. The xxx indicates the number of bytes to be read from the target registers. For example, xxx = 000 for 1 byte, and xxx = 111 for 8 bytes. After sending this command, the external SPI master should use RSR command to learn that the requested data is available and then use RDR command to receive the data.
IW_SFR	1011_xxxx (0xB0~0xBF)	Indirect Write SFR register. This command requests to indirectly write various bytes of data through the specified command index register in SFR to the given indirect register in this chip. The xxxx indicates the number of bytes to be written to target indirect register. For example, xxxx = 0000 for 1 byte, and xxxx = 1111 for 16 bytes. After sending this command, to avoid internal 16-byte FIFO being overflowed, the external SPI master should use RSR command to learn that this chip has finished processing the command before it can send next command. Typical indirect access registers uses following SFR register-pair to access through: DCIR/DDR, MCIR/MDR, EPCR/EPDR, TCIR/TDR, I2CCIR/I2CDR, OWCIR/OWDR, etc.
IR_SFR	0011_xxxx	Indirect Read SFR register.

	(0x30~0x3F)	This command requests to indirectly read various bytes of data through the specified command index register in SFR from the given indirect register in this chip. The xxxx indicates the number of bytes to be read from the target indirect register. For example, xxxx = 0000 for 1 byte, and xxxx = 1111 for 16 bytes. After sending this command, the external SPI master should use RSR command to learn that the requested data is available and then use RDR command to receive the data. Typical indirect access registers uses following SFR register-pair to access through: DCIR/DDR, MCIR/MDR, EPCR/EPDR, TCIR/TDR, I2CCIR/I2CDR, OWCIR/OWDR, etc.
BW_MEM	1100_xxxx (0xC0~0xCB)	Burst Write data to xDATA memory of this chip. This command requests to write the specified address of xDATA memory in this chip with the specified number of bytes and the given data. The {ADDR2, ADDR1, ADDR0} represents the real address of xDATA memory. The xxxx indicates the number of bytes to be written, starting with the specified address. For example, xxxx = 0000 for 1 byte, and xxxx = 1011 for 12 bytes. The Data0 is written to the {ADDR2, ADDR1, ADDR0}, and the Data1 is written to the {ADDR2, ADDR1, ADDR0}+1, and so on. Note that the fields of ADDR0, ADDR1, ADDR2, Data0, DataN, etc. in BW_MEM command are reference format and can allow making changes as long as the AX11025 software and external SPI master both agree on the format definition. Only that the xxxx in the Op-Code field should match the actual number of bytes being transferred. After sending this command, to avoid internal 16-byte FIFO being overflowed, the external SPI master should use RSR command to learn that this chip has finished processing the command before it can send next command.
BR_MEM	0100_xxxx (0x40~0x4F)	Burst Read Memory. This command requests to read the specified address of xDATA memory in this chip with the specified number of bytes. The {ADDR2, ADDR1, ADDR0} represents the real address of xDATA memory. The xxxx indicates the number of bytes to be read, starting with the specified address. For example, xxxx = 0000 for 1 byte, and xxxx = 1011 for 12 bytes. Note that the fields of ADDR0, ADDR1, ADDR2, etc. in BR_MEM command are reference format and can allow making changes as long as the AX11025 software and external SPI master both agree on the format definition. Only that the xxxx in the Op-Code field should match the actual number of bytes being transferred. After sending this command, the external SPI master should use RSR command to learn that the requested data is available and then use RDR command to receive the data.

Table 52: Command Instruction in SPI Slave Mode

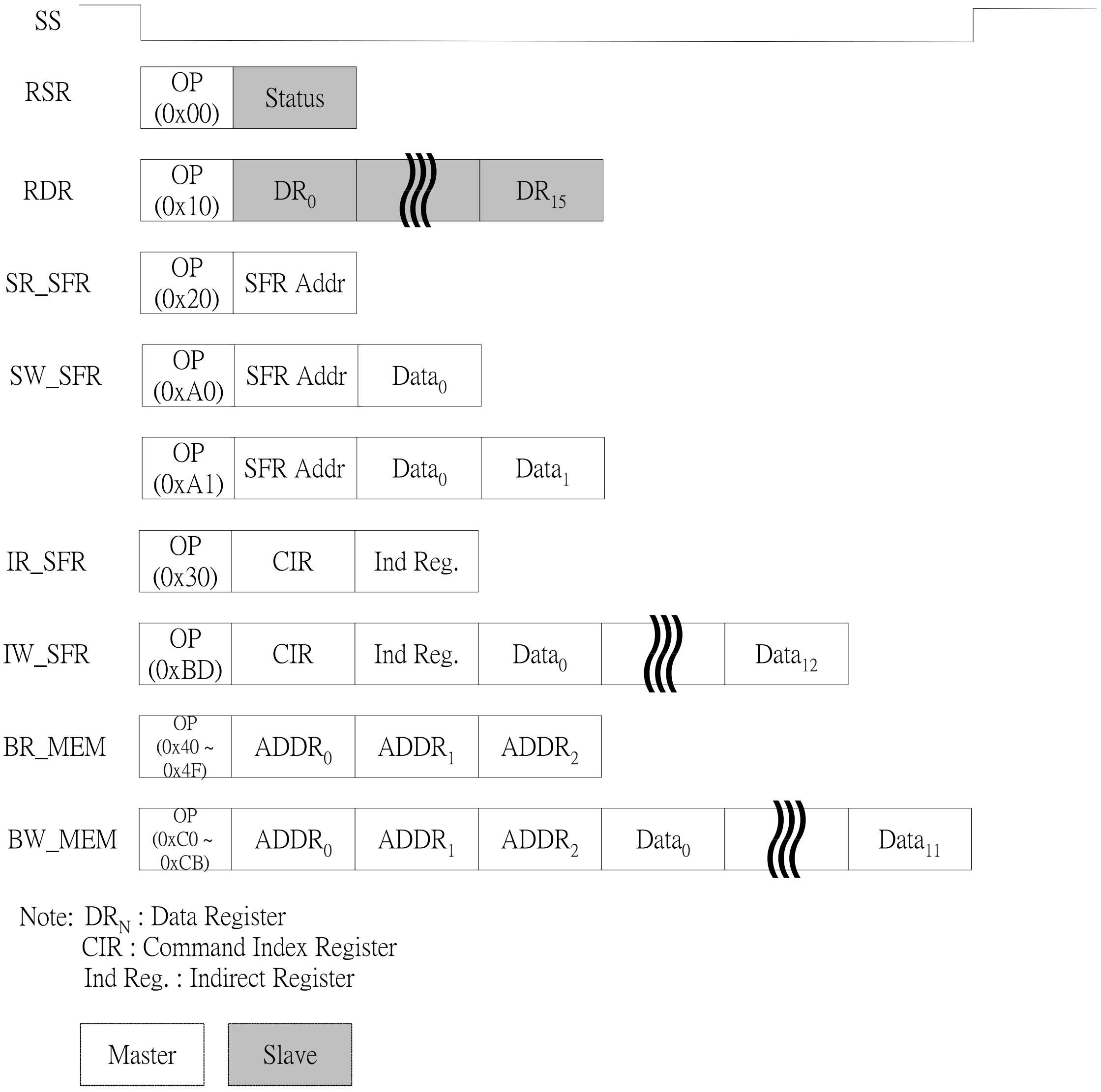
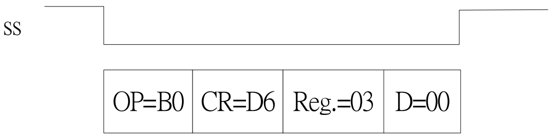


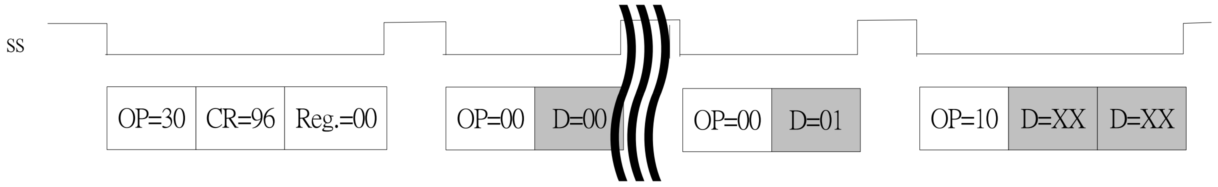
Figure 115: Command Frame Format in SPI Slave Mode

Example Command Frames in SPI Slave Mode

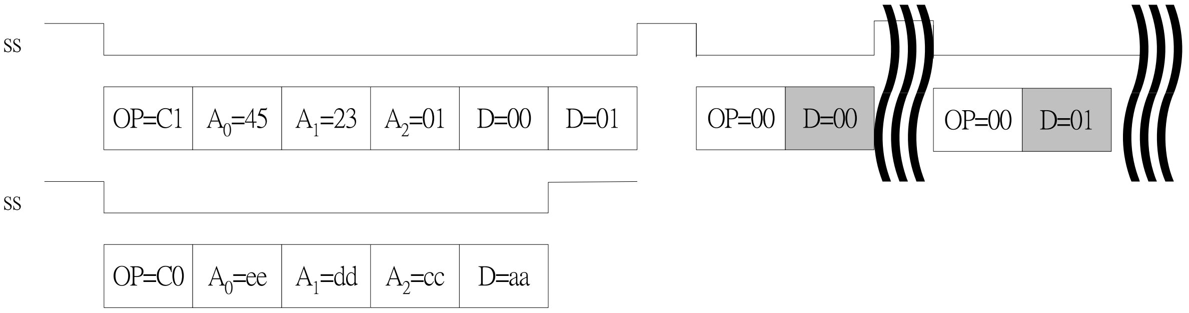
1) Write 0x00 to OWIER register



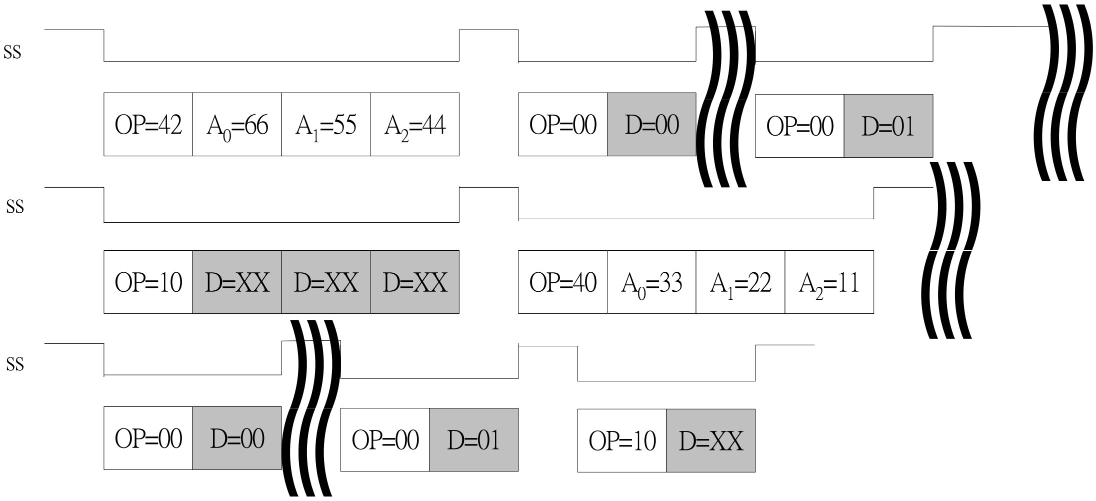
2) Read Data from I2CCPR register



3) Write two bytes data (00,01) starting at memory address = 0x012345 and write one byte data(aa) to the other address = 0xccddee



4) Read three bytes data starting from memory address = 0x445566 and read one byte from the other address = 0x112233



4.21 CAN Controller

The Controller Area Network (CAN) controller of AX11025 supports a serial, asynchronous, multi-master communication bus protocol for connecting microcontrollers in automotive and general industrial automation applications. The CAN is a two-wire, half duplex, high-speed network system and is well suited for high speed applications using short messages. Its robustness, reliability and the large following from the semiconductor industry are some of the benefits with CAN.

The CAN bus is a broadcast type of bus. That means all nodes can "hear" all transmissions. There is no way to send a message to just a specific node, all nodes will invariably pick up all traffic. The CAN controller hardware, however, provides local filtering so that each node may react only on the interesting messages. The bus uses Non-Return-To-Zero (NRZ) with bit stuffing. The modules are connected to the CAN bus in Wire-AND fashion; “Recessive” bits (mostly logic level “1”) are overwritten by “Dominant” bits (mostly logic level “0”).

The data messages transmitted from any node on a CAN bus do not contain addresses of either the transmitting node, or of any intended receiving node. Instead, the content of the message is labeled by an identifier that is unique throughout the network. The unique identifier also determines the priority of the message. The lower the numerical value of the identifier, the higher the priority. This allows arbitration if two (or more) nodes compete for access to the bus at the same time. The higher priority message is guaranteed to gain bus access as if it were the only message being transmitted. Lower priority messages are automatically re-transmitted in the next bus cycle, or in a subsequent bus cycle if there are still other higher priority messages waiting to be sent.

The main features of CAN controller are listed below,

- CAN 2.0B protocol compatible
- Support Standard frame (11-bits Identifier field) and Extended frame (29-bits Identifier field) format messages
- Supports two modes of operation: Basic mode (for Standard frame) and Extended mode (for Standard and Extended frames)
- Support 4 message types, Data frame, Remote frame, Error frame and Overload frame, as shown in Figure 117 ~ Figure 121. Data frame and Remote frame are processed by software; Error frame and Overload frame are processed by hardware
- Contain 64-bytes FIFO as receive buffer. When receive buffer is almost full (less than 23-bytes space left), support flow control in hardware by automatically sending out Overload frame to other nodes right after receiving Data frame. This can temporarily stop other nodes from sending more messages to avoid receive buffer overflow.
- Support 10-bytes transmit buffer in Basic mode or 13-bytes transmit buffer in Extended mode
- Data length from 0 to 8 bytes
- Programmable baud rate generator up to 1 Mbps to handle bit timing on CAN bus. Synchronized to the bit stream on the CAN-bus on a recessive to dominant edge during bus idle indicating Start-of-Frame, and resynchronized on further transitions during the reception of a message. Provides programmable time segment to compensate for the propagation delay times and phase shift and to define the sample point and the number of samples to be taken within a bit time
- Support acceptance filter to filter unwanted CAN message frames in hardware
- Support error detection, error signaling, and fault confinement in hardware
- Automatic retransmission of corrupted message as soon as the bus is idle again
- Extended mode extensions:
    - Arbitration lost interrupts with report of detailed bit position
    - Error counters with read/write access and programmable error warning limit
    - Error interrupts with report of CAN bus error types
    - Listen only mode (no acknowledge, no active error flags)
    - Acceptance filter extension

- Number of available messages within the RX FIFO

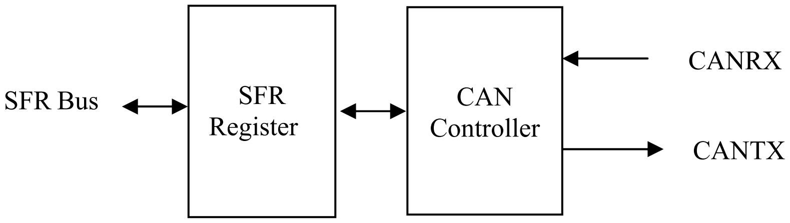


Figure 116: CAN Block Diagram

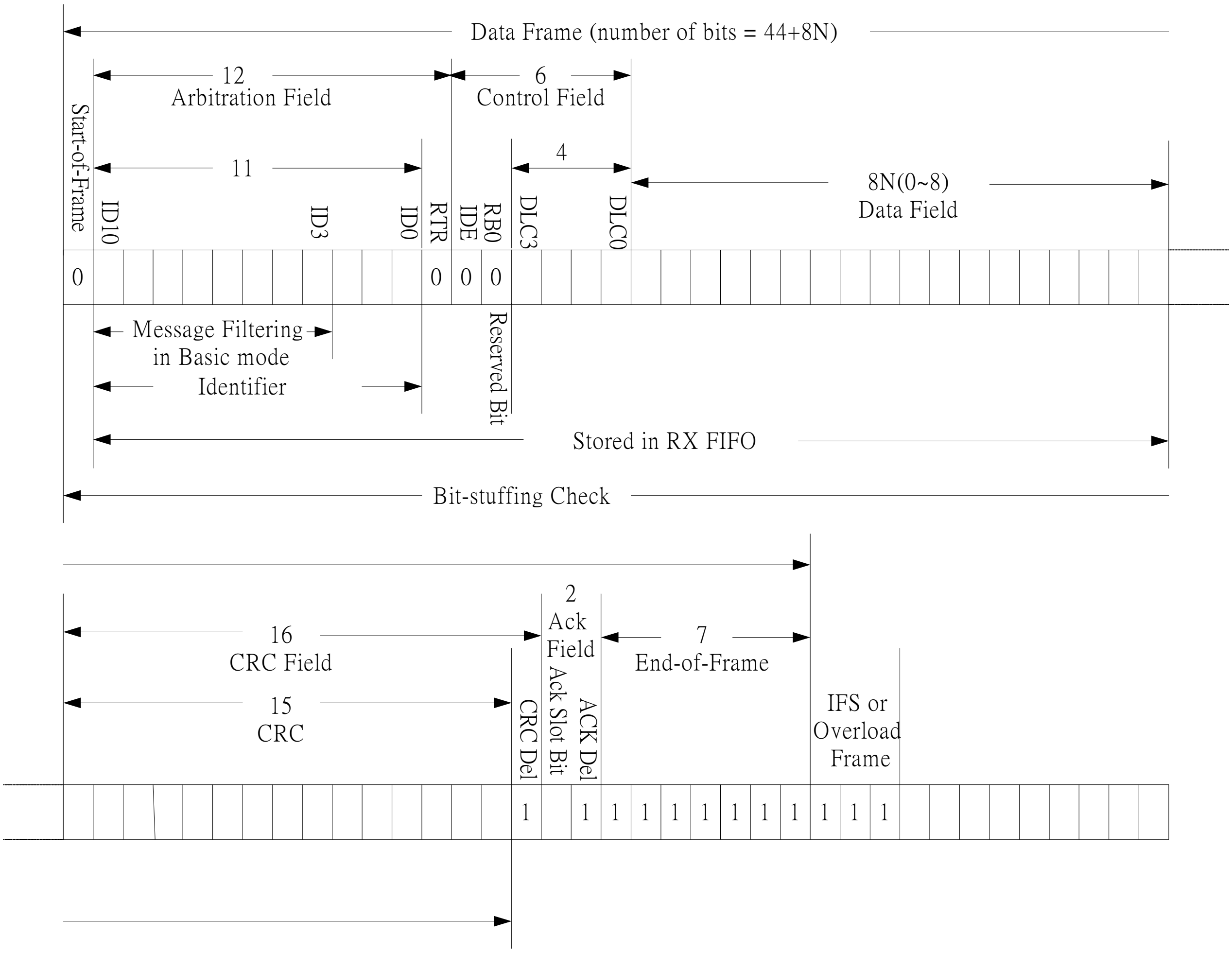


Figure 117: CAN Standard Data Frame (11-bits Identifier field) Format

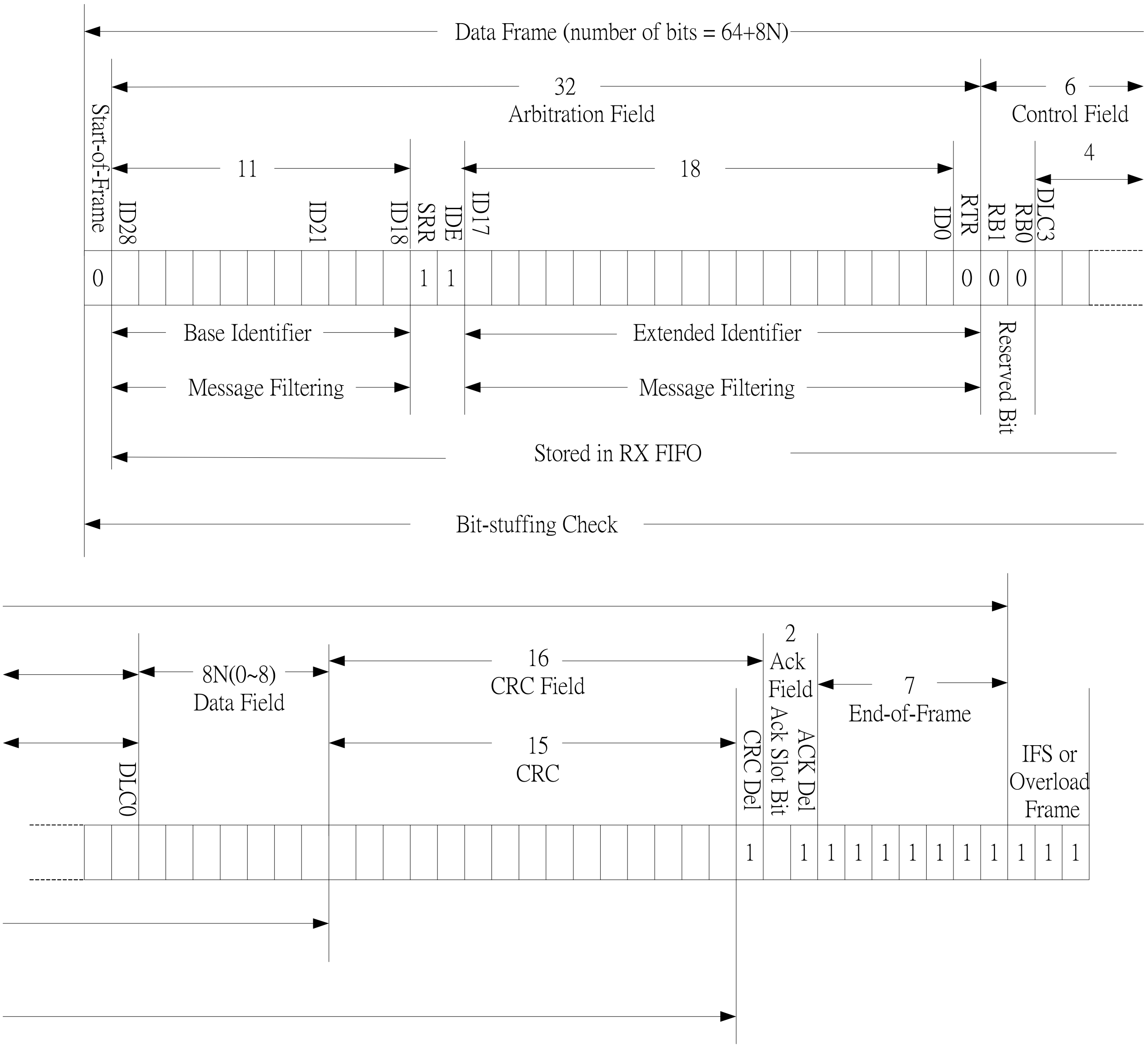


Figure 118: CAN Extended Data Frame (29-bits Identifier field) Format

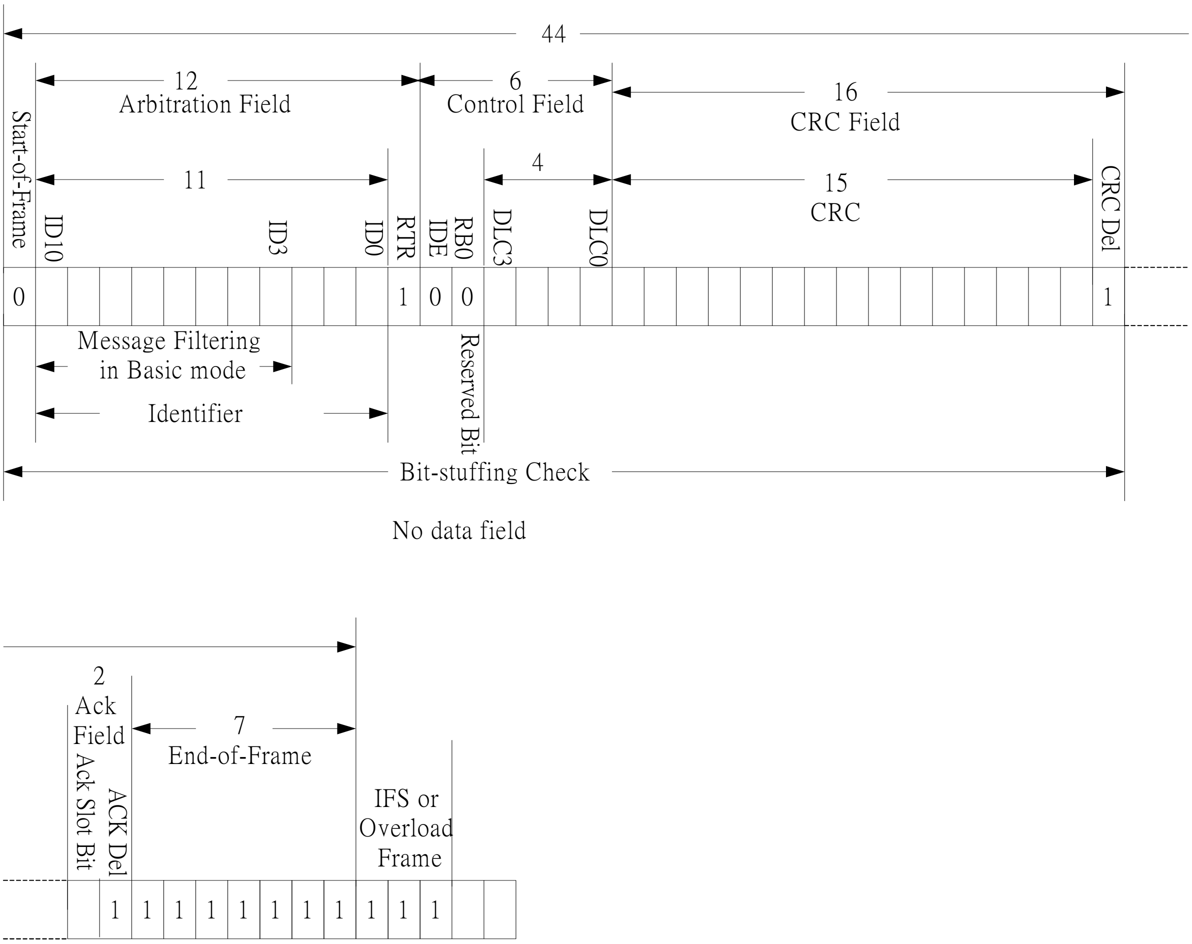


Figure 119: CAN Remote Frame (11-bits Identifier field shown, 29-bits Identifier also supported) Format

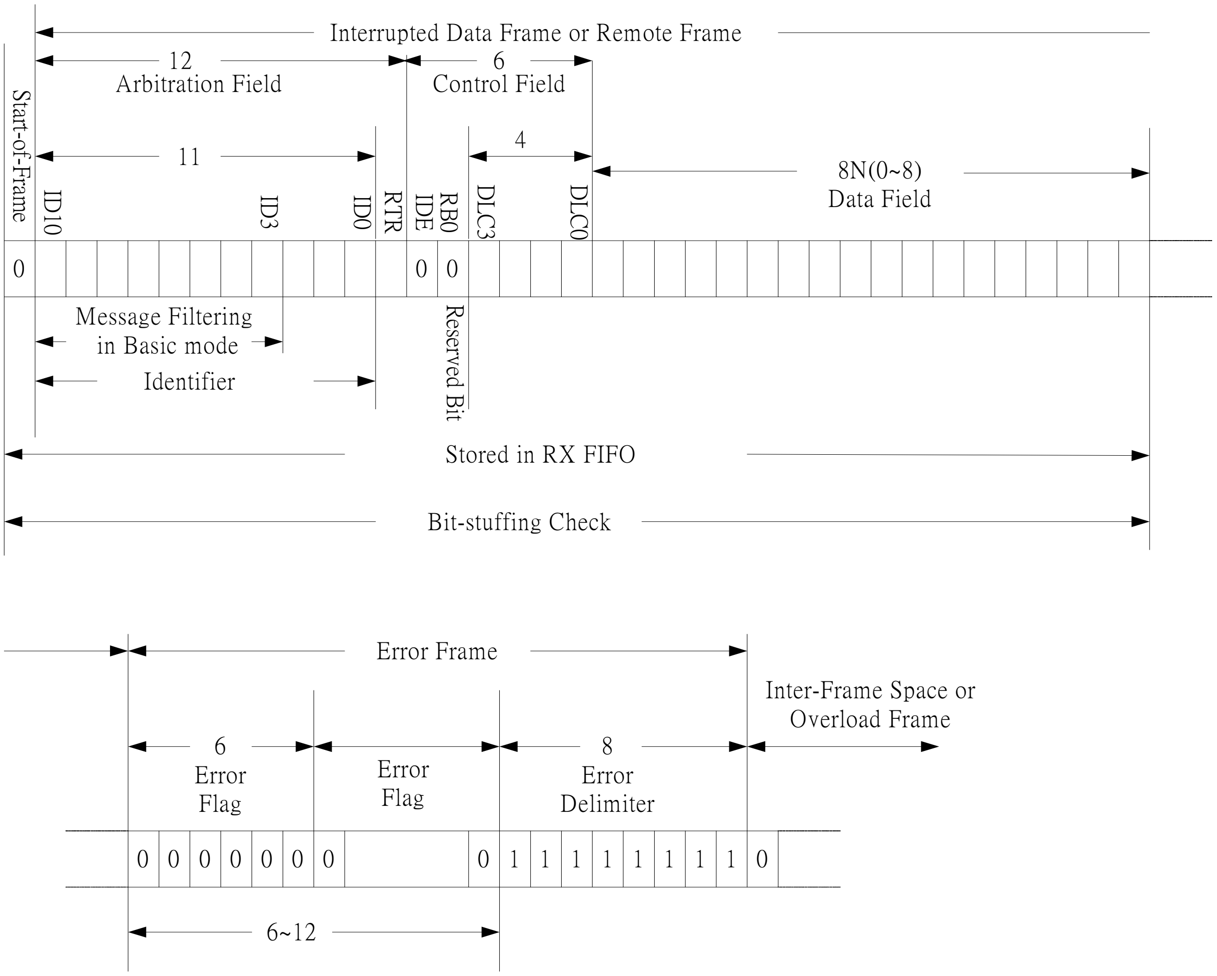


Figure 120: CAN Error Frame Format

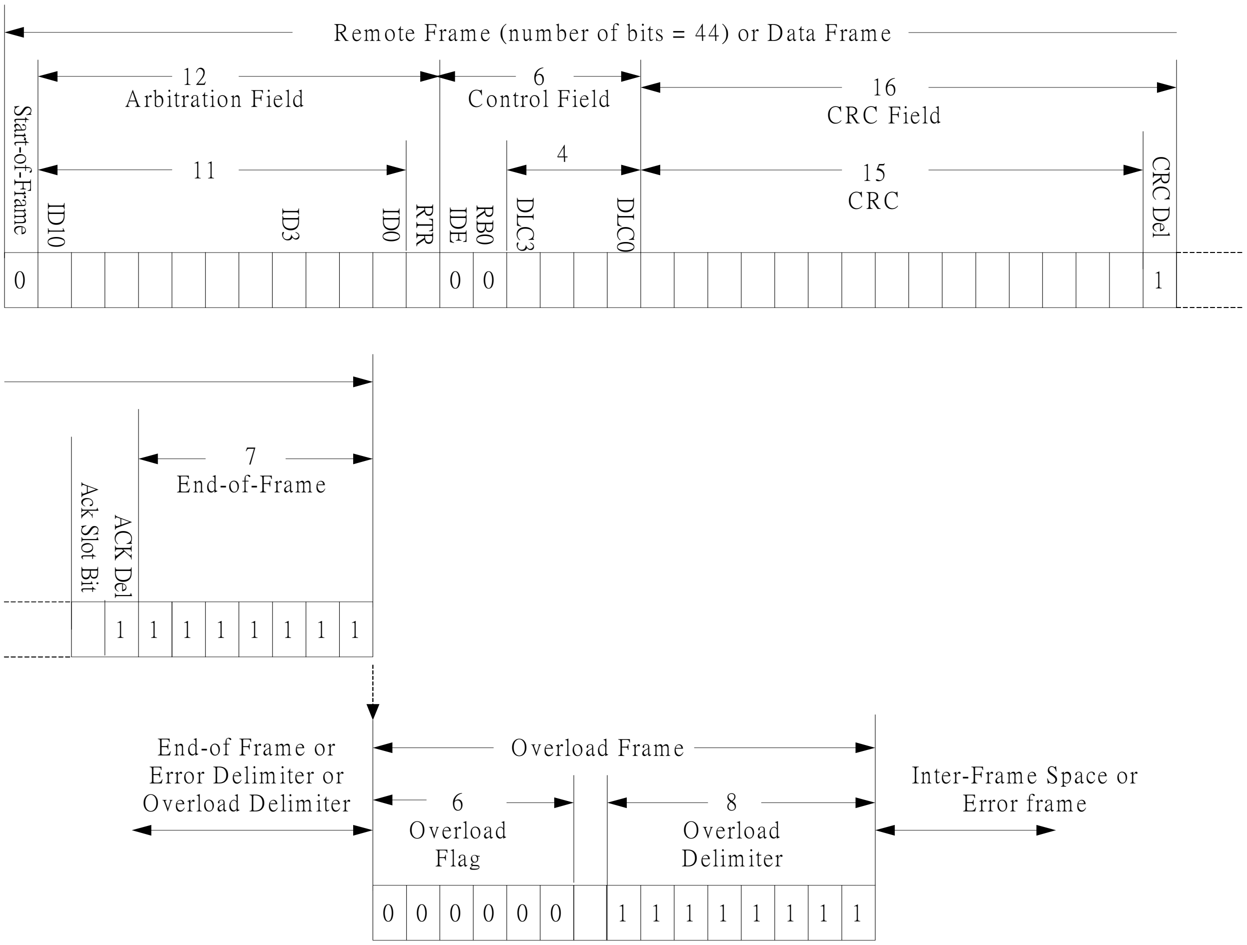


Figure 121: CAN Overload Frame Format

4.21.1 CAN SFR Register Map

Address	Name	Description
0xDE	CANCIR	CAN Command Index Register is used to indicate the address of CAN controller register.
0xDF	CANDR	CAN Data Register is used to read data from or write data to the specified CAN controller register.

Table 53: CAN Controller SFR Register Map

CAN Command Index Register (CANCIR, 0xDE)

Bit	7	6	5	4	3	2	1	0
Name	CANCIR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	CANCIR	WO	Indicate which of the CAN controller register as listed in Table 54 is to be accessed.

CAN Data Register (CANDR, 0xDF)

Bit	7	6	5	4	3	2	1	0
Name	CANDR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	CANDR	R/W	Data Register is used to write data to or read data from the CAN controller registers.

CAN Controller Register Indirect Access Method

Software shall use indirect access method through CANCIR and CANDR registers to do read and write access to the CAN controller registers as listed in Table 54 below.

Read a register from CAN controller:

Step 1. Write CANCIR: Software indicates the CAN controller register address to be accessed as the data and write it to the SFR register CANCIR.

Step 2. Read CANDR: Software then read SFR register CANDR. The data read from CANDR is the CAN controller register data indicated in step 1. Keep reading from CANDR if the CAN controller registers have more than one byte, in that case, the first byte being read back is LSB byte.

Write a register to CAN controller:

Step 1. Write CANDR: Software writes the data you want to write into CAN controller registers to the SFR register CANDR. Keep writing to CANDR if the CAN controller registers have more than one byte, in that case, the first byte being written should be LSB byte.

Step 2. Write CANCIR: After writing CAN controller register data to CANDR, software then indicates the target CAN controller register address as data and write it to CANCIR.

Note: While software is reading or writing CAN controller registers during a sequence of processes, software can abort that process by writing CANCIR with 0xFF.

CAN Controller Register Map

The CAN register map are different in Basic mode and Extended mode. For register access, two different modes have to be distinguished, i.e., Reset mode and Operating mode. The Reset mode is entered automatically after hardware reset.

Basic Mode						
Address	Name	Description	Operating Mode		Reset Mode	
			Read	Write	Read	Write
0x00	CANCR	CAN Control Register	CANCR	CANCR	CANCR	CANCR
0x01	CANCMR	CAN Command Register	0xFF	CANCMR	0xFF	-
0x02	CANSR	CAN Status Register	CANSR	-	CANSR	-
0x03	CANISR	CAN Interrupt Status Register	CANISR	-	CANISR	-
0x04	CANACR	CAN Acceptance Code Register	0xFF	-	CANACR	CANACR
0x05	CANAMR	CAN Acceptance Mask Register	0xFF	-	CANAMR	CANAMR
0x06	CANBTR0	CAN Bus Timing Register0	0xFF	-	CANBTR0	CANBTR0
0x07	CANBTR1	CAN Bus Timing Register1	0xFF	-	CANBTR1	CANBTR1
0x08~09		Reserved	-	-	-	-
0x0A	CANTFI	CAN Tx Frame Information Register (10 bytes)	CANTFI	CANTFI	-	-
0x0B~13		Reserved	-	-	-	-
0x14	CANRFI	CAN Rx Frame Information Register (10 bytes)	CANRFI	-	-	-
0x15~1E		Reserved	-	-	-	-
0x1F	CANMR	CAN Mode Register	CANMR	-	CANMR	CANMR

Table 54: CAN Controller Register Map in Basic Mode

Extended Mode						
Address	Name	Description	Operating Mode		Reset Mode	
			Read	Write	Read	Write
0x00	CANCR	CAN Control Register	CANCR	CANCR	CANCR	CANCR
0x01	CANCMR	CAN Command Register	0x00	CANCMR	0x00	-
0x02	CANSR	CAN Status Register	CANSR	-	CANSR	-
0x03	CANISR	CAN Interrupt Status Register	CANISR	-	CANISR	-
0x04	CANIER	CAN Interrupt Enable Register	CANIER	CANIER	CANIER	CANIER
0x05		Reserved	-	-	-	-
0x06	CANBTR0	CAN Bus Timing Register 0	CANBTR0	-	CANBTR0	CANBTR0
0x07	CANBTR1	CAN Bus Timing Register 1	CANBTR1	-	CANBTR1	CANBTR1
0x08~0A		Reserved	-	-	-	-
0x0B	CANALC	CAN Arbitration Lost Capture Register	CANALC	-	CANALC	-
0x0C	CANECC	CAN Error Code Capture Register	CANECC	-	CANECC	-
0x0D	CANEWL	CAN Error Warning Limit Register	CANEWL	-	CANEWL	CANEWL
0x0E	CANREC	CAN Rx Error Counter Register	CANREC	-	CANREC	CANREC
0x0F	CANTEC	CAN Tx Error Counter Register	CANTEC	-	CANTEC	CANTEC
0x10	CANACR CANRFI CANTFI	CAN Acceptance Code Register CAN Rx Frame Information Register (13 bytes) CAN Tx Frame Information Register (13 bytes)	CANRFI	CANTFI	CANACR	CANACR
0x11~13		Reserved	-	-	-	-
0x14	CANAMR	CAN Acceptance Mask Register	-	-	CANAMR	CANAMR
0x15~1C		Reserved	-	-	-	-
0x1D	CANRMC	CAN Rx Message Count Register	CANRMC	-	0x00	-
0x1E		Reserved	-	-	-	-
0x1F	CANMR	CAN Mode Register	CANMR	-	CANMR	CANMR

Table 55: CAN Controller Register Map in Extended Mode

4.21.2 Basic Mode Operation

CAN Control Register (CANCR, 0x00)

Bit	7	6	5	4	3	2	1	0
Name	Reserved			OVR_INTE	ERR_INTE	TX_INTE	RX_INTE	RESET
Reset Value	000			0	0	0	0	1

Bit	Name	Access	Description
0	RESET	R/W	CAN controller mode setting. 1: Configure CAN controller to Reset mode. Note that only in Reset mode can the CANACR, CANAMR, CANBTR0, CANBTR1 and CANMR be accessed by software. Also, when CAN controller transitions into Bus Off state, this bit will be set to “1” by hardware. Upon entering Reset mode, the current transmission/reception message will be aborted. 0: Configure CAN controller to Operating mode. When in Operating mode the CANACR, CANAMR, CANBTR0, CANBTR1can NOT be accessed by software and the CANMR is read-only by software.
1	RX_INTE	R/W	Receive Interrupt Enable. 1: When a message has been received without errors, the CAN controller will generate an interrupt on INT4 to the CPU. See CANISR.0 description. 0: Disable generating interrupt to CPU.
2	TX_INTE	R/W	Transmit Interrupt Enable. 1: When a message has been successfully transmitted or the transmit buffer is accessible again, the CAN controller will generate an interrupt on INT4 to the CPU. See CANISR.1 description. 0: Disable generating interrupt to CPU.
3	ERR_INTE	R/W	Error Interrupt Enable. 1: If the error status or bus status changed, the CAN controller will generate an interrupt on INT4 to the CPU. See CANISR.2 description. 0: Disable generating interrupt to CPU.
4	OVR_INTE	R/W	Overrun Interrupt Enable. 1: If data overrun condition occurred, the CAN controller will generate an interrupt on INT4 to the CPU. See CANISR.3 description. 0: Disable generating interrupt to CPU.
7:5	Reserved	-	

CAN Command Register (CANCMR, 0x01)

Bit	7	6	5	4	3	2	1	0
Name	Reserved				CDO	REL_BUF	ABT_TX	TX_REQ
Reset Value	0000				0	0	0	0

Bit	Name	Access	Description
0	TX_REQ	W1, see Note 1	Tx Request. 1: Setting “1” to request to send the message in CANTFI out to CANTX line. This bit will be cleared by hardware automatically. 0: No action.
1	ABT_TX	W1	Abort Tx. 1: If not already in progress, a pending transmission request is cancelled, then the CANISR.1 (TI) would be set to “1” if enabled CANCR.2. This bit will be cleared by hardware automatically. 0: Clear this bit.
2	REL_BUF	W1	Release message in Receive Buffer. 1: The received CAN messages are stored in RX FIFO, and software can read CANRFI register to retrieve the CAN messages in RX FIFO. After done reading CANRFI for one message, software then sets this bit to “1” to allow the next

			message in RX FIFO to become available in CANRFI and also to release buffer space of current message in RX FIFO. This bit will be cleared by hardware automatically. 0: No action.
3	CDO	W1	Clear Data Overrun. 1: Set "1" to clear the Overrun Status bit (CANSR.1). When a message is being received into RX FIFO during the time the RX FIFO has no enough space to store it, that message in RX FIFO can be corrupted. In that case, the overrun status flag associated with that message will be set and recorded in RX FIFO. Later when software retrieves that message from CANRFI, the ORS bit (CANSR.1) will reflect the overrun condition. Software must drop that message by setting CDO bit to clear ORS flag and setting REL_BUF bit (CANCMR.2) to read the next message in RX FIFO instead. This bit will be cleared by hardware automatically. 0: No action.
7:4	Reserved	-	

Note 1: Software reading CANCMR register will always return undefined value.

CAN Status Register (CANSR, 0x02)

Bit	7	6	5	4	3	2	1	0
Name	BS	ES	TS	RS	OF	TBS	ORS	RBS
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	RBS	RO	Receive Buffer Status. 1: One or more received CAN messages are available in the RX FIFO. After software reads one message from CANRFI register and sets REL_BUF (CANCMR.2) to release RX FIFO space, this bit would be cleared simultaneously. If RX FIFO still contains some messages, this bit would be set to "1" right away. 0: No message is available.
1	ORS	RO	Overrun Status. 1: The next message in CANRFI register, which has just been retrieved from RX FIFO was corrupted because there was not enough space for that massage in RX FIFO during receiving that message. 0: Absent.
2	TBS	RO	Transmit Buffer Status. 1: The software may write a message into transmit buffer, CANTFI register. 0: The software can't write transmit buffer, CANTFI register; a message is either waiting for transmission or is in the process of being transmitted.
3	OF	RO	Overload Frame received. 1: Reading "1" indicates receiving an Overload frame from CANRX line. In that case, the software may choose to wait for a predefined time period or receiving a Remote frame from the other end before requesting to transmit next message out onto CANTX line. 0: Absent. After receiving Overload frame to cause OF bit being set, upon receiving Data frame or Remote frame can clear this bit automatically.
4	RS	RO	Receive Data. 1: The CAN controller is busy receiving serial data from CANRX line. 0: The CAN controller is idle.
5	TD	RO	Transmit Data. 1: The CAN controller is busy transmitting serial data onto CANTX line. 0: The CAN controller is idle.
6	ES	RO	Error Status. 1: Either Rx or Tx error count or both has reached or exceeded the error warning limit of 96 (0x60). 0: Both Rx and Tx error count are below the error warning limit of 96.
7	BS	RO	Bus Status. 1: In Bus-Off state, where the CAN controller is not involved in bus activities. When

			transmission error counter exceeds the limit of 255, the CAN controller will set the RESET bit of CANCR.0 to logic 1 to enter Reset mode. It will stay in Reset mode until the software clears the RESET bit of CANCR.0. Once cleared the CAN controller will wait the minimum protocol-defined time (128 occurrences of the bus-free signal). After that the bus status (BS) bit is cleared (Bus-On), the error status (ES) bit is cleared to “0”, the error counters are reset, and the EI bit of CANISR.2 is set to “1” if ERR_INTE bit of CANCR.3 is enabled. 0: In Bus-On state, where the CAN controller is involved in bus activities.

CAN Interrupt Status Register (CANISR, 0x03)

Bit	7	6	5	4	3	2	1	0
Name	Reserved				DOI	EI	TI	RI
Reset Value	1110				0	0	0	0

Bit	**Name**	**Access**	**Description**
0	RI	CR	Receive Interrupt Status. 1: Reading “1” indicates that the RX FIFO is not empty (RBS bit of CANSR.0 changed 0-to-1), when RX_INTE bit of CANCR.1 is enabled. 0: This bit is cleared after software reads it.
1	TI	CR	Transmit Interrupt Status. 1: Reading “1” indicates that the transmission status had changed (TBS bit of CANSR.2 changed 0-to-1), when TX_INTE bit of CANCR.2 is enabled. 0: This bit is cleared after software reads it.
2	EI	CR	Error Interrupt Status. 1: Reading “1” indicates that either the error status (ES bit of CANSR.6 changed 0-to-1 or 1-to-0) or bus status (BS bit of CANSR.7 changed 0-to-1 or 1-to-0) or both has changed, when ERR_INTE bit of CANCR.3 is enabled. 0: This bit is cleared after software reads it.
3	DOI	CR	Data Overrun Interrupt Status. 1: Reading “1” indicates that the data overrun status has changed (ORS bit of CANSR.1 changed 0-to-1), when OVR_INTE bit of CANCR.4 is enabled. 0: This bit is cleared after software reads it.
7:4	Reserved	-	

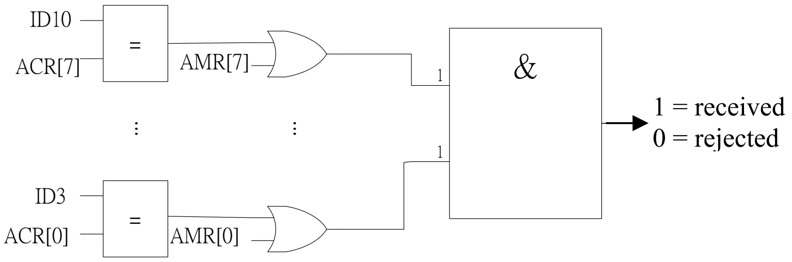
CAN Acceptance Code Register (CANACR, 0x04)

Bit	7	6	5	4	3	2	1	0
Name	ACR							
Reset Value	0x00							

Bit	**Name**	**Access**	**Description**
7:0	ACR	R/W	Acceptance Code Register. The CANACR and CANAMR registers together are used as acceptance filter to control receiving of CAN messages. See CANAMR register description. The write access to this register is possible only in Reset mode while in Operating mode this register is read-only and read value is always 0xFF.

CAN Acceptance Mask Register (CANAMR, 0x05)

Bit	7	6	5	4	3	2	1	0
Name	AMR							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	AMR	R/W	Acceptance Mask Register. The CANACR and CANAMR registers together are used as acceptance filter to control receiving of CAN messages. The bit patterns of messages to be received are defined within the Acceptance Code register. The corresponding Acceptance Mask register allows to define certain bit positions to be “don’t care”. The write access to this register is possible only in Reset mode while in Operating mode this register is read-only and read value is always 0xFF. The logic diagram for determining receive condition is as shown below.  ID10 ACR[7] = AMR[7] & 1 1 1 = received 0 = rejected ⋮ ⋮ ID3 ACR[0] = AMR[0] For example, 1. To enable receiving all messages with any Identifiers, software can configure AMR = 0xFF with any value of ACR. 2. To enable receiving messages of ID[10:0] = 1100_0001_xxx, where xxx is dont_care, software should configure ACR = 1100_0001 and AMR = 0000_0000. 3. To enable receiving messages of ID[10:0] = 1100_xx01_xxx, software should configure ACR = 1100_xx01 and AMR = 0000_1100.

CAN Bus Timing Register0 (CANBTR0, 0x06)

Bit	7	6	5	4	3	2	1	0
Name	SYNJW		BRP					
Reset Value	00		000000					

Bit	Name	Access	Description
5:0	BRP	R/W	Baud Rate Prescaler. $TQ = 2 * (BRP + 1) * T_{sys\text{-}clk}$ Time Quantum (TQ) is a fixed time unit used in each of the segments that make up a bit time of CAN serial data. The $T_{sys\text{-}clk}$ is cycle time of the operating system clock. The write access to this register is possible only in Reset mode while in Operating mode this register is read-only and read value is always “11_1111”.
7:6	SYNJW	R/W	Synchronization Jump Width. $t_{SYNJW} = (SYNJW + 1) * TQ$ To compensate for phase shifts between clocks oscillators of different bus controllers, and bus controller must re-synchronize on any relevant signal edge of the current transmission. See also CANBTR1 description. The write access to this register is possible only in Reset mode while in Operating mode this register is read-only and read value is always “11”.

CAN Bus Timing Register1 (CANBTR1, 0x07)

Bit	7	6	5	4	3	2	1	0
Name	TRS	TSEG2			TSEG1			
Reset Value	0	000			0000			

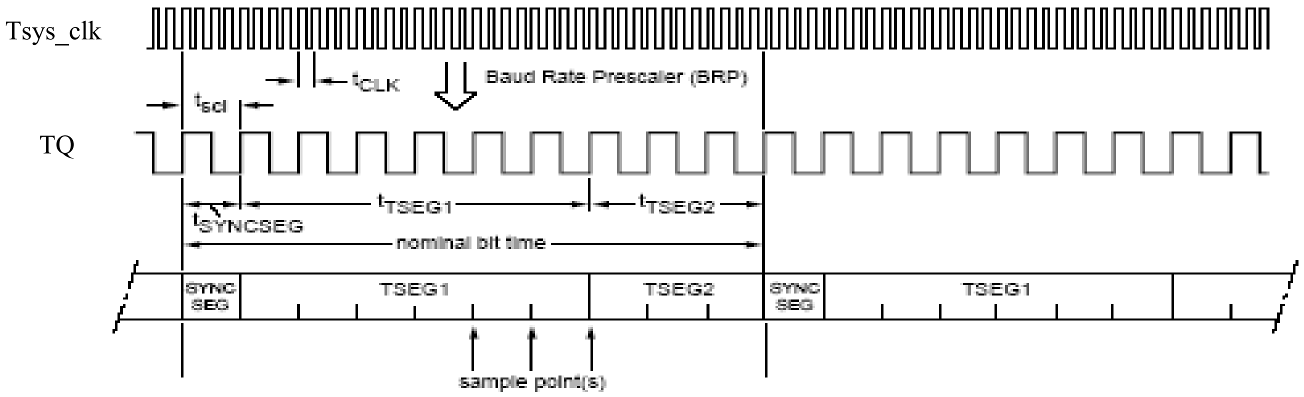
Bit	**Name**	**Access**	**Description**
3:0	TSEG1	R/W	Timing Segment 1. The TSEG1 defines the number of TQ for t_{TSEG1} in Figure 122 below. $t_{TSEG1} = (TSEG1 + 1) * TQ$ The TSEG1 and TSEG2 together define the length of the bit time of CAN serial data and the location of the sample point. The write access to this register is possible only in Reset mode while in Operating mode this register is read-only and read value is always “1111”.
6:4	TSEG2	R/W	Timing Segment 2. The TSEG2 defines the number of TQ for t_{TSEG2} in Figure 122 below. $t_{TSEG2} = (TSEG2 + 1) * TQ$ Note: $t_{TSEG2} > t_{SYNJW}$. The write access to this register is possible only in Reset mode while in Operating mode this register is read-only and read value is always “111”.
7	TRS	R/W	Triple Sampling. 1: The bus is sampled three times (normally for low/medium speed bus). 0: The bus is sampled once (normally for high speed bus). The write access to this register is possible only in Reset mode while in Operating mode this register is read-only and read value is always “1”.

Resynchronization is achieved by comparing the actual position of a recessive-to-dominant edge on the bus to the position of the expected edge (within the $t_{SYNCSEG}$) and adjusting the bit time as necessary. The phase error of a bit is given by the position of the edge in relation to the $t_{SYNCSEG}$, measured in TQ, and is defined as follows:

1. e = 0; the actual edge lies within the $t_{SYNCSEG}$.
   => Sample point = (TSEG1 + 1) * TQ

2. e > 0; the actual edge lies after $t_{SYNCSEG}$ and before the sample point, resynchronizing to a slower transmitter.
   => Sample point = (TSEG1 * TQ) + Tdrift; where Tdrift is equal to t_{SYNJW} if the actual edge lies beyond $t_{SYNJW,}$ or Tdrift is equal to the actual drift time if the actual edge lies before $t_{SYNJW.}$

3. e < 0; the actual edge lies after the sample point of the previous bit, resynchronizing to a faster transmitter. (TSEG2 is shortened)



Note: CAN baud rate = 1 / ((1 + (TSEG1 + 1) + (TSEG2 + 1)) * TQ)

Figure 122: Example Bit Time Structure of CAN Serial Data

CAN Tx Frame Information Register (CANTFI, 0x0A)

Bit	7	6	5	4	3	2	1	0
Name	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
	ID2	ID1	ID0	RTR	DLC			
	TD0							
	TD1							
	TD2							
	TD3							
	TD4							
	TD5							
	TD6							
	TD7							
Reset Value	0x0000_0000_0000_0000_0000							

Bit	Name	Access	Description
7:0 15:13	ID10:ID3 ID2:ID0	R/W	Message Identifier. The Identifier consists of 11 bits (ID10~ID0). The ID10 is the most significant bit, which is transmitted first on the bus. The Identifier acts as the message's name. It is used in a receiver for acceptance filtering and also determining the bus access priority during the arbitration process. The lower the binary value of the Identifier the higher the priority. This is due to a larger number of leading dominant bits during arbitration.
11:8	DLC	R/W	Data Length Code. The number of bytes in the data field of a message is coded by the data length code. For remote frame transmission, the data length code must be filled with "0000" with the RTR bit being set to "1". Say if the DLC is filled with "0001" with RTR bit being set to "1", this remote frame being transmitted will have 1 byte data frame, which should be avoided. The data length code must be specified correctly to avoid bus error if two CAN controller start a remote frame transmission with the same identifier simultaneously. For reasons of compatibility no data length code >8 should be used. If a value >8 is written, 8 bytes are transmitted in the data frame with the data length code specified in DLC.
12	RTR	R/W	Remote Transmission Request. 1: A Remote frame will be transmitted on the bus. This means that no data bytes are included within this frame. Also, DLC field must be filled with "0000". 0: A Data frame will be sent including the number of data bytes as specified by the data length code.
23:16 … 79:72	TD0 … TD7	R/W	Transmit Data. The number of transmitted data bytes is determined by the data length code. The first bit transmitted is the most significant bit of data byte TD0. The transmission data buffer has 8 bytes.

CAN Rx Frame Information Register (CANRFI, 0x14)

Bit	7	6	5	4	3	2	1	0
Name	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
	ID2	ID1	ID0	RTR	DLC			
	RD0							
	RD1							
	RD2							
	RD3							
	RD4							
	RD5							
	RD6							
	RD7							

Reset Value	0x0000_0000_0000_0000_0000

Bit	Name	Access	Description
7:0 15:13	ID10:ID3 ID2:ID0	RO	Identifier. See description in CANTFI.
11:8	DLC	RO	Data Length Code. See description in CANTFI. When received one message with DLC >8, only 8 bytes of data are stored in to RX FIFO.
12	RTR	RO	Remote Transmission Request. See description in CANTFI.
23:16 … 79:72	RD0 … RD7	RO	Receive Data. The received data of the CAN message that has passed the acceptance test defined in CANACR and CANAMR registers. The number of received data bytes is determined by the data length code. The first bit received is the most significant bit of data byte RD0.

CAN Mode Register (CANMR, 0x1F)

Bit	7	6	5	4	3	2	1	0
Name	EXTM	Reserved						
Reset Value	0	000_0000						

Bit	Name	Access	Description
6:0	Reserved	-	
7	EXTM	R/W	Extended Mode. 1: Configure CAN controller to operate in Extended mode. 0: Configure CAN controller to operate in Basic mode. The write access to this register is possible only in Reset mode while in Operating mode this register is read-only.

Example Programming Procedure in CAN Controller Basic Mode

Configure CAN controller to Basic mode, Baud rate = 1Mbps, and the operating system clock is 100Mhz. Transmit one message with ID = 0x143 and data length = 3 bytes. The software is running in polling mode.

1. Write 0x01 to CANCR register to enter Reset mode.
2. Write 0xC4 to CANBTR0 register.
3. Write 0x34 to CANBTR1 register.
4. Write 0x28 to CANACR register.
5. Write 0x00 to CANAMR register.
6. Write 0x00 to CANCR register to exit Reset mode.
7. Write 0x0000_0000_0012_8001_6328 to CANTFI register, where the most significant 5 bytes of “00_0000_0000” are just dummy data for writing CANTFI register.
8. Write 0x01 to CANCMR register to start transmitting the message.
9. Read CANSR register for TBS bit changing 0-to-1 for transmit complete indication.

For receiving one message with ID = 0x143.

1. Write CANACR, CANAMR with proper value, such as example value in above step 4 and 5.
2. Continue polling CANSR register for RBS bit changing 0-to-1 for RX FIFO receiving one message indication. Also check the ORS bit.
3. Read the received message from CANRFI register up to 10 bytes of data.
4. Write 0x04 to CANCMR register to release buffer space in RX FIFO.

4.21.3 Extended Mode Operation

For CAN Extended mode register map table, please refer to Table 55.

CAN Control Register (CANCR, 0x00)

Bit	7	6	5	4	3	2	1	0
Name	Reserved				AF	ST	LO	RESET
Reset Value	0000				0	0	0	1

Bit	Name	Access	Description
0	RESET	R/W	CAN controller mode setting. 1: Configure CAN controller to Reset mode. Note that only in Reset mode can the CANACR, CANAMR, CANBTR0, CANBTR1, CANEWL, CANREC, CANTEC, and CANMR be accessed by software. Also, when CAN controller transitions into Bus Off state, this bit will be set to "1" by hardware. Upon entering Reset mode, the current transmission/reception message will be aborted. 0: Configure CAN controller to Operating mode. When in Operating mode the CANACR, CANAMR, CANBTR0, CANBTR1, CANEWL, CANREC, CANTEC, can NOT be accessed by software and the CANMR is read-only by software.
1	LO	R/W	Listen Only mode. 1: In Listen-Only mode the CAN controller would give no acknowledge to the CAN bus, even if a message is received successfully. The error counters are stopped at current value. 0: Normal mode. The write access to this bit is only possible after entering Reset mode.
2	ST	R/W	Self Test mode. 1: In self test mode, a full node test is possible without any other active node on the bus using the self reception request command; the CAN controller will perform a successful transmission, even if there is no acknowledge received. 0: In normal mode, an acknowledge is required for successful transmission. The write access to this bit is only possible after entering Reset mode.
3	AF	R/W	Acceptance Filter. 1: The dual acceptance filter option is enabled in CANACR and CANAMR registers. In this mode, two short filters can be defined. A received message is compared with both filters to decide, whether this message should be stored into the RX FIFO or not. If at least one of the filters signals an acceptance, the received message becomes valid. 0: The single acceptance filter option is enabled (one filter with the length of 32 bits is active) in CANACR and CANAMR registers. The write access to this bit is only possible after entering Reset mode.
7:4	Reserved	-	

CAN Command Register (CANCMR, 0x01)

Bit	7	6	5	4	3	2	1	0
Name	Reserved			SRR	CDO	REL_BUF	ABT_TX	TX_REQ
Reset Value	000			0	0	0	0	0

Bit	Name	Access	Description
0	TX_REQ	W1	Tx Request. 1: Setting "1" to request to send the message in CANTFI out to CANTX line. This bit will be cleared by hardware automatically. 0: No action.
1	ABT_TX	W1	Abort Tx. 1: If not already in progress, a pending transmission request is cancelled, then the CANISR.1 (TI) would be set to "1" if enabled CANIER.1. This bit will be cleared by hardware automatically. 0: Clear this bit. Example usage: 1. Setting the ABT_TX and TX_REQ bits simultaneously results in sending the transmit message once. No re-transmission will be performed in the event of an error or arbitration lost (single-shot transmission). 2. Setting the ABT_TX and SRR bits simultaneously results in sending the transmit message once using the self reception feature. No re-transmission will be performed in the event of an error or arbitration lost. 3. Setting the ABT_TX, TX_REQ, and SRR bits simultaneously results in sending the transmit message once as described for ABT_TX and TX_REQ.
2	REL_BUF	W1	Release message in Receive Buffer. 1: The received CAN messages are stored in RX FIFO, and software can read CANRFI register to retrieve the CAN messages in RX FIFO. After done reading CANRFI for one message, software then sets this bit to "1" to allow the next message in RX FIFO to become available in CANRFI and also to release buffer space of current message in RX FIFO. This bit will be cleared by hardware automatically. 0: No action.
3	CDO	W1	Clear Data Overrun. 1: Set "1" to clear the Overrun Status bit (CANSR.1). When a message is being received into RX FIFO during the time the RX FIFO has no enough space to store it, that message in RX FIFO can be corrupted. In that case, the overrun status flag associated with that message will be set and recorded in RX FIFO. Later when software retrieves that message from CANRFI, the ORS bit (CANSR.1) will reflect the overrun condition. Software must drop that message by setting CDO bit to clear ORS flag and setting REL_BUF bit (CANCMR.2) to read the next message in RX FIFO instead. This bit will be cleared by hardware automatically. 0: No action.
4	SRR	W1	Self Rx Request. During self test mode when ST bit (CANCE.2) = 1, this bit is used to request to send out a message in CANTFI register. 1: A message will be transmitted and received simultaneously. 0: Absent. Example usage: 1. Setting the ST (CANCR.2) and SRR bits simultaneously the message in CANTFI will be sent and this message will be received into RX FIFO if the message identifier passes acceptance filter test. During this process, if there is arbitration loss or bus error, the CAN controller will retransmit this message automatically but it will not be received into RX FIFO anymore regardless of message identifier value. This is due to that the SRR bit is cleared after the first transmission of this message. 2. Setting SRR bit when ST bit = 0, this is the same result as setting TX_REQ bit

			alone where the message in CANTFI register will be sent but will not be received into RX FIFO. Note that this is not recommended to replace TX_REQ bit for normal transmit request. 3. If SRR and TX_REQ bits are set simultaneously when ST bit = 1, the result will be the same as setting only TX_REQ bit.
7:5	Reserved	-	

CAN Status Register (CANSR, 0x02)

Bit	7	6	5	4	3	2	1	0
Name	BS	ES	TS	RS	OF	TBS	ORS	RBS
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	RBS	RO	Receive Buffer Status. 1: One or more received CAN messages are available in the RX FIFO. After software reads one message from CANRFI register and sets REL_BUF (CANCMR.2) to release RX FIFO space, this bit would be cleared simultaneously. If RX FIFO still contains some messages, this bit would be set to “1” right away. 0: No message is available.
1	ORS	RO	Overrun Status. 1: The next message in CANRFI register, which has just been retrieved from RX FIFO was corrupted because there was not enough space for that massage in RX FIFO during receiving that message. 0: Absent.
2	TBS	RO	Transmit Buffer Status. 1: The software may write a message into transmit buffer, CANTFI register. 0: The software can’t write transmit buffer, CANTFI register; a message is either waiting for transmission or is in the process of being transmitted.
3	OF	RO	Overload Frame received. 1: Reading “1” indicates receiving an Overload frame from CANRX line. In that case, the software may choose to wait for a predefined time period or receiving a Remote frame from the other end before requesting to transmit next message out onto CANTX line. 0: Absent. After receiving Overload frame to cause OF bit being set, upon receiving Data frame or Remote frame can clear this bit automatically.
4	RD	RO	Receive Data. 1: The CAN controller is busy receiving serial data from CANRX line. 0: The CAN controller is idle.
5	TD	RO	Transmit Data. 1: The CAN controller is busy transmitting serial data onto CANTX line. 0: The CAN controller is idle.
6	ES	RO	Error Status. 1: Either Rx or Tx error count or both has reached or exceeded the error warning limit defined in CANEWL register. 0: Both Rx and Tx error count are below the error warning limit defined in CANEWL.
7	BS	RO	Bus Status. 1: In Bus-Off state, where the CAN controller is not involved in bus activities. When transmission error counter exceeds the limit of 255, the CAN controller will set the RESET bit of CANCR.0 to logic 1 to enter Reset mode. It will stay in Reset mode until the software clears the RESET bit of CANCR.0. Once cleared the CAN controller will wait the minimum protocol-defined time (128 occurrences of the bus-free signal). After that the bus status (BS) bit is cleared (Bus-On), the error status (ES) bit is cleared to “0”, the error counters are reset, and the EI bit of CANISR.2 is set to “1” if EIE bit of CANIER.2 is enabled. 0: In Bus-On state, where the CAN controller is involved in bus activities.

CAN Interrupt Status Register (CANISR, 0x03)

Bit	7	6	5	4	3	2	1	0
Name	BEI	ALI	EPI	Reserved	DOI	EI	TI	RI
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	RI	CR	Receive Interrupt Status. 1: Reading “1” indicates that the RX FIFO is not empty (RBS bit of CANSR.0 changed 0-to-1), when RIE bit of CANIER.0 is enabled. 0: This bit is cleared after software reads it.
1	TI	CR	Transmit Interrupt Status. 1: Reading “1” indicates that the transmission status had changed (TBS bit of CANSR.2 changed 0-to-1), when TIE bit of CANIER.1 is enabled. 0: This bit is cleared after software reads it.
2	EI	CR	Error Interrupt Status. 1: Reading “1” indicates that either the error status (ES bit of CANSR.6 changed 0-to-1 or 1-to-0) or bus status (BS bit of CANSR.7 changed 0-to-1 or 1-to-0) or both has changed, when EIE bit of CANIER.2 is enabled. 0: This bit is cleared after software reads it.
3	DOI	CR	Data Overrun Interrupt Status. 1: Reading “1” indicates that the data overrun status has changed (ORS bit of CANSR.1 changed 0-to-1), when DOIE bit of CANIER.3 is enabled. 0: This bit is cleared after software reads it.
4	Reserved	-	
5	EPI	CR	Error Passive Interrupt. 1: This bit will be set whenever the CAN controller has reached the error passive status (at least one error counter exceeds the protocol-defined level of 127) or the CAN controller is in the error passive status and enters the error active status again, when EPIE bit of CANIER.5 is enabled. 0: This bit is cleared after software reads it.
6	ALI	CR	Arbitration Lost Interrupt. 1: This bit will be set if the CAN controller lost the arbitration and become a receiver, when ALIE bit of CANIER.6 is enabled. 0: This bit is cleared after software reads it.
7	BEI	CR	Bus Error Interrupt. 1: When the CAN controller detects error frame on the CAN bus, when BEIE bit of CANIER.7 is enabled. 0: No error frame detected.

CAN Interrupt Enable Register (CANIER, 0x04)

Bit	7	6	5	4	3	2	1	0
Name	BEIE	ALIE	EPIE	Reserved	DOIE	EIE	TIE	RIE
Reset Value	0	0	0	0	0	0	0	0

Bit	Name	Access	Description
0	RIE	R/W	Receive Interrupt Enable. 1: When a message has been received without errors, the CAN controller will generate an interrupt on INT4 to the CPU. See CANISR.0 description. 0: Disable generating interrupt to CPU.
1	TIE	R/W	Transmit Interrupt Enable. 1: When a message has been successfully transmitted or the transmit buffer is accessible again, the CAN controller will generate an interrupt on INT4 to the CPU. See CANISR.1 description. 0: Disable generating interrupt to CPU.
2	EIE	R/W	Error Interrupt Enable. 1: If the error status or bus status changed, the CAN controller will generate an interrupt on INT4 to the CPU. See CANISR.2 description. 0: Disable generating interrupt to CPU.
3	DOIE	R/W	Data Overrun Interrupt Enable. 1: If data overrun condition occurred, the CAN controller will generate an interrupt on INT4 to the CPU. See CANISR.3 description. 0: Disable generating interrupt to CPU.
4	Reserved	-	
5	EPIE	R/W	Error Passive Interrupt Enable 1: Enable generating an interrupt on INT4 to the CPU when the CAN controller changes from error active to error passive or vice versa. See CANISR.5 description. 0: Disable generating interrupt to CPU.
6	ALIE	R/W	Arbitration Lost Interrupt Enable. 1: Enable generating an interrupt on INT4 to the CPU when the CAN controller lost arbitration. See CANISR.6 description. 0: Disable generating interrupt to CPU.
7	BEIE	R/W	Bus Error Interrupt Enable. 1: Enable generating an interrupt on INT4 to the CPU when the CAN controller receives error frame. See CANISR.7 description. 0: Disable generating interrupt to CPU.

CAN Bus Timing Register0 (CANBTR0, 0x06)

For description, please refer to the same CANBTR0 register in Basic mode in section 4.21.2. In Operating mode this register is read-only and read value is always the current actual value of the register.

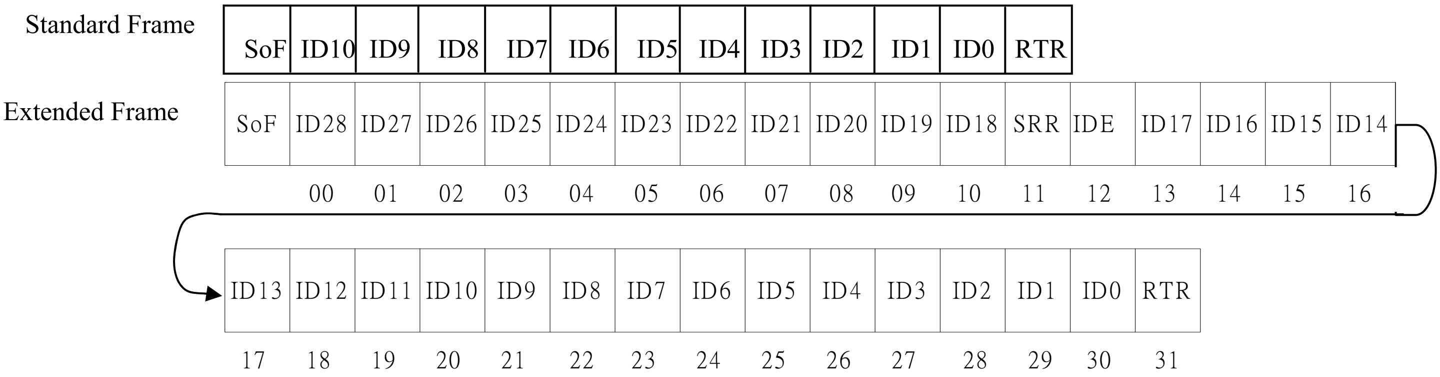
CAN Bus Timing Register1 (CANBTR1, 0x07)

For description, please refer to the CANBTR1 register in Basic mode in section 4.21.2. In Operating mode this register is read-only and read value is always the current actual value of the register.

CAN Arbitration Lost Capture Register (CANALC, 0x0B)

Bit	7	6	5	4	3	2	1	0
Name	Reserved			ALC				
Reset Value	000			00000				

Bit	Name	Access	Description
4:0	ALC	RO	Arbitration Lost Capture. On arbitration lost, the current bit position of the received bit stream from CANRX that is different from the transmitted bit stream is captured into the CANALC register. The content in this register is fixed until the software has read this register once or the software sets the TX_REQ bit (CANCMR.0) again or sets the SRR bit (CANCMR.4). The capture mechanism is then activated again. A new arbitration lost interrupt is not possible until the CANALC register is read out once.
7:5	Reserved	-	



Standard Frame

SoF	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR

Extended Frame

SoF	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	SRR	IDE	ID17	ID16	ID15	ID14
	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16

ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Figure 123: Arbitration Lost Capture Register Content Interpretation

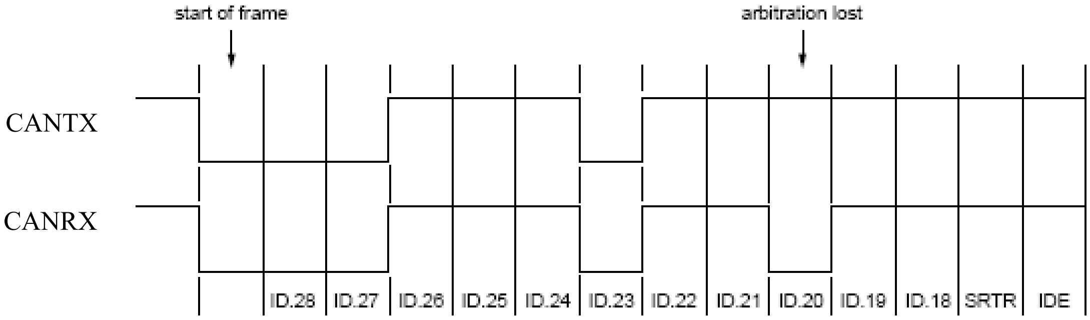


Figure 124: Arbitration Lost Example (Resultant ALC = 0x08)

CAN Error Code Capture Register (CANECC, 0x0C)

<table>
<tr><th>Bit</th><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><th>Name</th><td colspan="2">CCT</td><td>CCD</td><td colspan="5">CCS</td></tr>
<tr><th>Reset Value</th><td colspan="2">0</td><td>0</td><td colspan="5">00000</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Name</th><th>Access</th><th>Description</th></tr>
<tr><td>4:0</td><td>CCS</td><td>RO</td><td>Error Capture Code Segment.
This register contains information about the type and location of errors on the CAN bus. When a bus error occurs, the current position of the bit stream is captured into the CANECC register. The content in this register is fixed until the software has read out its content once. The capture mechanism is then activated again.
<table>
<tr><th>CCS[4:0]</th><th>Description</th></tr>
<tr><td>0_0011</td><td>Start of frame</td></tr>
<tr><td>0_0010</td><td>ID.28 to ID.21</td></tr>
<tr><td>0_0110</td><td>ID.20 to ID.18</td></tr>
<tr><td>0_0100</td><td>SRR bit</td></tr>
<tr><td>0_0101</td><td>IDE bit</td></tr>
<tr><td>0_0111</td><td>ID.17 to ID.13</td></tr>
<tr><td>0_1111</td><td>ID.12 to ID.5</td></tr>
<tr><td>0_1110</td><td>ID.4 to ID.0</td></tr>
<tr><td>0_1100</td><td>RTR bit</td></tr>
<tr><td>0_1101</td><td>RB1</td></tr>
<tr><td>0_1001</td><td>RB0</td></tr>
<tr><td>0_1011</td><td>Data length code</td></tr>
<tr><td>0_1010</td><td>Data field</td></tr>
<tr><td>0_1000</td><td>CRC field</td></tr>
<tr><td>1_1000</td><td>CRC delimiter</td></tr>
<tr><td>1_1001</td><td>Acknowledge slot</td></tr>
<tr><td>1_1011</td><td>Acknowledge delimiter</td></tr>
<tr><td>1_1010</td><td>End of frame</td></tr>
<tr><td>1_0010</td><td>Intermission</td></tr>
<tr><td>1_0001</td><td>Active error frame</td></tr>
<tr><td>1_0110</td><td>Passive error frame</td></tr>
<tr><td>1_0011</td><td>Tolerate dominant bit</td></tr>
<tr><td>1_0111</td><td>Error delimiter</td></tr>
<tr><td>1_1100</td><td>Overload flag</td></tr>
</table>
In reset mode, read value is always ”11111”.</td></tr>
<tr><td>5</td><td>CCD</td><td>RO</td><td>Error Capture Code Direction.
1: Rx error occurred during reception.
0: Tx error occurred during transmission.
In reset mode, read value is always “1”.</td></tr>
<tr><td>7:6</td><td>CCT</td><td>RO</td><td>Error Capture Code Type.
<table>
<tr><th>CCT</th><th>Description</th></tr>
<tr><td>00</td><td>Bit error</td></tr>
<tr><td>01</td><td>Form error</td></tr>
<tr><td>10</td><td>Stuff error</td></tr>
<tr><td>11</td><td>Other error</td></tr>
</table>
When more than one error types occurred in same message. The CCT only records the error type that occurred first. In reset mode, read value is always “11”.</td></tr>
</table>

CAN Error Warning Limit Register (CANEWL, 0x0D)

Bit	7	6	5	4	3	2	1	0
Name	EWL							
Reset Value	0x60							

Bit	Name	Access	Description
7:0	EWL	R/W	Error Warning Limit. This register defines the error warning limit of Rx error count and Tx error count to cause ES bit of CANSR.6 to be asserted. The write access to this register is possible only in Reset mode while in Operating mode it is read-only.

CAN Rx Error Counter Register (CANREC, 0x0E)

Bit	7	6	5	4	3	2	1	0
Name	REC							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	REC	R/W	Rx Error Counter. This register reflects the current value of the Rx error counter. After hardware reset or entering Reset mode or entering the Bus-Off state, this register is cleared to 0x00. The write access to this register is possible only in Reset mode while in Operating mode this register is read-only. During the time Bus-Off is valid, writing to this register has no effect. An error status change, an error warning or an error passive interrupt forced by the new register content will not occur, until the reset mode is cancelled again.

CAN Tx Error Counter Register (CANTEC, 0x0F)

Bit	7	6	5	4	3	2	1	0
Name	TEC							
Reset Value	0x00							

Bit	Name	Access	Description
7:0	TEC	R/W	Tx Error Counter. This register reflects the current value of the Tx error counter. The write access to this register is possible only in Reset mode while in Operating mode this register is read-only. After hardware reset, this register is cleared to 0x00. If a Bus-Off state occurs, this register is set to 128 (0x80) to count the minimum protocol-defined time (128 occurrences of the bus-free signal). Reading this register during this time gives information about the status of the Bus-Off recovery. If Bus-Off state is active, a write access to CANTEC in the range from 0 to 254 (0xFE) clears the Bus-Off state. After the Reset mode has been cleared, the CAN controller will wait for one occurrence of 11 consecutive Recessive bits (bus-free). Writing 255 (0xFF) to CANTEC allows initiating a software-driven Bus-Off state. An error or bus status change in CANSR, an error warning or an error passive interrupt of CANISR forced by the new register content will not occur until the Reset mode is cancelled again. After leaving the Reset mode, the new Tx counter content is interpreted and the Bus-Off state is performed in the same way, as if it was forced by a bus error event. That means, that the Reset mode is entered again, the Tx error counter is initialized to 128 (0x80), the Rx error counter is cleared and all concerned status and interrupt register bits are set. Clearing of Reset mode now will perform the protocol-defined Bus-Off recovery sequence (waiting for 128 occurrences of the bus-free signal). If the Reset mode is entered again before the end of Bus-Off recovery (CANTEC > 0), Bus-Off keeps active and CANTEC is frozen.

CAN Acceptance Code Register (CANACR, 0x10) ← **used in Reset mode**

Bit	7	6	5	4	3	2	1	0
Name	ACR0							
	ACR1							
	ACR2							
	ACR3							
Reset Value	0x0000_0000							

Bit	Name	Access	Description
7:0 … 31:24	ACR0 … ACR3	R/W	Acceptance Code Register. The CANACR and CANAMR registers together are used as acceptance filter to control receiving of CAN messages. See CANAMR register description. The write and read access to this register is possible only in Reset mode.

CAN Acceptance Mask Register (CANAMR, 0x14) ← **used in Reset mode**

Bit	7	6	5	4	3	2	1	0
Name	AMR0							
	AMR1							
	AMR2							
	AMR3							
Reset Value	0x0000_0000							

Bit	Name	Access	Description
7:0 … 31:24	AMR0 … AMR3	R/W	Acceptance Mask Register. The CANACR and CANAMR registers together are used as acceptance filter to control receiving of CAN messages. The bit patterns of messages to be received are defined within the Acceptance Code register. The corresponding Acceptance Mask register allows to define certain bit positions to be "don't care". The write and read access to this register is possible only in Reset mode. The acceptance filter supports two modes: Single Filter mode and Dual Filter mode, determined by AF bit of CANCR.3. The logic diagrams for determining receive condition are as shown in Figure 125 to Figure 128.

Figure 125 below shows the Single Filter mode for receiving Standard Frame Format message. Note that the ACR1[3:0] and AMR1[3:0] are not used and should be set to "don't-care" by setting AMR1[3:0] = "1111". When a Standard Frame Format message is received, the complete standard Identifier including the RTR bit and the first two data bytes are used for acceptance filtering. Messages may also be accepted if there are no data bytes existing due to a set RTR bit or if there is none or only one data byte because of the corresponding data length code. For a successful reception of a message, all single bit comparisons have to signal acceptance.

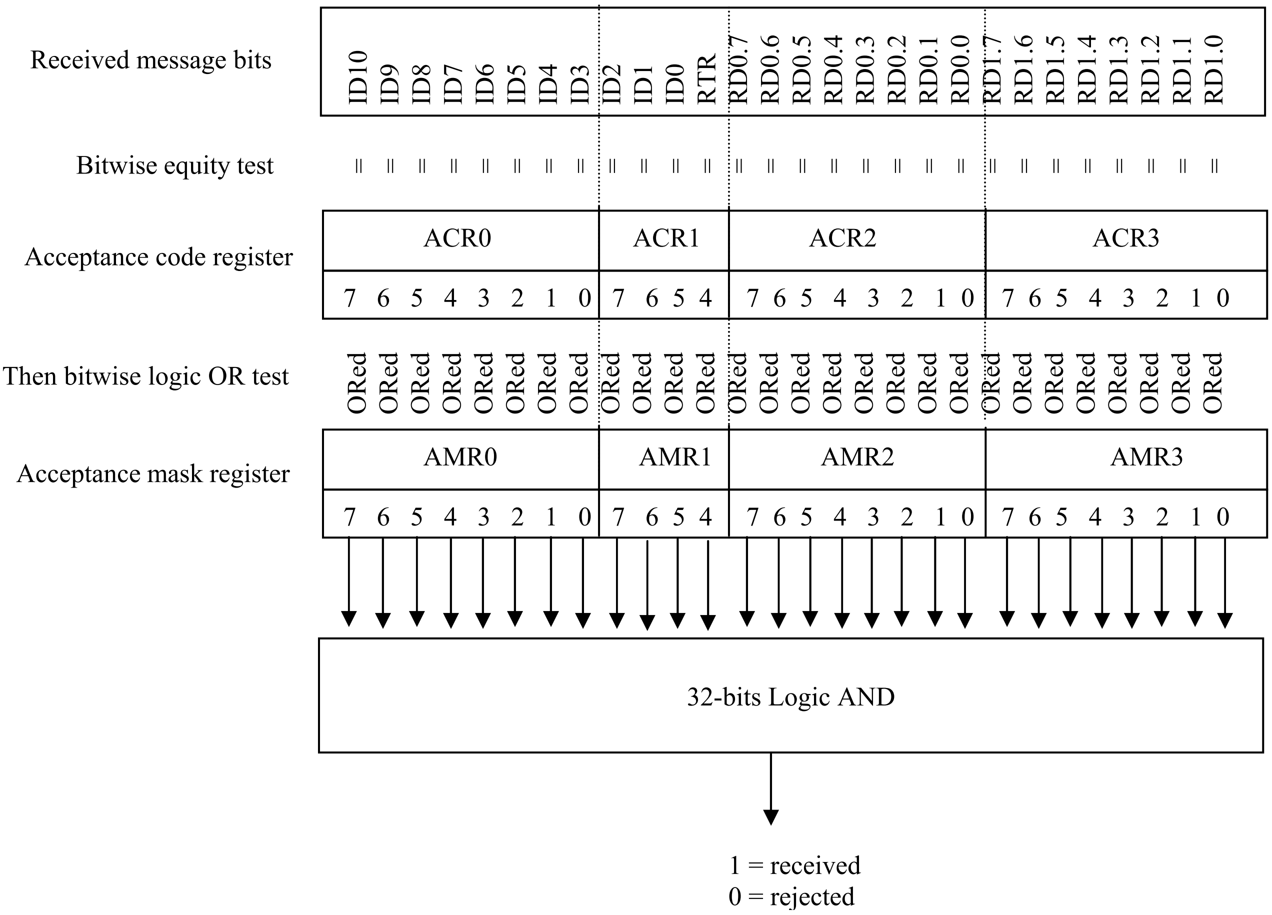


Figure 125: Single Filter Mode For Receiving Standard Frame Format

Figure 126 below shows the Single Filter mode for receiving Extended Frame Format message. Note that the AMR3[1:0] and ACR3[1:0] are not used and should be set to "don't-care" by setting AMR3[1:0] = "11". When an Extended Frame Format message is received, the complete extended Identifier including the RTR bit is used for acceptance filtering. For a successful reception of a message, all single bit comparisons have to signal acceptance.

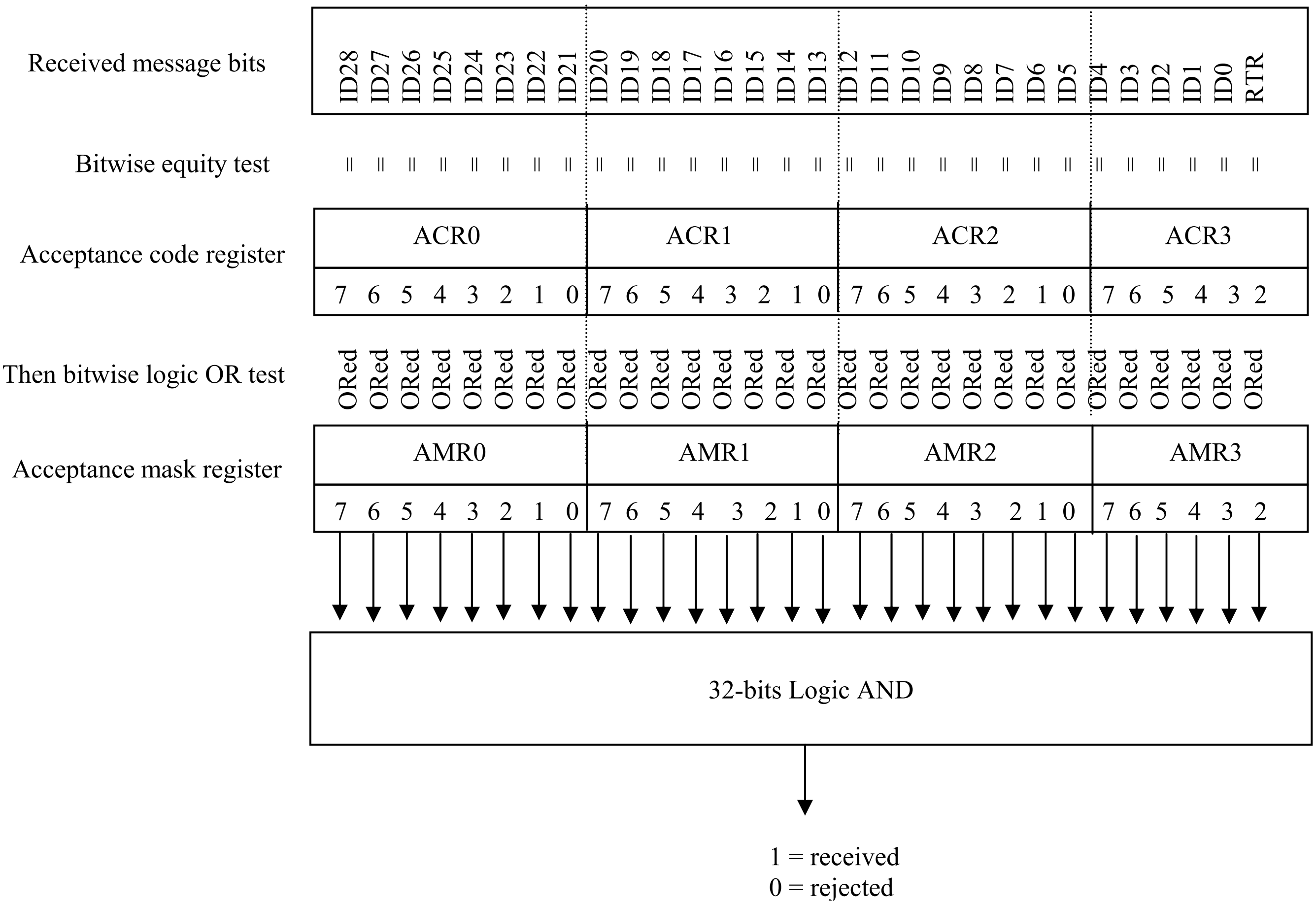


Figure 126: Single Filter Mode For Receiving Extended Frame Format

Figure 127 below shows the Dual Filter mode for receiving Standard Frame Format message. When a Standard Frame Format message is received, the two defined filters are looking differently. The first filter compares the complete standard Identifier including the RTR bit and the first data byte of the message. The second filter just compares the complete standard Identifier including the RTR bit. For a successful reception of a message, all single bit comparisons of at least one filter have to signal acceptance. For first filter, in case of a set RTR bit or a data length code of "0000", no data byte is existing. Nevertheless a message may pass first filter, if the first part up to the RTR bit signals acceptance. If no data byte filtering is required for first filter, the AMR1[3:0] and AMR3[3:0] have to be set to "1111" (don't care). Then both filters are working identically using the standard Identifier range including the RTR bit.

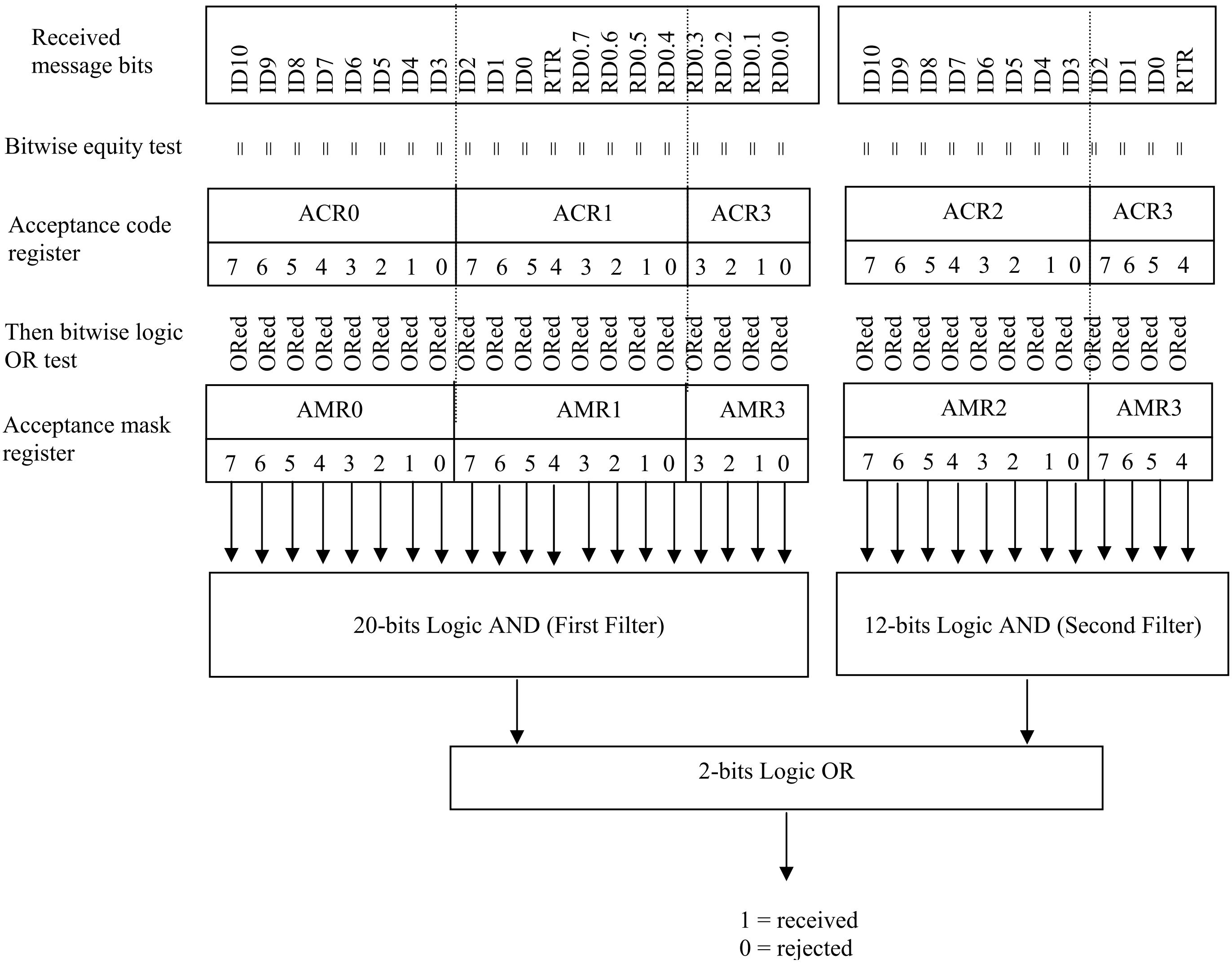


Figure 127: Dual Filter Mode For Receiving Standard Frame Format

Figure 128 below shows the Dual Filter mode for receiving Extended Frame Format message. When an Extended Frame Format message is received, the two defined filters are looking identically. Both filters are comparing the first two bytes of the extended Identifier range only. For a successful reception of a message, all single bit comparisons of at least one filter have to indicate acceptance.

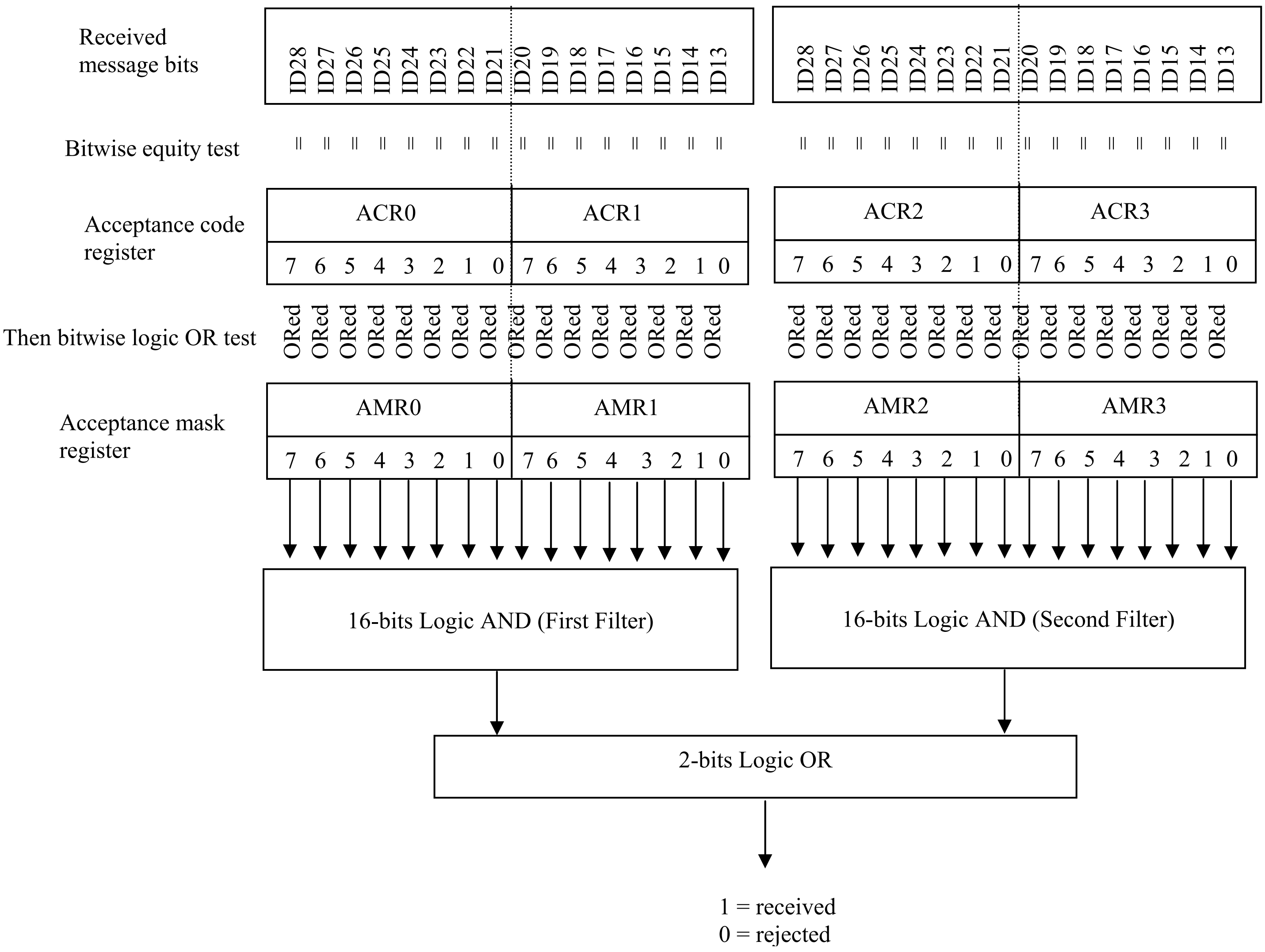

Received message bits
ID28 ID27 ID26 ID25 ID24 ID23 ID22 ID21 ID20 ID19 ID18 ID17 ID16 ID15 ID14 ID13
ID28 ID27 ID26 ID25 ID24 ID23 ID22 ID21 ID20 ID19 ID18 ID17 ID16 ID15 ID14 ID13
Bitwise equity test
Acceptance code register
ACR0
ACR1
ACR2
ACR3
7 6 5 4 3 2 1 0
Then bitwise logic OR test
ORed
Acceptance mask register
AMR0
AMR1
AMR2
AMR3
16-bits Logic AND (First Filter)
16-bits Logic AND (Second Filter)
2-bits Logic OR
1 = received
0 = rejected

Figure 128: Dual Filter Mode For Receiving Extended Frame Format

CAN RX/TX Frame Information Register (CANRFI/CANTFI, 0x10) ← **used in Operating mode**

The read access to this register always returns CANRFI register value, which reflects the received message in RX FIFO. The write access to this register always writes to CANTFI register.

✧ Rx and Tx Frame Information with Standard Frame Format (SFF)

Bit	7	6	5	4	3	2	1	0
Name	IDE	RTR	Reserved		DLC			
	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
	ID20	ID19	ID18	Reserved				
	SFFR/TD0							
	SFFR/TD1							
	SFFR/TD2							
	SFFR/TD3							
	SFFR/TD4							
	SFFR/TD5							
	SFFR/TD6							
	SFFR/TD7							
	Reserved							
	Reserved							
Reset Value	0x00_0000_0000_0000_0000_0000_0000							

Bit	**Name**	**Access**	**Description**
3:0	DLC	R/W	Data Length Code. The number of bytes in the data field of a message is coded by the data length code. For remote frame transmission, the data length code must be filled with "0000" with the RTR bit being set to "1". Say if the DLC is filled with "0001" with RTR bit being set to "1", this remote frame being transmitted will have 1 byte data frame, which should be avoided. The data length code must be specified correctly to avoid bus error if two CAN controllers start a remote frame transmission with the same identifier simultaneously. For reasons of compatibility no data length code >8 should be used. If a value >8 is written, 8 bytes are transmitted in the data frame with the data length code specified in DLC.
5:4	Reserved	-	
6	RTR	R/W	Remote Transmission Request. 1: A Remote frame is received or will be transmitted on the bus. This means that no data bytes are included within this frame. Also, DLC field must be filled with "0000". 0: A Data frame is received or will be sent including the number of data bytes as specified by the data length code.
7	IDE	R/W	Identifier Extension Bit. 1: The Extended Frame Format is received or will be transmitted by the CAN controller. 0: Standard Frame Format is received or will be transmitted by the CAN controller.
15:8 23:21	ID28:ID21 ID20:ID18	R/W	Message Identifier. The Identifier consists of 11 bits (ID28~ID18). The ID28 is the most significant bit, which is received/transmitted first on the bus. The Identifier acts as the message's name. It is used in a receiver for acceptance filtering and also determining the bus access priority during the arbitration process. The lower the binary value of the Identifier the higher the priority. This is due to a larger number of leading dominant bits during arbitration.
31:24 … 87:80	SFFR/TD0 … SFFR/TD7	R/W	Standard Frame Format Rx or Tx Data. The received data of the CAN message that has passed the acceptance test defined in CANACR and CANAMR registers. The number of received data bytes is determined by the data length code. The first bit received is the most significant bit of data byte SFFRD0.

			The number of transmitted data bytes is determined by the data length code. The first bit transmitted is the most significant bit of data byte SFFTD0. The transmission data buffer has 8 bytes.

✧ **Rx and Tx Frame Information with Extended Frame Format (EFF)**

Bit	7	6	5	4	3	2	1	0
Name	IDE	RTR	Reserved		DLC			
	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5
	ID4	ID3	ID2	ID1	ID0	Reserved		
	EFFR/TD0							
	EFFR/TD1							
	EFFR/TD2							
	EFFR/TD3							
	EFFR/TD4							
	EFFR/TD5							
	EFFR/TD6							
	EFFR/TD7							
Reset Value	0x00_0000_0000_0000_0000_0000_0000							

Bit	**Name**	**Access**	**Description**
3:0	DLC	R/W	Data Length Code. The number of bytes in the data field of a message is coded by the data length code. For remote frame transmission, the data length code must be filled with “0000” with the RTR bit being set to “1”. Say if the DLC is filled with “0001” with RTR bit being set to “1”, this remote frame being transmitted will have 1 byte data frame, which should be avoided. The data length code must be specified correctly to avoid bus error if two CAN controllers start a remote frame transmission with the same identifier simultaneously. For reasons of compatibility no data length code >8 should be used. If a value >8 is written, 8 bytes are transmitted in the data frame with the data length code specified in DLC.
5:4	Reserved	-	
6	RTR	R/W	Remote Transmission Request. 1: A Remote frame is received or will be transmitted on the bus. This means that no data bytes are included within this frame. Also, DLC field must be filled with “0000”. 0: A Data frame is received or will be sent including the number of data bytes as specified by the data length code.
7	IDE	R/W	Identifier Extension Bit. 1: The Extended Frame Format is received or will be transmitted by the CAN controller. 0: Standard Frame Format is received or will be transmitted by the CAN controller.
15:8 … 39:35	ID28:ID21 … ID4:ID0	R/W	Message Identifier. The Identifier consists of 29 bits (ID28~ID0). The ID28 is the most significant bit, which is received/transmitted first on the bus. The Identifier acts as the message’s name. It is used in a receiver for acceptance filtering and also determining the bus access priority during the arbitration process. The lower the binary value of the Identifier the higher the priority. This is due to a larger number of leading dominant bits during arbitration.
47:40 … 103:96	EFFR/TD0 … EFFR/TD7	R/W	Extended Frame Format Rx or Tx Data. The received data of the CAN message that has passed the acceptance test defined in CANACR and CANAMR registers. The number of received data bytes is determined by the data length code. The first bit received is the most significant bit of data byte EFFRD0.

			The number of transmitted data bytes is determined by the data length code. The first bit transmitted is the most significant bit of data byte EFFTD0. The transmission data buffer has 8 bytes.

CAN Rx Message Count Register (CANRMC, 0x1D)

Bit	7	6	5	4	3	2	1	0
Name	Reserved	RMC						
Reset Value	0	000_0000						

Bit	Name	Access	Description
6:0	RMC	RO	Rx Message Count. The RMC register reflects the number of messages available within the RX FIFO. Note that upon entering Reset mode, this value will be cleared to 0x00 by hardware.
7	Reserved	-	

CAN Mode Register (CANMR, 0x1F)

Bit	7	6	5	4	3	2	1	0
Name	EXTM	Reserved						
Reset Value	0	000_0000						

Bit	Name	Access	Description
6:0	Reserved	-	
7	EXTM	R/W	Extended Mode. 1: Configure CAN controller to operate in Extended mode. 0: Configure CAN controller to operate in Basic mode. The write access to this register is possible only in Reset mode while in Operating mode this register is read-only.

Example Programming Procedure in CAN Controller Extended Mode

Configure CAN controller to Extended mode, Baud rate = 1Mbps, and the operating system clock is 100Mhz. Transmit one Standard Frame Format message with ID = 0x264 and data length = 4 bytes. The software is running in interrupt mode.

1. Write 0x01 to CANCR register to enter Reset mode.
2. Write 0xC4 to CANBTR0 register.
3. Write 0x34 to CANBTR1 register.
4. Write 0x0000_804C to CANACR register, where the least significant byte is ACR0 = 0x4C.
5. Write 0xFFFF_1F00 to CANAMR register, where the least significant byte is AMR0 = 0x00.
6. Write 0xEF to CANIER register to enable interrupt.
7. Write 0x00 to CANCR register to exit Reset mode.
8. Write 0x00_0000_0000_0043_2180_0180_4C04 to CANTFI register, where the most significant 6 bytes of "0000_0000_0000" are just dummy data for writing CANTFI register.
9. Write 0x01 to CANCMR register.
10. Wait for interrupt and then check CANISR.1 (TI bit), which should be asserted. Read CANSR.2 (TBS) which should be "1" to indicate transmit complete.

For receiving one Standard Frame Format message with ID = 0x264.

1. Write CANACR, CANAMR with proper value, such as example value in above step 4 and 5.
2. Wait for interrupt and then check CANISR.0 (RI bit), which should be asserted. Read CANSR.0 (RBS) which should be "1" to indicate at least one message is available in RX FIFO. Also check the ORS bit.
3. Read the received message from CANRFI register up to 13 bytes of data.
4. Write 0x04 to CANCMR register to release buffer space in RX FIFO.

5.0 Electrical Specifications

5.1 DC Characteristics

5.1.1 Absolute Maximum Ratings

Symbol	Parameter	Rating	Unit
VCC18, VCCK	Output voltage of on-chip voltage regulator or digital core power supply.	- 0.3 to 2.16	V
VCC3R, VCCIO	Power supply of on-chip voltage regulator or 3.3V I/O.	- 0.3 to 4.0	V
VCC18A	Analog power supply for oscillator, PLL, etc.	- 0.3 to 2.16	V
VCC3A	Analog power supply for bandgap.	- 0.3 to 3.8	V
V_{IN18}	Input voltage of 1.8V I/O.	- 0.3 to 2.16	V
V_{IN3}	Input voltage of 3.3V I/O.	- 0.3 to 4.0	V
	Input voltage of 3.3V I/O with 5V tolerant.	- 0.3 to 5.8	V
T_{STG}	Storage temperature.	- 40 to 150	℃
I_{IN}	DC input current.	20	mA
I_{OUT}	Output short circuit current.	20	mA

Note: Permanent device damage may occur if absolute maximum ratings are exceeded. Functional operation should be restricted in the recommended operating condition section of this datasheet. Exposure to absolute maximum rating condition for extended periods may affect device reliability.

5.1.2 Recommended Operating Condition

Symbol	Parameter	Min	Typ	Max	Unit
VCC3R	Power supply of on-chip voltage regulator.	3.0	3.3	3.6	V
VCCIO	Power supply of 3.3V I/O.	3.0	3.3	3.6	V
VCC3A	Analog power supply for bandgap.	3.0	3.3	3.6	V
VCC18	Output voltage of on-chip voltage regulator.	1.62	1.8	1.98	V
VCCK	Digital core power supply.	1.62	1.8	1.98	V
VCC18A	Analog power supply for oscillator, PLL, etc.	1.62	1.8	1.98	V
V_{IN18}	Input voltage of 1.8 V I/O.	0	1.8	1.98	V
V_{IN3}	Input voltage of 3.3 V I/O.	0	3.3	3.6	V
	Input voltage of 3.3 V I/O with 5 V tolerance.	0	3.3	5.25	V
T_j	AX11025 LF operating junction temperature.	0	25	125	℃
	AX11025 LI operating junction temperature.	-40	25	125	℃
T_a	AX11025 LF operating ambient temperature.	0	-	70	℃
	AX11025 LI operating ambient temperature.	-40	-	85	℃

5.1.3 Leakage Current and Capacitance

Symbol	Parameter	Condition	Min	Typ	Max	Unit
I_{IN}	Input current .	No pull-up or pull-down	-10	±1	10	µA
I_{OZ}	Tri-state leakage current .		-10	±1	10	µA
C_{IN}	Input capacitance.		-	2.2	-	pF
C_{OUT}	Output capacitance.		-	2.2	-	pF
C_{BID}	Bi-directional buffer capacitance.		-	2.2	-	pF

Note: The capacitance listed above does not include pad capacitance and package capacitance. One can estimate pin capacitance by adding a pad capacitance of about 0.5pF and the package capacitance.

5.1.4 DC Characteristics of 3.3V I/O Pins

Symbol	Parameter	Condition	Min	Typ	Max	Unit
VCCIO	Power supply of 3.3V I/O.	3.3V I/O	3.0	3.3	3.6	V
Tj	Junction temperature.		-40	25	125	℃
Vil	Input low voltage.	LVTTL	-	-	0.8	V
Vih	Input high voltage.		2.0	-	-	V
Vt	Switching threshold.			1.5		V
Vt-	Schmitt trigger negative going threshold voltage.	LVTTL	0.8	1.1	-	V
Vt+	Schmitt trigger positive going threshold voltage		-	1.6	2.0	V
Vol	Output low voltage.	Iol = 4~8mA	-	-	0.4	V
Voh	Output high voltage.	Ioh = -4~-8mA	2.4	-	-	V
Rpu	Input pull-up resistance.	Vin = 0	40	75	190	KΩ
Rpd	Input pull-down resistance.	Vin = VCCIO	40	75	190	KΩ
Iin	Input leakage current.	Vin = VCCIO or 0	-10	±1	10	µA
	Input leakage current with pull-up resistance.	Vin = 0	-15	45	-85	µA
	Input leakage current with pull-down resistance.	Vin = VCCIO	15	45	85	µA
Ioz	Tri-state output leakage current.		-10	±1	10	µA

5.1.5 DC Characteristics of 3.3V with 5V Tolerant I/O Pins

Symbol	Parameter	Condition	Min	Typ	Max	Unit
VCCIO	Power supply of 3.3V I/O.	3.3V I/O	3.0	3.3	3.6	V
Tj	Junction temperature.		-40	25	125	℃
Vil	Input low voltage.	LVTTL	-	-	0.8	V
Vih	Input high voltage.		2.0	-	-	V
Vt	Switching threshold.			1.5		V
Vt-	Schmitt trigger negative going threshold voltage.	LVTTL	0.8	1.1	-	V
Vt+	Schmitt trigger positive going threshold voltage		-	1.6	2.0	V
Vol	Output low voltage.	Iol = 4~8mA	-	-	0.4	V
Voh	Output high voltage.	Ioh = -4~-8mA	2.4	-	-	V
Rpu	Input pull-up resistance.	Vin = 0	40	75	190	KΩ
Rpd	Input pull-down resistance.	Vin = VCCIO	40	75	190	KΩ
Iin	Input leakage current.	Vin = 5.5V or 0	-	±5	-	µA
	Input leakage current with pull-up resistance.	Vin = 0	-15	-45	-85	µA

	Input leakage current with pull-down resistance	Vin = VCCIO	15	45	85	μA
Ioz	Tri-state output leakage current.	Vin = 5.5V or 0	-	±10	-	μA

5.1.6 DC Characteristics of Voltage Regulator

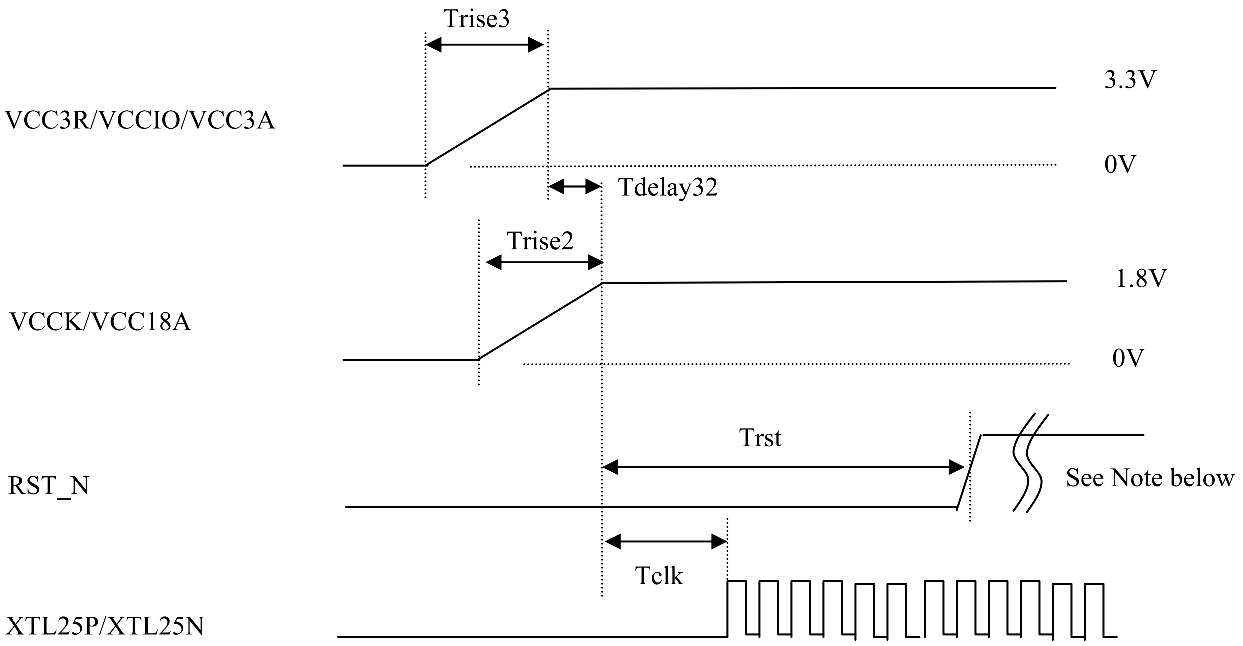
Symbol	Description	Condition	Min	Typ	Max	Unit
VCC3R	Power supply of on-chip voltage regulator.		3.0	3.3	3.6	V
Tj	Operating junction temperature.		-40	25	125	℃
Iload	Driving current.	Normal operation, RSM bit = 0 (SFR register PCON.3)	-	-	240	mA
	Driving current.	Standby mode enabled, RSM bit = 1 (PCON.3)	-	-	30	mA
VCC18	Output voltage of on-chip voltage regulator.	VCC3R = 3.3V	1.71	1.8	1.89	V
Vdrop	Dropout voltage.	△VCC18 = -1%, Iload = 10mA	-	0.1	0.2	V
$\frac{\triangle VCC18}{(\triangle VCC3R \text{ x } VCC18)}$	Line regulation.	VCC3R = 3.3V, Iload = 50mA	-	0.2	0.4	%/V
$\frac{\triangle VCC18}{(\triangle Iload \text{ x } VCC18)}$	Load regulation.	VCC3R = 3.3V, 1mA ≦ Iload ≦ 240mA	-	0.02	0.05	%/mA
$\frac{\triangle VCC18}{\triangle Tj}$	Temperature coefficient.	VCC3R = 3.3V,-40℃ ≦ Tj ≦ 125℃	-	+/-0.2	+/-0.5	mV/℃
Iq_25℃	Quiescent current at 25 ℃.	VCC3R = 3.3V, RSM bit = 1 (PCON.3)	-	70	100	μA
		VCC3R = 3.3V, RSM bit = 0 (SFR register PCON.3)	-	100	125	μA
Iq_125℃	Quiescent current at 125 ℃.	VCC3R = 3.3V, RSM bit = 1 (PCON.3)	-	85	115	μA
		VCC3R = 3.3V, RSM bit = 0 (SFR register PCON.3)	-	125	170	μA
Cout	Output external capacitor.		0.1	1	-	μF
ESR	Allowable effective series resistance of external capacitor.		-	0.5	1	Ω

5.2 Power Consumption

Symbol	Description	System Clock	Condition	Min	Typ	Max	Unit
I3.3V	Total current of 3.3V power supply including VCCIO, VCC3A, and VCC3R. Note VCC3R includes VCC18, VCCK, and VCC18A	25 Mhz	CPU full speed, Ethernet 10Mbps full duplex	-	166	-	mA
			CPU full speed, Ethernet 100Mbps full duplex	-	177	-	mA
			CPU in PMM, Ethernet 10Mbps full duplex (Ethernet PHY not powered down)	-	139.3	-	mA
			CPU in PMM, Ethernet 100Mbps full duplex (Ethernet PHY not powered down)	-	147.7	-	mA
			CPU in PMM, Ethernet PHY powered down	-	21.6	-	mA
			CPU in STOP, Ethernet 10M full duplex mode (Ethernet PHY not powered down and OSC/PLL still running)	-	138.8	-	mA
			CPU in STOP, Ethernet 100M full duplex mode (Ethernet PHY not powered down and OSC/PLL still running)	-	146.8	-	mA
			CPU in STOP, Ethernet PHY powered down (OSC/PLL still running)	-	21.3	-	mA
			CPU in STOP, OSC/PLL stopped (TOFFOP of I2C EEPROM offset 0x01 = 1)	-	0.46	-	mA
		50 Mhz	CPU full speed, Ethernet 10Mbps full duplex	-	188	-	mA
			CPU full speed, Ethernet 100Mbps full duplex	-	197	-	mA
			CPU in PMM, Ethernet 10Mbps full duplex (Ethernet PHY not powered down)	-	139.7	-	mA
			CPU in PMM, Ethernet 100Mbps full duplex (Ethernet PHY not powered down)	-	148.8	-	mA
			CPU in PMM, Ethernet PHY powered down	-	22.1	-	mA
			CPU in STOP, Ethernet 10M full duplex mode (Ethernet PHY not powered down and OSC/PLL still running)	-	139	-	mA
			CPU in STOP, Ethernet 100M full duplex mode (Ethernet PHY not powered down and OSC/PLL still running)	-	147.3	-	mA
			CPU in STOP, Ethernet PHY powered down (OSC/PLL still running)	-	21.5	-	mA
			CPU in STOP, OSC/PLL stopped (TOFFOP of I2C EEPROM offset 0x01 = 1)	-	0.52	-	mA
		100 Mhz	CPU full speed, Ethernet 10Mbps full duplex	-	227	-	mA
			CPU full speed, Ethernet 100Mbps full duplex	-	235	-	mA
			CPU in PMM, Ethernet 10Mbps full duplex (Ethernet PHY not powered down)	-	140.3	-	mA
			CPU in PMM, Ethernet 100Mbps full duplex (Ethernet PHY not powered down)	-	149.1	-	mA
			CPU in PMM, Ethernet PHY powered down	-	22.7	-	mA
			CPU in STOP, Ethernet 10M full duplex mode (Ethernet PHY not powered down and OSC/PLL still running)	-	139.2	-	mA
			CPU in STOP, Ethernet 100M full duplex mode (Ethernet PHY not powered down and OSC/PLL still running)	-	147.3	-	mA
			CPU in STOP, Ethernet PHY powered down (OSC/PLL still running)	-	21.7	-	mA
			CPU in STOP, OSC/PLL stopped (TOFFOP of I2C EEPROM offset 0x01 = 1)	-	0.59	-	mA
Θ_{JC}	Thermal resistance of junction to case			-	9.7	-	°C/W
Θ_{JA}	Thermal resistance of junction to ambient		Still air	-	45.0	-	°C/W

5.3 Power-up Sequence

At power-up, AX11025 requires the VCC3R/VCCIO/VCC3A power supply to rise to nominal operating voltage within Trise3 and the VCCK/VCC18A power supply to rise to nominal operating voltage within Trise2.



Symbol	Parameter	Condition	Min	Typ	Max	Unit
T_{rise3}	3.3V power supply rise time	From 0V to 3.3V	1	-	10	ms
T_{rise2}	1.8V power supply rise time	From 0V to 1.8V	-	-	10	ms
$T_{delay32}$	3.3V rise to 1.8V rise time delay		-5	-	5	ms
Tclk	25Mhz crystal oscillator start-up time	From VCC18A = 1.8V to first clock transition of XTL25P or XTL25N	-	1	-	ms
Trst	RST_N low level interval	From VCCK = 1.8V to RST_N going high	4	-	-	ms

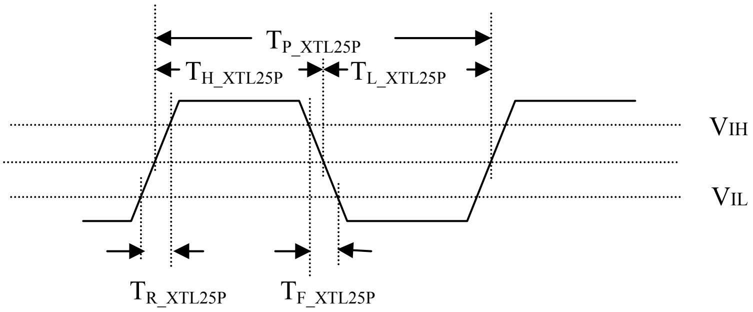
Note: After RST_N input pin is negated during power-on, the internal I2C boot loader may start loading I2C EEPROM automatically, upon enabled. User should avoid generating 2nd reset pulse to RST_N pin of AX11025 because it will cause the I2C boot loader to be reset again during the process of loading I2C EEPROM configuration parameter. This may cause the I2C EEPROM itself to remain in the "read state", which it may not respond to AX11025's later I2C commands properly, because the I2C EEPROM normally does not have reset pin to reset it while the AX11025 is being reset again and restarting a new I2C command.

Figure 129: Power-up Sequence Timing Diagram and Table

5.4 AC Timing Characteristics

5.4.1 Clock Timing

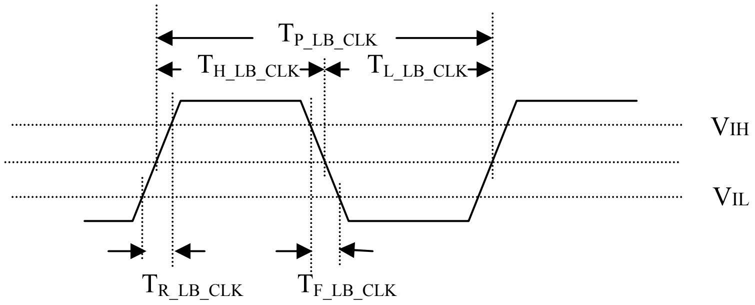
XTL25P



Symbol	Parameter	Condition	Min	Typ	Max	Unit
	XTL25P reference frequency		25-0.005%	25	25+0.005%	Mhz
	XTL25P clock duty cycle		40	50	60	%
T_{P_XTL25P}	XTL25P clock cycle time		-	40	-	ns
T_{H_XTL25P}	XTL25P clock high time		-	20	-	ns
T_{L_XTL25P}	XTL25P clock low time		-	20	-	ns
T_{R_XTL25P}	XTL25P rise time	V_{IL} (max) to V_{IH} (min)	-	-	1.0	ns
T_{F_XTL25P}	XTL25P fall time	V_{IH} (min) to V_{IL} (max)	-	-	1.0	ns

Figure 130: XTL25P Clock Timing Diagram and Table

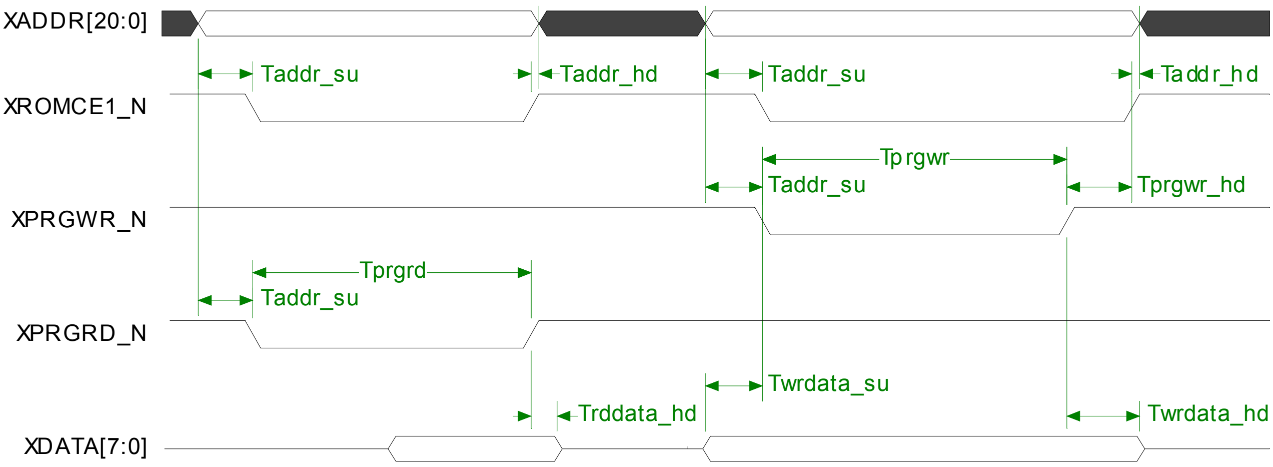
LB_CLK



Symbol	Parameter	Condition	Min	Typ	Max	Unit
$T_{P\_LB\_CLK}$	LB_CLK clock cycle time		10	-	40	ns
$T_{H\_LB\_CLK}$	LB_CLK clock high time		5	-	20	ns
$T_{L\_LB\_CLK}$	LB_CLK clock low time		5	-	20	ns
$T_{R\_LB\_CLK}$	LB_CLK rise time	V_{IL} (max) to V_{IH} (min)	-	-	1.0	ns
$T_{F\_LB\_CLK}$	LB_CLK fall time	V_{IH} (min) to V_{IL} (max)	-	-	1.0	ns

Figure 131: LB_CLK Clock Timing Diagram and Table

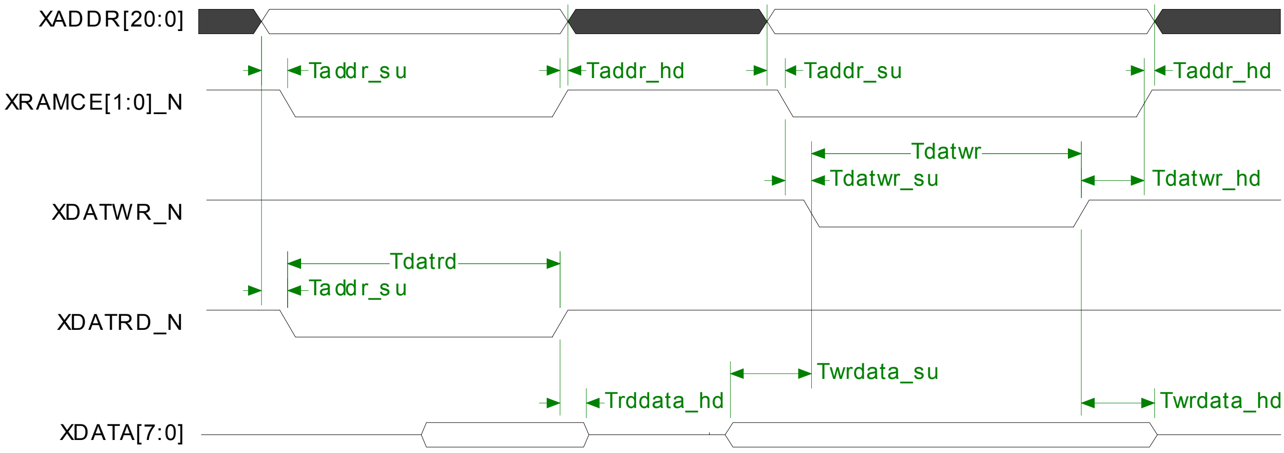
5.4.2 External Memory Interface Timing



Symbol	Description	Min	Typ	Max	Unit
Taddr_su	Address setup time before XROMCE1_N, XPRGRD_N, or XPRGWR_N activation.	0.5	-	-	Tsys_clk[1]
Taddr_hd	Address hold time after XROMCE1_N negation.	0	-	-	ns
Tprgrd	XPRGRD_N activation width.	-	WTST[2:0]+1	-	Tsys_clk
Trddata_hd	Read data hold time after XROMCE1_N or XPRGRD_N negation.	0	-	-	ns
Tprgwr	XPRGWR_N activation width.	-	WTST[2:0]+1	-	Tsys_clk
Tprgwr_hd	XPRGWR_N negation to XROMCE1_N negation time.	0.5	-	-	Tsys_clk
Twrdata_su	Write data setup time before XPRGWR_N activation.	0.5	-	-	Tsys_clk
Twrdata_hd	Write data hold time after XPRGWR_N negation.	0.5	-	-	Tsys_clk

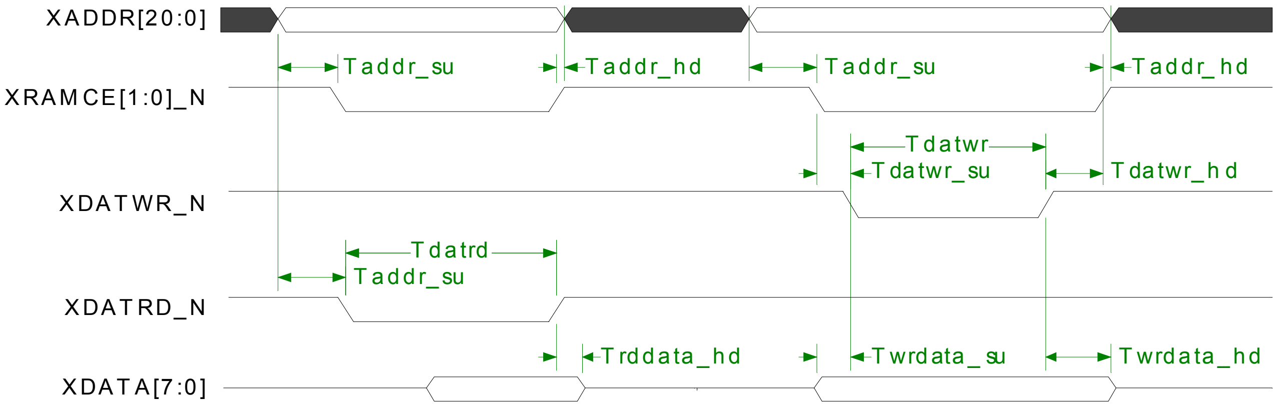
Figure 132: CPU Program Read and Program Write Timing Diagram and Table

[1] Tsys_clk = 10/20/40ns for 100/50/25Mhz operating system clock.



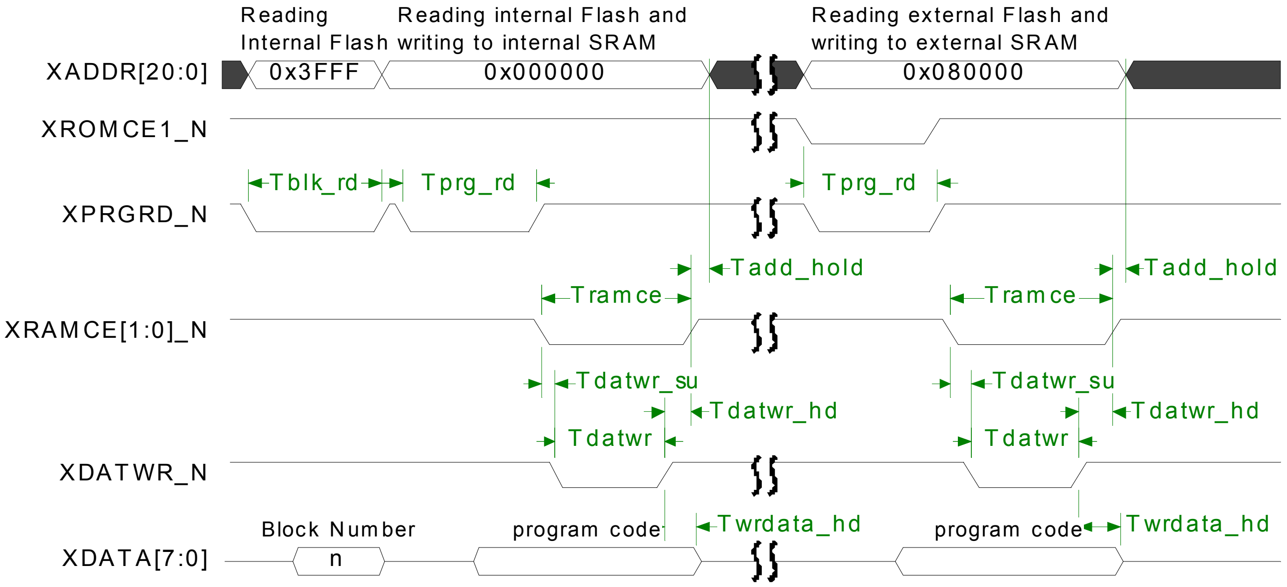
Symbol	Description	Min	Typ	Max	Unit
Taddr_su	Address setup time before XRAMCE[1:0]_N or XDATRD_N activation.	0	-	-	ns
Taddr_hd	Address hold time after XRAMCE[1:0]_N or XDATRD_N negation .	0	-	-	ns
Tdatrd	XDATRD_N activation width.	-	CKCON[2:0]+1	-	Tsys_clk
Trddata_hd	Read data hold time after XRAMCE[1:0]_N or XDATRD_N negation.	0	-	-	ns
Tdatwr_su	XRAMCE[1:0] activation to XDATWR_N activation.	1			ns
Tdatwr	XDATWR_N activation width.	-	CKCON[2:0]+1	-	Tsys_clk
Tdatwr_hd	XDATWR_N negation to XRAMCE[1:0]_N negation time.	0.5	-	-	Tsys_clk
Twrdata_su	Write data setup time before XDATWR_N activation.	0.5	-	-	Tsys_clk
Twrdata_hd	Write data hold time after XDATWR_N negation.	0.5	-	-	Tsys_clk

Figure 133: CPU Data Read and Data Write Timing Diagram and Table



Symbol	Description	Min	Typ	Max	Unit
Taddr_su	Address setup time before XRAMCE[1:0]_N or XDATRD_N activation.	0.5	-	-	Tsys_clk
Taddr_hd	Address hold time after XRAMCE[1:0]_N or XDATRD_N negation .	0	-	-	ns
Tdatrd	XDATRD_N activation width.	-	CKCON[2:0]+1	-	Tsys_clk
Trddata_hd	Read data hold time after XRAMCE[1:0]_N or XDATRD_N negation.	0	-	-	ns
Tdatwr_su	XRAMCE[1:0] activation to XDATWR_N activation.	1	-	-	ns
Tdatwr	XDATWR_N activation width.	-	CKCON[2:0]	-	Tsys_clk
Tdatwr_hd	XDATWR_N negation to XRAMCE[1:0]_N negation time.	0.5	-	-	Tsys_clk
Twrdata_su	Write data setup time before XDATWR_N activation.	0	-	-	ns
Twrdata_hd	Write data hold time after XDATWR_N negation.	0.5	-	-	Tsys_clk

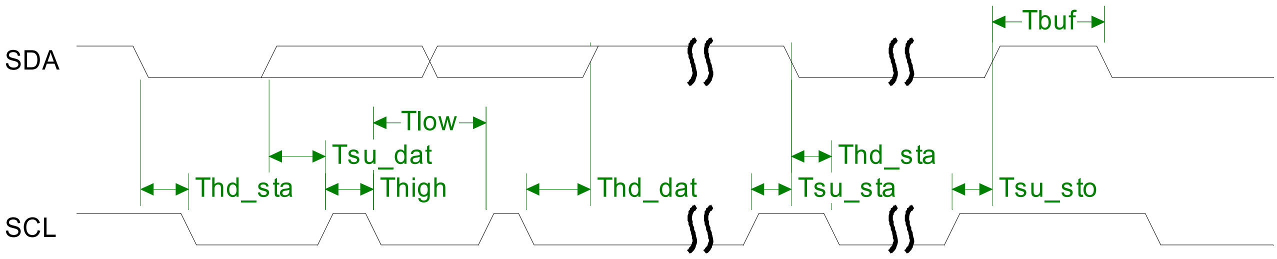
Figure 134: DMA Data Read and Data Write Timing Diagram and Table



Symbol	Description	Min	Typ	Max	Unit
Tblk_rd	XPRGRD_N activation width to read the “block number” located in program address 0x003FFF.	-	120	-	ns
Tprg_rd	XPRGRD_N activation width to read 1 byte program code from Flash memory.	120	-	-	ns
Tadd_hold	Address hold time after XRAMCE[1:0]_N negation.	0	-	-	ns
Tramce	XRAMCE[1:0] activation width to write 1 byte program code to SRAM.	120			ns
Tdatwr_su	XRAMCE[1:0] activation to XDATWR_N activation.	1			ns
Tdatwr	XDATWR_N activation width to write 1 byte program code to SRAM.	80	-	-	ns
Tdatwr_hd	XDATWR_N negation to XRAMCE[1:0]_N negation time.	0.5			Tsys_clk
Twrdata_hd	Write data hold time after XDATWR_N negation.	0.5	-	-	Tsys_clk

Figure 135: Boot Loader Reading Flash and Writing SRAM Timing Diagram and Table

5.4.3 I2C Interface Timing



Symbol	Parameter	Min	Typ	Max	Unit
Fclk	SCL clock frequency.	-	100, 400	-	KHz
Thd_sta	Hold time of (repeated) START condition. After this period, the first clock pulse is generated.	-	2	-	Tprsc[2]
Thigh	High period of the SCL clock.	-	2	-	Tprsc
Tlow	Low period of the SCL clock.	-	3	-	Tprsc
Tsu_sta	Setup time for a repeated START condition.	-	2	-	Tprsc
Tsu_dat	Data Setup time.	-	1	-	Tprsc
Thd_dat	Data hold time.	-	2	-	Tprsc
Tsu_sto	Setup time for STOP condition	-	2	-	Tprsc
Tbuf	Bus free time between a STOP and START condition.	-	4	-	Tprsc

Table 56: I2C Master Controller Timing Table

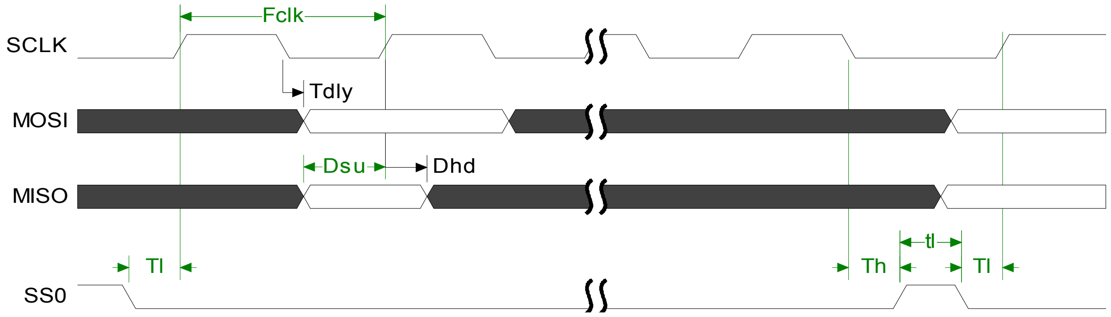
Symbol	Parameter	Min	Typ	Max	Unit
Fclk	SCL clock frequency	-	-	380	KHz
Thd_sta	Hold time of (repeated) START condition. After this period, the first clock pulse is generated.	3	-	-	Tsys_clk[3]
Thigh	High period of the SCL clock in Standard mode.	4	-	-	µs
	High period of the SCL clock in Fast mode .	0.6	-	-	µs
Tlow	Low period of the SCL clock.	0.4	-	-	µs
Tsu_sta	Setup time for a repeated START condition.	1	-	-	Tsys_clk
Tsu_dat	Data Setup time.	3	-	-	Tsys_clk
Thd_dat	Data hold time.	0.4	-	-	µs
Tsu_sto	Setup time for STOP condition	3	-	-	Tsys_clk
Tbuf	Bus free time between a STOP and START condition.	1.3	-	-	µs

Table 57: I2C Slave Controller Timing Table

[2] Tprsc = 1/Fprsc, where Fprsc = Operating system clock frequency / (PRER + 1) and the PRER is I2C Clock Prescale Register.

[3] Tsys_clk = 10/20/40ns for 100/50/25Mhz operating system clock.

5.4.4 SPI Interface Timing



Note: Above diagram only shows setup and hold time relationship of SPI pins in Mode 0. For other 3 modes, they are quite similar except that the clock polarity is reversed.

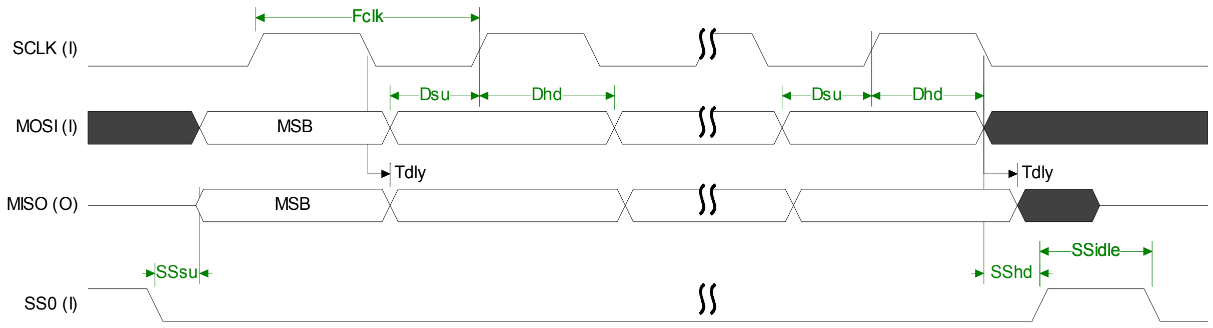
Symbol	Description	Min	Typ	Max	Unit
Fclk	SCLK clock frequency.	-	$\frac{\text{Fsys\_clk}}{\text{(SPIBRR+1)*2}}$	-	MHz[4]
Tl	Setup time of SS[2:0] to the first SCLK edge.	-	0.5	-	Tclk[5]
Th	Hold time of SS[2:0], after the last SCLK edge.	-	0.5	-	Tclk
tl	Minimum idle time between transfers (minimum SS[2:0] high time).	-	0.5	-	Tclk
Tdly	MOSI data valid time, after SCLK edge.	-	-	1	Tsys_clk[6]
Dsu	MISO data setup time before SCLK edge.	5.5	-	-	ns
Dhd	MISO data hold time after SCLK edge.	6	-	-	ns
	Internal time base period.	-	0.5	-	Tclk

Figure 136: SPI Master Controller Timing Diagram and Table

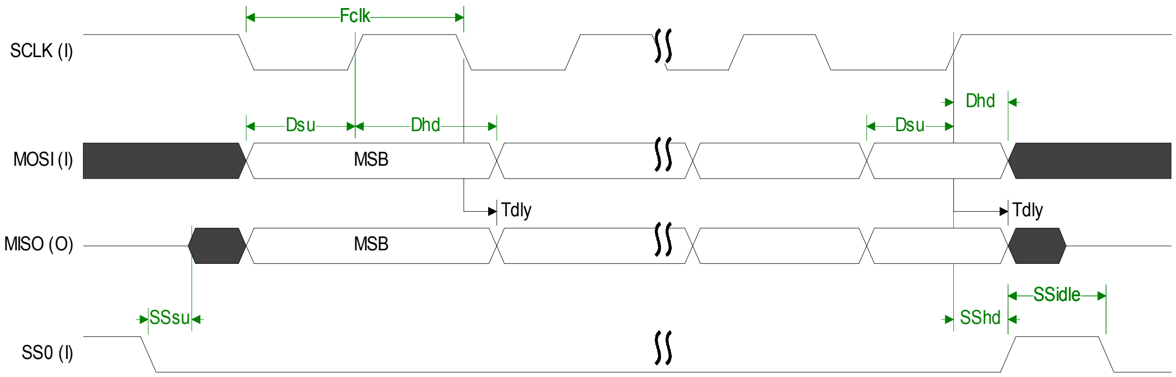
[4] Fsys_clk is the operating system clock frequency, 25Mhz, 50Mhz, or 100Mhz. The SPIBRR is SPI Baud Rate Register and its minimum setting value is 0x01 and setting to 0x00 is invalid..

[5] Tclk = 1/Fclk.

[6] Tsys_clk = 10/20/40ns for 100/50/25Mhz operating system clock.



SPI Slave Mode Timing Diagram in Mode 0

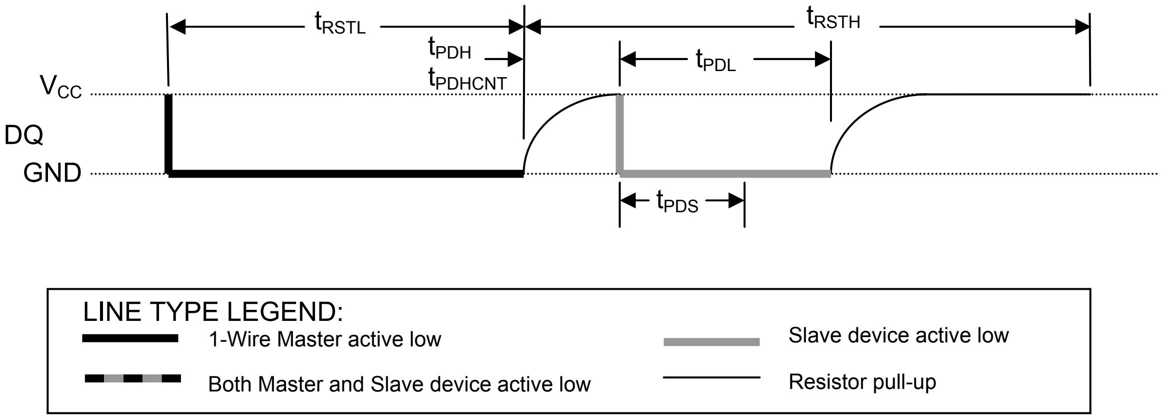


SPI Slave Mode Timing Diagram in Mode 3

Symbol	Description	Min	Typ	Max	Unit
Fclk	SCLK clock frequency at 100Mhz system clock.	-	-	6	MHz
	SCLK clock frequency at 50Mhz system clock.	-	-	3	MHz
	SCLK clock frequency at 25Mhz system clock.	-	-	1.5	MHz
Tdly	MISO data valid time after SCLK edge.	-	-	2 + (12ns)	Tsys_clk
Dsu	MOSI data setup time before SCLK edge.	3	-	-	ns
Dhd	MOSI data hold time after SCLK edge.	2 + (2ns)	-	-	Tsys_clk
SSsu	SS0 setup time before SCLK edge.	8	-	-	ns
SShd	SS0 hold time after SCLK edge.	2 + (2ns)	-	-	Tsys_clk
SSidle	SS0 negation to next SS0 active time	2			Tsys_clk

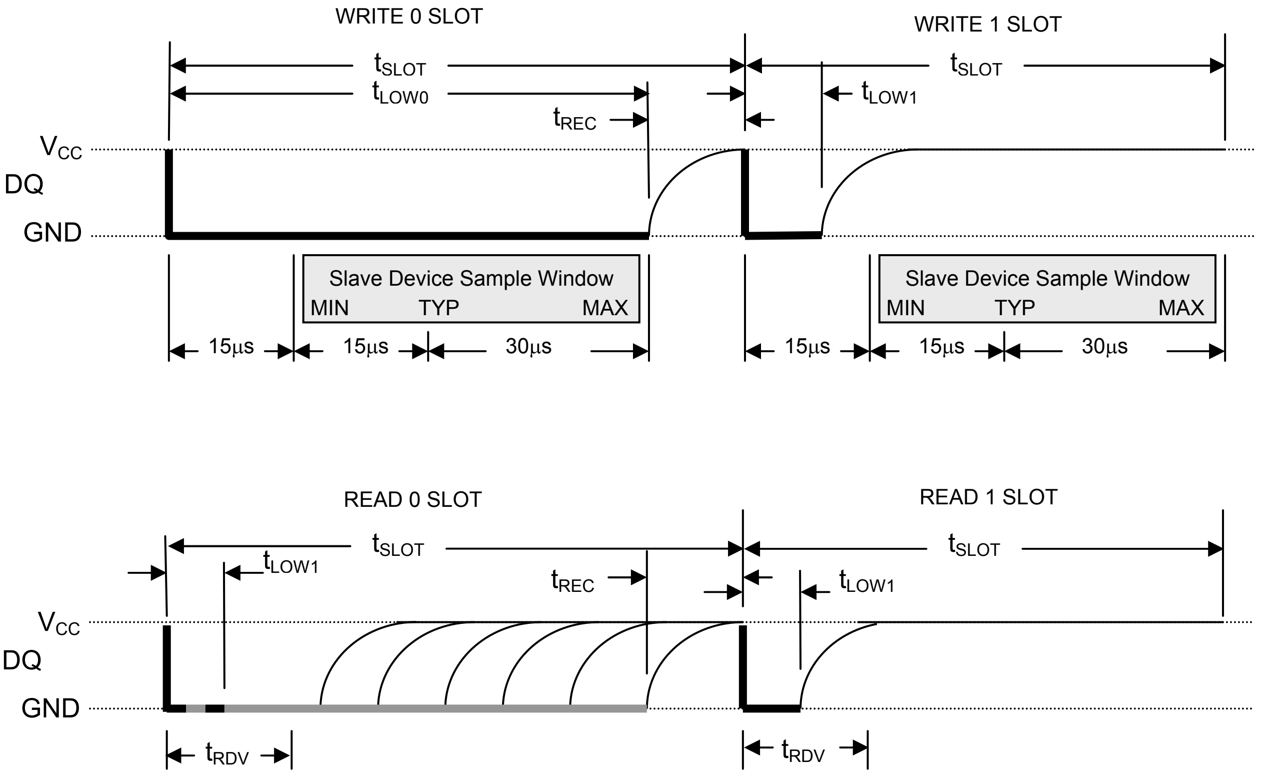
Figure 137: SPI Slave Controller Timing Diagram and Table

5.4.5 1-Wire Interface Timing



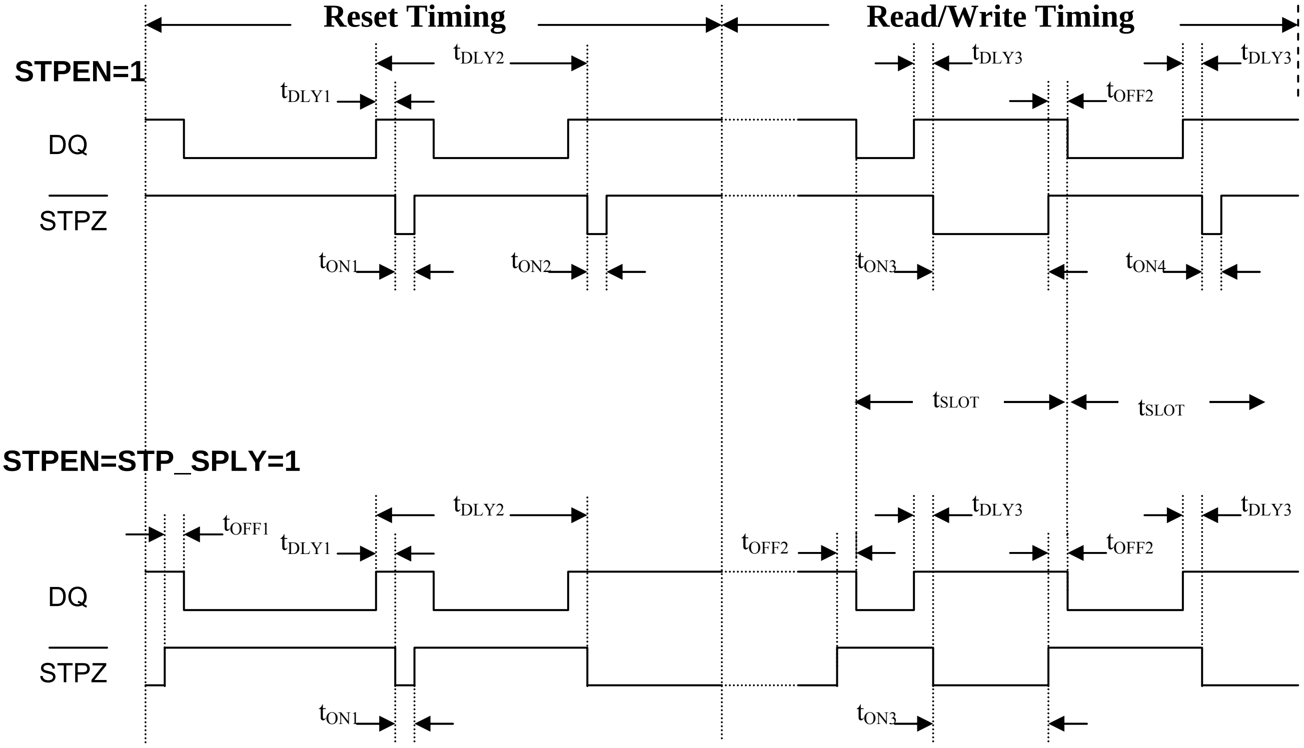
Symbol	Parameter	Conditions	Min	Max	Units
t_{RSTL}	Reset Time Low	Standard	500.8	626	µs
		Overdrive	50.4	63	µs
t_{RSTH}	Reset Time High	Standard	508.8	636	µs
		Overdrive	59.2	74	µs
t_{PDH}	Presence Detect High	Standard	15	60	µs
		Overdrive	2	6	µs
t_{PDL}	Presence Detect Low	Standard	60	240	µs
		Overdrive	6	24	µs
t_{PDS}	Presence Detect Sample	Standard	24	31	µs
		Standard – Long Line Mode	30.4	38	µs
		Overdrive	2.4	4	µs

Figure 138: 1-Wire Reset Pulse and Presence Pulse Timing Diagram and Table



Symbol	Parameter	Conditions	Min	Max	Units
t_{SLOT}	Time Slot	Standard	68.8	86	µs
		Overdrive	12	15	µs
t_{LOW0}	Write 0 Low Time	Standard	62.4	78	µs
		Overdrive	8	12	µs
t_{LOW1}	Write 1 Low Time	Standard	4.8	6	µs
		Standard – Long Line Mode	7.2	9	µs
		Overdrive	0.8	1	µs
t_{RDV}	Read Data Value	Standard	12	15	µs
		Standard – Long Line Mode	20	25	µs
		Overdrive	1.6	2	µs
t_{REC}	Recovery Time	Standard	5.5	8	µs
		Standard – Long Line Mode	11.2	14	µs
		Overdrive	4	5	µs
	Time base Period		0.96	1	µs

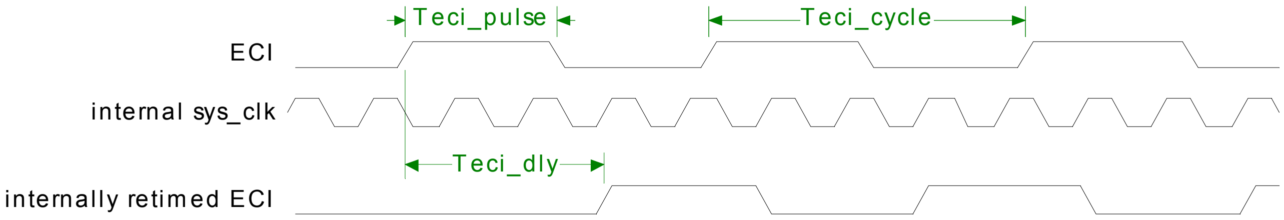
Figure 139: 1-Wire Write and Read Time Slot Timing Diagram and Table



Symbol	Description	Conditions	Min	Max	Units
t_{OFF1}	Turn Off Time for 1-Wire Reset	Standard	1.6	2	µs
		Overdrive	1.6	2	µs
t_{DLY1}	Delay Time for Presence Detect	Standard	0.8	1	µs
		Overdrive	0.8	1	µs
t_{ON1}	Active Time for Presence Detect	Standard	6.4	8	µs
		Overdrive	0.8	1	µs
t_{DLY2}	Delay Time for Presence Detect Recovery	Standard	399.2	499	µs
		Overdrive	31.2	39	µs
t_{ON2}	Active Time for Presence Detect Recovery	Standard	8	10	µs
		Overdrive	8	10	µs
t_{DLY3}	Delay Time for Write1/Write0 Recovery	Standard	0.8	1	µs
		Overdrive	0.8	1	µs
t_{ON3}	Active Time for Write 1 Recovery	Standard	51.2	78	µs
		Overdrive	7.2	9	µs
t_{OFF2}	Turn Off Time for Write1/Write0	Standard	0.8	1	µs
		Overdrive	0.8	4.8	µs
t_{ON4}	Active Time for Write 0 Recovery	Standard	4	12	µs
		Overdrive	0.8	1	µs

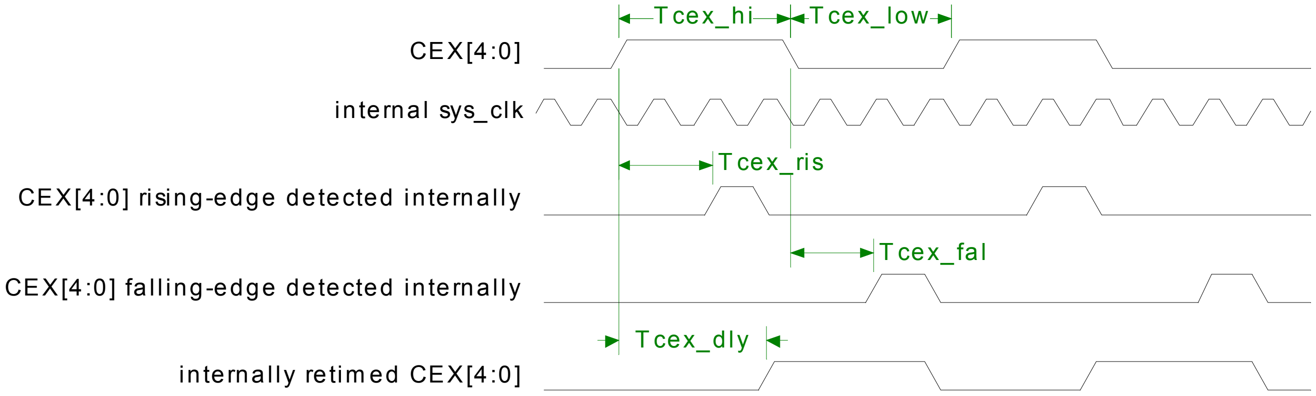
Figure 140: 1-Wire STPZ Reset and Read Write Timing Diagram and Table

5.4.6 Programmable Counter Array Interface Timing



Symbol	Description	Min	Typ	Max	Units
Teci_cycle	ECI cycle time	> 2	-	-	Tsys_clk[7]
Teci_pulse	ECI pulse width	> 1	-	-	Tsys_clk
Teci_dly	ECI internally retimed delay	2	-	3	Tsys_clk

Figure 141: ECI Timing Diagram and Table

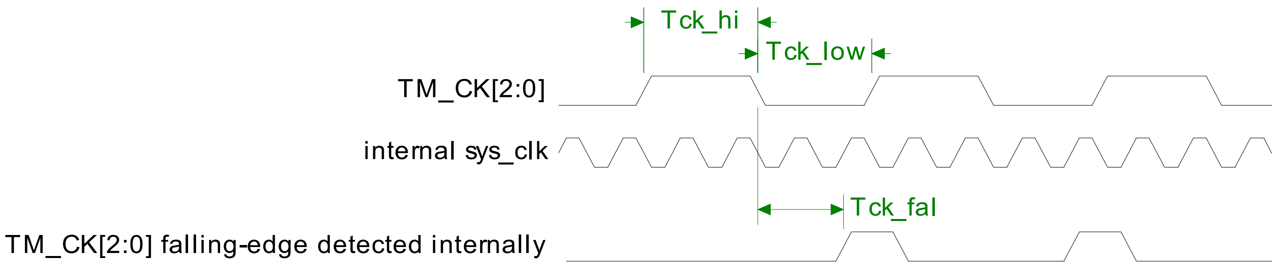

Tcex_hi
Tcex_low
CEX[4:0]
internal sys_clk
Tcex_ris
CEX[4:0] rising-edge detected internally
Tcex_fal
CEX[4:0] falling-edge detected internally
Tcex_dly
internally retimed CEX[4:0]

Symbol	Description	Min	Typ	Max	Units
Tcex_hi	CEX[4:0] (as input) high pulse width	1.5	-	-	Tsys_clk
Tcex_low	CEX[4:0] (as input) low pulse width	1.5	-	-	Tsys_clk
Tcex_ris	CEX[4:0] (as input) rising-edge internal detection time	1~2	-	2~3	Tsys_clk
Tcex_fal	CEX[4:0] (as input) falling-edge internal detection time	1~2	-	2~3	Tsys_clk
Tcex_dly	CEX[4:0] (as input) internally retimed delay	2	-	3	Tsys_clk

Figure 142: CEX[4:0] Timing Diagram and Table

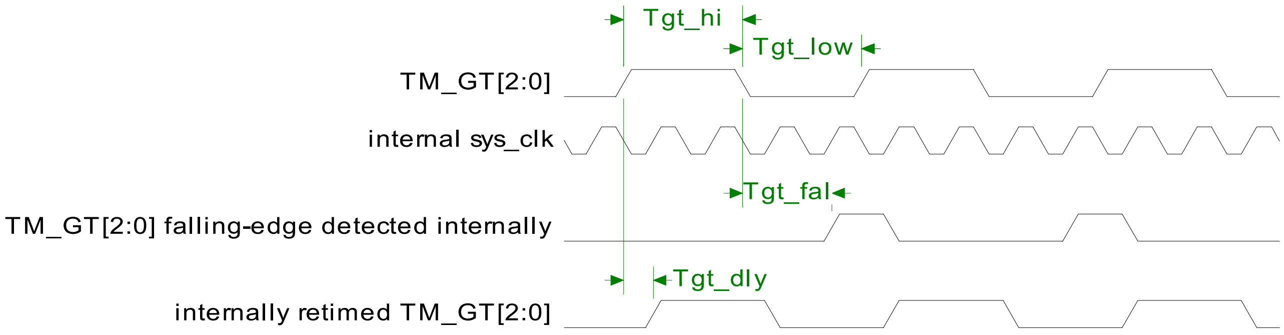
[7] Tsys_clk = 10/20/40ns for 100/50/25Mhz operating system clock.

5.4.7 Timer 0/1/2 Interface Timing



Symbol	Description	Min	Typ	Max	Units
Tck_hi	TM_CK[2:0] high pulse width	2	-	-	Tsys_clk
Tck_low	TM_CK[2:0] low pulse width	2	-	-	Tsys_clk
Tck_fal	TM_CK[2:0] falling-edge internal detection time	1~2	-	2	Tsys_clk

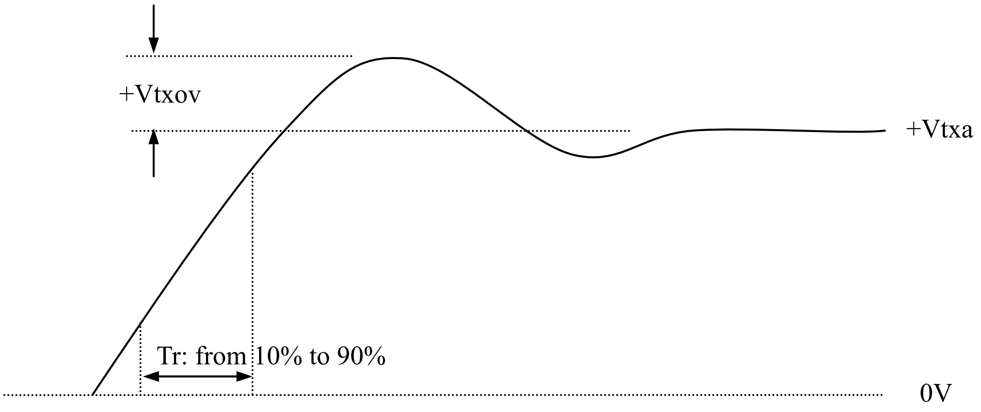
Figure 143: TM_CK[2:0] Timing Diagram and Table


Tgt_hi
Tgt_low
TM_GT[2:0]
internal sys_clk
Tgt_fal
TM_GT[2:0] falling-edge detected internally
Tgt_dly
internally retimed TM_GT[2:0]

Symbol	Description	Min	Typ	Max	Units
Tgt_hi	TM_GT[2:0] high pulse width	2	-	-	Tsys_clk
Tgt_low	TM_GT[2:0] low pulse width	2	-	-	Tsys_clk
Tgt_fal	TM_GT[2:0] falling-edge internal detection time	1~2	-	2	Tsys_clk
Tgt_dly	TM_GT[2:0] internally retimed delay	0.5	-	1	Tsys_clk

Figure 144: TM_GT[2:0] Timing Diagram and Table

5.4.8 10/100M Ethernet PHY Interface Timing



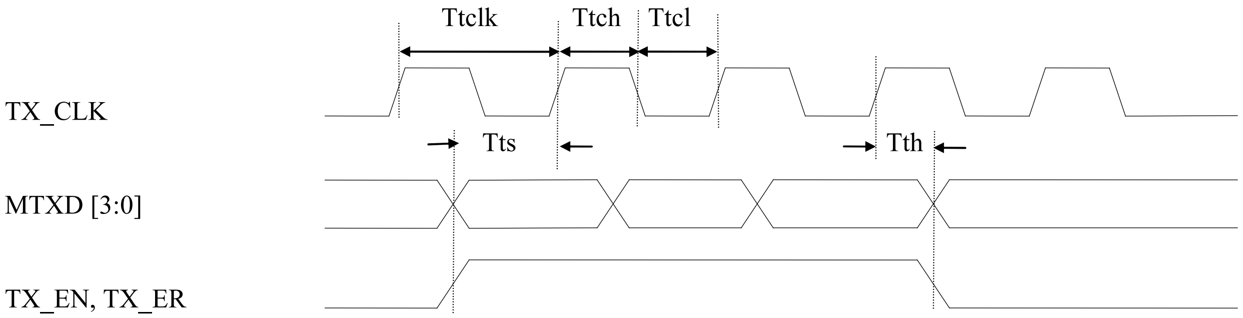
Symbol	Description	Condition	Min	Typ	Max	Units
	Peak-to-peak differential output voltage	10BASE-T mode	4.4	5	5.6	V
Vtxa *2	Peak-to-peak differential output voltage	100BASE-TX mode	1.9	2	2.1	V
Tr / Tf	Signal rise / fall time	100BASE-TX mode	3	4	5	ns
	Output jitter	100BASE-TX mode, scrambled idle signal	-	-	1.4	ns
Vtxov	Overshoot	100BASE-TX mode	-	-	5	%

Figure 145: 10/100M Ethernet PHY Transmitter Waveform and Spec

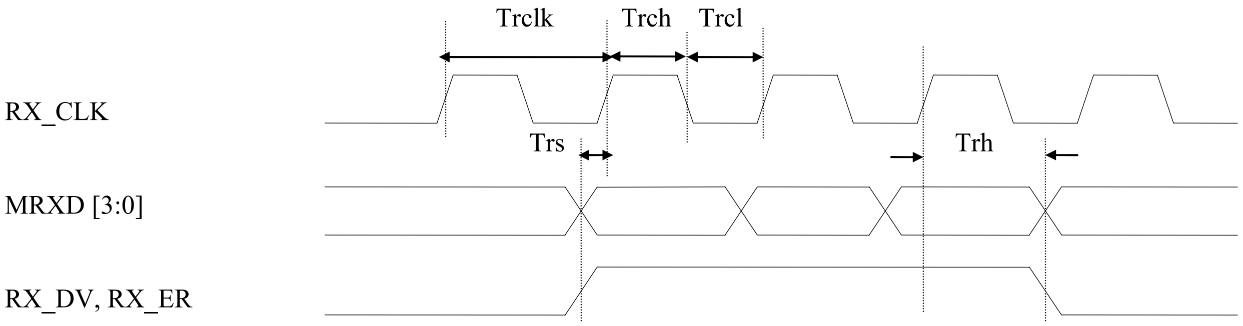
Symbol	Description	Condition	Min	Typ	Max	Units
	Receiver input impedance		10	-	-	KΩ
	Differential squelch voltage	10BASE-T mode	300	400	500	mV
	Common mode input voltage		2.97	3.3	3.63	V
	Maximum error-free cable length		100	-	-	meter

Table 58: 10/100M Ethernet PHY Receiver Spec

5.4.9 MII Timing



Symbol	Description	Min	Typ	Max	Units
Ttclk	TX_CLK clock cycle time *1	-	40.0	-	ns
Ttch	TX_CLK clock high time *2	-	20.0	-	ns
Ttcl	TX_CLK clock low time *2	-	20.0	-	ns
Tts	TXD [3:0], TX_EN, TX_ER setup time	28.0	-	-	ns
Tth	TXD [3:0], TX_EN, TX_ER hold time	4.5	-	11.5	ns

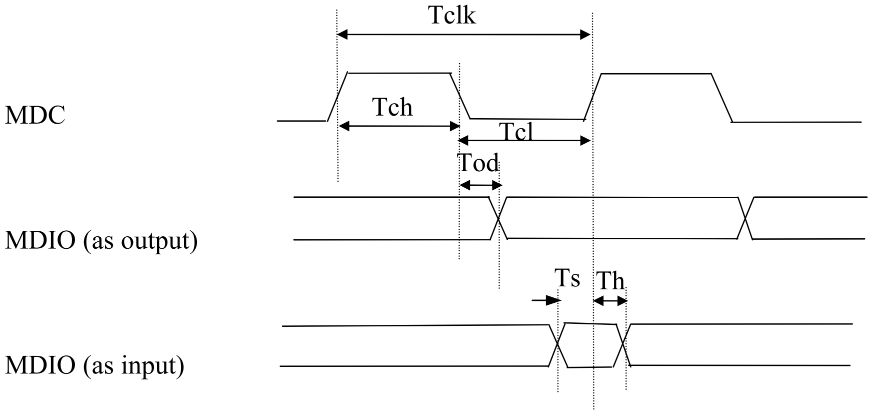


Symbol	Description	Min	Typ	Max	Units
Trclk	RX_CLK clock cycle time *1	-	40.0	-	ns
Trch	RX_CLK clock high time *2	-	20.0	-	ns
Trcl	RX_CLK clock low time *2	-	20.0	-	ns
Trs	RXD [3:0], RX_DV, and RX_ER setup time	3.0	-	-	ns
Trh	RXD [3:0], RX_DV, and RX_ER hold time	0.5	-	-	ns

*1: For 10Mbps, the typical value of Ttclk and Trclk shall scale to 400ns.

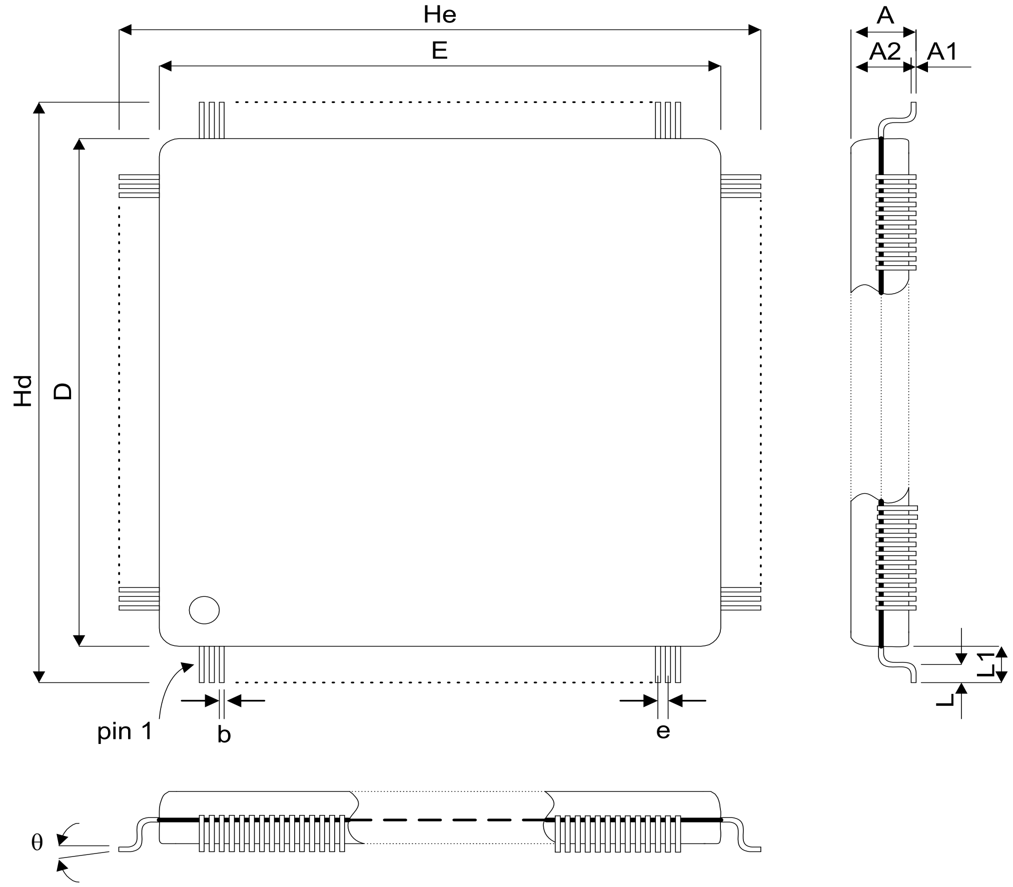
*2: For 10Mbps, the typical value of Ttch, Ttcl, Trch, and Trcl shall scale to 200ns.

5.4.10 Station Management Timing



Symbol	Description	Min	Typ	Max	Units
Tclk	MDC clock cycle time	-	640	-	ns
Tch	MDC clock high time	-	320	-	ns
Tcl	MDC clock low time	-	320	-	ns
Tod	MDC clock falling edge to MDIO output delay	0	-	2	ns
Ts	MDIO data input setup time	10	-	-	ns
Th	MDIO data input hold time	0	-	-	ns

6.0 Package Information



Symbol	Millimeter		
	Min	Typ	Max
A1	0.05	-	-
A2	1.35	1.40	1.45
A	-	-	1.60
b	0.13	0.18	0.23
D	13.90	14.00	14.10
E	13.90	14.00	14.10
e	-	0.4 BSC [8]	-
Hd	15.85	16.00	16.15
He	15.85	16.00	16.15
L	0.45	0.60	0.75
L1	-	1.00 REF	-
θ	0°	3.5°	7°

[8] BSC stands for Basic Spacing between Centers. Please refer to JEDEC Standard 95, page 4.17 for details.

7.0 Ordering Information

Part Number	Description
AX11025 LI	Lead free package, Industrial temperature range, -40 to 85˚C.

8.0 Revision History

Revision	Date	Comments
V1.0	2006/10/09	First release.
V1.1	2007/03/26	Updated section 5.2 power consumption for CPU in STOP, OSC/PLL stopped. Added Tbuf and Tsu_sto value in section 5.4.3 I2C interface timing. Corrected Iol and Ioh value in section 5.1.4 and 5.1.5. Added min value for T_{rise3} in section 5.3.
V1.2	2007/05/04	1. Corrected XTL25P pin type to O18 in section 1.4 Signal Description. 2. Removed T2IF in Table 4 SFR Register Map.
V1.3	2007/12/20	1. Added Θ_{JC} and Θ_{JA} data in section 5.2. 2. Added the device address (1010000b) information of the I2C Configuration EEPROM in section 3.1.
V1.04	2008/06/06	1. Add the “US Patent Pending” string in the Features page.
V1.05	2008/07/30	1. Update the protocol support information in the Features page.
V1.06	2008/09/15	1. Added more information into Section 4.6.5.
V1.07	2008/10/30	1. Updated the Trise3 timing information in Section 5.3.
V1.08	2009/11/26	1. Updated the SPI related timing decription. 2. Updated Table 9: On-chip Flash Memory Read Protection descption.
V1.09	2011/06/14	1. Added legal disclaimer description.

ASIX Electronics Corporation.

4F, No. 8, Hsin Ann Rd., HsinChu Science Park,
HsinChu, Taiwan, R.O.C.

TEL: 886-3-5799500
FAX: 886-3-5799558

Email: support@asix.com.tw
Web: http://www.asix.com.tw