NS9750 Hardware Reference

Making DEVICE NETWORKING easy™

90000624_G

NS9750 Hardware Reference

Part number/version: 90000624_G
Release date: March 2008
www.digiembedded.com

©2008 Digi International Inc.
Printed in the United States of America. All rights reserved.

Digi, Digi International, the Digi logo, the Making Device Networking Easy logo, NetSilicon, a Digi International Company, NET+, NET+OS and NET+Works are trademarks or registered trademarks of Digi International, Inc. in the United States and other countries worldwide. All other trademarks are the property of their respective owners.

Information in this document is subject to change without notice and does not represent a committment on the part of Digi International.

Digi provides this document "as is," without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of, fitness or merchantability for a particular purpose. Digi may make improvements and/or changes in this manual or in the product(s) and/or the program(s) described in this manual at any time.

This product could include technical inaccuracies or typographical errors. Changes are made periodically to the information herein; these changes may be incorporated in new editions of the publication.

Digi International
11001 Bren Road East
Minnetonka, MN 55343 U.S.A.
United States: +1 877 912-3444
Other locations: +1 952 912-3444

www.digiembedded.com

Contents

Chapter 1: About NS9750 1

NS9750 Features 2

System-level interfaces 8

System boot 10

Reset 10

RESET_DONE as an input 11

RESET_DONE as an output 11

System clock 13

USB clock 15

Chapter 2: NS9750 Pinout 17

Pinout and signal descriptions 18

System Memory interface 18

System Memory interface signals 22

Ethernet interface 25

Clock generation/system pins 26

bist_en_n, pll_test_n, and scan_en_n 28

PCI interface 28

GPIO MUX 34

LCD module signals 42

I2C interface 43

USB interface 43

JTAG interface for ARM core/boundary scan 43

Reserved 45

Power ground 46

Chapter 3: Working with the CPU 47

About the processor 48

Instruction sets.... 49

ARM instruction set 50

Thumb instruction set.... 50

Java instruction set 50

System control processor (CP15) registers.... 51

ARM926EJ-S system addresses 51

Accessing CP15 registers.... 52

Terms and abbreviations 52

Register summary.... 53

R0: ID code and cache type status registers.... 55

R1: Control register 58

R2: Translation Table Base register.... 61

R3: Domain Access Control register.... 61

R4 register 62

R5: Fault Status registers.... 62

R6: Fault Address register 64

R7: Cache Operations register 64

R8:TLB Operations register.... 68

R9: Cache Lockdown register 69

R10: TLB Lockdown register.... 73

R11 and R12 registers 74

R13: Process ID register.... 75

R14 register.... 77

R15: Test and debug register 77

Jazelle (Java) 77

DSP.... 78

Memory Management Unit (MMU) 78

MMU Features 78

Address translation 81

MMU faults and CPU aborts.... 95

Domain access control 98

Fault checking sequence 99

External aborts....102

Enabling the MMU....103

Disabling the MMU 104

TLB structure 104
Caches and write buffer 105
Cache features 105
Write buffer 106
Enabling the caches 107
Cache MVA and Set/Way formats 109
Noncachable instruction fetches 111
Self-modifying code 112
AHB behavior 112
Instruction Memory Barrier 113
IMB operation 113
Sample IMB sequences 114

Chapter 4: System Control Module 115
System Control Module features 116
Bus interconnection 116
System bus arbiter 116
Arbiter configuration examples 120
Address decoding 123
Programmable timers 125
Software watchdog timer 125
General purpose timers/counters 125
Interrupt controller 129
Vectored interrupt controller (VIC) flow 132
System attributes 133
PLL configuration 133
Bootstrap initialization 134
System configuration registers 138
AHB Arbiter Gen Configuration register 144
BRC0, BRC1, BRC2, and BRC3 registers 145
Timer 0–15 Reload Count registers 146
Timer 0–15 Read register 147
Interrupt Vector Address Register Level 0–31 147
Int (Interrupt) Config (Configuration) registers (0–31) 148
ISRADDR register 150
Interrupt Status Active 151

Interrupt Status Raw 152
Timer Interrupt Status register 153
Software Watchdog Configuration register 153
Software Watchdog Timer register 155
Clock Configuration register 155
Reset and Sleep Control register 157
Miscellaneous System Configuration and Status register 158
PLL Configuration register....161
Active Interrupt Level Status register 163
Timer 0–15 Control registers 163
System Memory Chip Select 0 Dynamic Memory Base and Mask registers.. 165
System Memory Chip Select 1 Dynamic Memory Base and Mask registers.. 166
System Memory Chip Select 2 Dynamic Memory Base and Mask registers.. 167
System Memory Chip Select 3 Dynamic Memory Base and Mask registers.. 168
System Memory Chip Select 0 Static Memory Base and Mask registers.169
System Memory Chip Select 1 Static Memory Base and Mask registers.170
System Memory Chip Select 2 Static Memory Base and Mask registers.171
System Memory Chip Select 3 Static Memory Base and Mask registers.172
Gen ID register 173
External Interrupt 0–3 Control register....175

Chapter 5: Memory Controller 177

Features....178
System overview 179
Low-power operation 180
Memory map....180
Static memory controller....183
Write protection 184
Extended wait transfers 184
Memory mapped peripherals....185
Static memory initialization 185
Byte lane control 211
Address connectivity 212
Byte lane control and databus steering 216

Dynamic memory controller 224
Write protection 224
Access sequencing and memory width 224
Address mapping 225
Registers 264
Register map 264
Reset values 266
Control register 267
Status register 269
Configuration register 269
Dynamic Memory Control register 270
Dynamic Memory Refresh Timer register 272
Dynamic Memory Read Configuration register 274
Dynamic Memory Precharge Command Period register 275
Dynamic Memory Active to Precharge Command Period register 276
Dynamic Memory Self-refresh Exit Time register 277
Dynamic Memory Last Data Out to Active Time register 278
Dynamic Memory Data-in to Active Command Time register 279
Dynamic Memory Write Recovery Time register 280
Dynamic Memory Active to Active Command Period register 281
Dynamic Memory Auto Refresh Period register 282
Dynamic Memory Exit Self-refresh register 283
Dynamic Memory Active Bank A to Active Bank B Time register 284
Dynamic Memory Load Mode register to Active Command Time register .. 285
Static Memory Extended Wait register 286
Dynamic Memory Configuration 0–3 registers 287
Dynamic Memory RAS and CAS Delay 0–3 registers 291
Static Memory Configuration 0–3 registers 292
Static Memory Write Enable Delay 0–3 registers 296
Static Memory Output Enable Delay 0–3 registers 297
Static Memory Read Delay 0–3 registers 298
Static Memory Page Mode Read Delay 0–3 registers 299
Static Memory Write Delay 0–3 registers 300
Static Memory Turn Round Delay 0–3 registers 301

Chapter 6: Ethernet Communication Module 315

Overview 316
Ethernet MAC 317
    Station address logic (SAL) 321
    Statistics module 321
Ethernet front-end module 323
    Receive packet processor 324
    Transmit packet processor 327
    Ethernet Slave Interface 330
    Interrupts 331
    Resets 332
External CAM filtering 334
Ethernet Control and Status registers 337
    Ethernet General Control Register #1 339
    Ethernet General Control Register #2 342
    Ethernet General Status register 344
    Ethernet Transmit Status register 344
    Ethernet Receive Status register 347
    MAC Configuration Register #1 348
    MAC Configuration Register #2 351
    Back-to-Back Inter-Packet-Gap register 354
    Non Back-to-Back Inter-Packet-Gap register 355
    Collision Window/Retry register 355
    Maximum Frame register 357
    PHY Support register 358
    MII Management Configuration register 359
    MII Management Command register 360
    MII Management Address register 361
    MII Management Write Data register 362
    MII Management Read Data register 363
    MII Management Indicators register 363
    Station Address registers 364
    Station Address Filter register 366
    Register Hash Tables 366
    Statistics registers 368
    RX_A Buffer Descriptor Pointer register 383
    RX_B Buffer Descriptor Pointer register 383
    RX_C Buffer Descriptor Pointer register 384

RX_D Buffer Descriptor Pointer register 384
Ethernet Interrupt Status register 385
Ethernet Interrupt Enable register...387
TX Buffer Descriptor Pointer register..389
Transmit Recover Buffer Descriptor Pointer register 389
TX Error Buffer Descriptor Pointer register.................................390
RX_A Buffer Descriptor Pointer Offset register 391
RX_B Buffer Descriptor Pointer Offset register 392
RX_C Buffer Descriptor Pointer Offset register 393
RX_D Buffer Descriptor Pointer Offset register 393
Transmit Buffer Descriptor Pointer Offset register........................394
RX Free Buffer register 395
TX buffer descriptor RAM..396
Sample hash table code 397

Chapter 7: PCI-to-AHB Bridge 403
About the PCI-to-AHB Bridge 404
PCI-to-AHB bridge functionality 405
Cross-bridge transaction error handling.....................................407
AHB address decoding and translation 408
PCI address decoding and mapping 408
Interrupts 409
Transaction ordering 410
Endian configuration 411
Configuration registers...411
Bridge Configuration registers 413
PCI bus arbiter 418
PCI arbiter functional description..419
Slave interface 420
PCI Arbiter Configuration registers 420
PCI system configurations 456
Device selection for configuration 458
PCI interrupts...458
PCI central resource functions...458
CardBus Support 461
Configuring NS9750 for CardBus support 463
CardBus adapter requirements 464

CardBus interrupts 465

Chapter 8: BBus Bridge 467

BBus bridge functions 468
Bridge control logic 469
DMA accesses 471
BBus control logic 472
BBus bridge masters and slaves....................472
Cycles and BBus arbitration 473
BBus peripheral address map (decoding) 473
Two-channel AHB DMA controller (AHB bus) 474
DMA buffer descriptor....................474
Descriptor list processing....................476
Peripheral DMA read access 477
Peripheral DMA write access 478
Peripheral REQ signaling....................479
Design Limitations 480
Calculating AHB DMA response latency....................480
Static RAM chip select configuration 482
Interrupt aggregation 483
Bandwidth requirements 483
SPI-EEPROM boot logic 484
Serial Channel B configuration 485
Memory Controller configuration....................486
SDRAM boot algorithm 488
BBus Bridge Control and Status registers 490
Buffer Descriptor Pointer register 491
DMA Channel 1/2 Control register 491
DMA Status and Interrupt Enable register....................494
DMA Peripheral Chip Select register 496
BBus Bridge Interrupt Status register 498
BBus Bridge Interrupt Enable register 499

Chapter 9: BBus DMA Controller 501

About the BBus DMA controllers....................502
DMA context memory 503

DMA buffer descriptor 504
DMA channel assignments 509
DMA Control and Status registers 510
- DMA Buffer Descriptor Pointer 512
- DMA Control register 514
- DMA Status/Interrupt Enable register 516

Chapter 10: BBus Utility 521
BBus Utility Control and Status registers 522
- Master Reset register 523
- GPIO Configuration registers 524
- GPIO Control registers 529
- GPIO Status registers 532
- BBus Monitor register 535
- BBus DMA Interrupt Status register 536
- BBus DMA Interrupt Enable register 537
- USB Configuration register 538
- Endian Configuration register 539
- ARM Wake-up register 541

Chapter 11: I2C Master/Slave Interface 543
Overview 544
- Physical I2C bus 544

I2C external addresses 545
I2C command interface 545
- Locked interrupt driven mode 546
- Master module and slave module commands 546
- Bus arbitration 547

I2C registers 547
- Command Transmit Data register 548
- Status Receive Data register 549
- Master Address register 550
- Slave Address register 551
- Configuration register 552

Interrupt Codes 553
Software driver 555

Flow charts 556
Master module (normal mode, 16-bit) 556
Slave module (normal mode, 16-bit) 557

Chapter 12: LCD Controller 559

LCD features 560
Programmable parameters 560
LCD panel resolution 561
LCD panel support 561
Number of colors 562
LCD power up and power down sequence support 563
LCD controller functional overview 564
Clocks 565
Signals and interrupts 566
AHB interface 568
AHB master and slave interfaces 568
Dual DMA FIFOs and associated control logic 568
Pixel serializer 569
RAM palette 573
Grayscaler 574
Upper and lower panel formatters 574
Panel clock generator 574
Timing controller 574
Generating interrupts 575
External pad interface signals 575
LCD panel signal multiplexing details 575
Registers 579
LCDTiming0 580
LCDTiming1 582
LCDTiming2 register 583
LCDTiming3 587
LCDUPBASE and LCDLPBASE 587
LCDINTRENABLE 589
LCDControl register 590
LCDStatus register 593
LCDInterrupt register 594
LCDUPCURR and LCDLPCURR 594

LCDPalette register 595
Interrupts 598
MBERRORINTR — Master bus error interrupt 598
VCOMPINTR — Vertical compare interrupt 598
LBUINTR — Next base address update interrupt 599

Chapter 13: Serial Control Module: UART 601
Features 602
Bit-rate generator 603
UART mode 604
FIFO management 605
Transmit FIFO interface 605
Receive FIFO interface 606
Serial port performance 608
Serial port control and status registers 608
Serial Channel B/A/C/D Control Register A 611
Serial Channel B/A/C/D Control Register B 614
Serial Channel B/A/C/D Status Register A 617
Serial Channel B/A/C/D Bit-rate register 624
Serial Channel B/A/C/D FIFO Data register 629
Serial Channel B/A/C/D Receive Buffer GAP Timer 630
Serial Channel B/A/C/D Receive Character GAP Timer 632
Serial Channel B/A/C/D Receive Match register 634
Serial Channel B/A/C/D Receive Match MASK register 635
Serial Channel B/A/C/D Flow Control register 636
Serial Channel B/A/C/D Flow Control Force register 638

Chapter 14: Serial Control Module: SPI 643
Features 644
Bit-rate generator 645
SPI mode 646
SPI modes 646
FIFO management 647
Transmit FIFO interface 647
Receive FIFO interface 648
Serial port performance 650

Serial port control and status registers 650
Serial Channel B/A/C/D Control Register A 652
Serial Channel B/A/C/D Control Register B 655
Serial Channel B/A/C/D Status Register A 657
Serial Channel B/A/C/D Bit-rate register 660
Serial Channel B/A/C/D FIFO Data register 665

Chapter 15: IEEE 1284 Peripheral Controller 669
Requirements 670
Overview 670
Compatibility mode 671
Nibble mode 672
Byte mode 672
ECP mode 673
Data and command FIFOs....675
IEEE 1284 negotiation 676
BBus slave and DMA interface 677
BBus slave and DMA interface register map 677
IEEE 1284 General Configuration register 679
Interrupt Status and Control register 681
FIFO Status register 684
Forward Command FIFO Read register 686
Forward Data FIFO Read register 687
Reverse FIFO Write register/Reverse FIFO Write Register — Last 687
Forward Command DMA Control register 689
Forward Data DMA Control register 690
Printer Data Pins register 691
Port Status register, host 692
Port Control register 693
Port Status register, peripheral 694
Feature Control Register A 694
Feature Control Register B 695
Interrupt Enable register 695
Master Enable register 697
Extensibility Byte Requested by Host 698
Extended Control register 698
Interrupt Status register 699

Pin Interrupt Mask register 700
Pin Interrupt Control register..701
Granularity Count register 702
Forward Address register 703
Core Phase (IEEE1284) register 704

Chapter 16: USB Controller Module 707

Overview 708
USB module architecture 708
USB device block..710
Control and status 710
Packet and data flow...711
Logical and physical endpoints 712
Slew rates 712
Host block...712
Control and status 712
Packet data flow 713
USB device endpoint 714
Transmission error handling..714
Handling USB-IN packet errors..715
Handling USB-OUT packet errors 715
USB block registers 716
USB Global registers..716
Global Control and Status register 717
Device Control and Status register...718
Global Interrupt Enable register 720
Global Interrupt Status register 721
Device IP Programming Control/Status register 724
USB host block registers 725
Reserved bits 725
USB host block register address map 725
HCRevision register 726
HcControl register 727
HcCommandStatus register 730
HcInterruptStatus register..733
HcInterruptEnable register 735
HcInterruptDisable register 737

HcHCCA register 739
HcPeriodCurrentED register 740
HcControlHeadED register 741
HcControlCurrentED register 742
HcBulkHeadED register 743
HcBulkCurrentED register 744
HcDoneHead register 746
HcFmInterval register 747
HcFmRemaining register 748
HcFmNumber register 749
HcPeriodicStart register 750
HcLsThreshold register 751
Root hub partition registers 752
HcRhDescriptorA register 753
HcRhDescriptorB register 755
HcRhStatus register 756
HcRhPortStatus[1] register 759
USB Device Block registers 765
Device Descriptor/Setup Command register 765
Endpoint Descriptor #0–#11 registers 766
USB Device Endpoint FIFO Control and Data registers 767
FIFO Interrupt Status registers 769
FIFO Interrupt Enable registers 776
FIFO Packet Control registers 780
FIFO Status and Control registers 781

Chapter 17: Timing 787
Electrical characteristics 788
Absolute maximum ratings 788
Recommended operating conditions 788
Maximum power dissipation 789
Typical power dissipation 789
DC electrical characteristics 790
Inputs 790
Outputs 791
Reset and edge sensitive input timing requirements 792
Power sequencing 794

Memory timing 795
- SDRAM burst read (16-bit) 796
- SDRAM burst read (16-bit), CAS latency = 3 797
- SDRAM burst write (16-bit) 798
- SDRAM burst read (32-bit) 799
- SDRAM burst read (32-bit), CAS latency = 3 800
- SDRAM burst write (32-bit) 801
- SDRAM load mode 802
- SDRAM refresh mode 803
- Clock enable timing 803
- Static RAM read cycles with 0 wait states 805
- Static RAM asynchronous page mode read, WTPG = 1 806
- Static RAM read cycle with configurable wait states 807
- Static RAM sequential write cycles 808
- Static RAM write cycle 809
- Static write cycle with configurable wait states....810
- Slow peripheral acknowledge timing 811

Ethernet timing 813
- Ethernet MII timing 814
- Ethernet RMII timing 815

PCI timing....816
- Internal PCI arbiter timing 818
- PCI burst write from NS9750 timing 818
- PCI burst read from NS9750 timing 819
- PCI burst write to NS9750 timing....819
- PCI burst read to NS9750 timing....820
- PCI clock timing....820

I2C timing....821

LCD timing 822
- Horizontal timing for STN displays 824
- Vertical timing for STN displays 825
- Horizontal timing for TFT displays 825
- Vertical timing for TFT displays 825
- HSYNC vs VSYNC timing for STN displays....826
- HSYNC vs VSYNC timing for TFT displays....826
- LCD output timing 826

SPI timing 827

SPI master mode 0 and 1: 2-byte transfer 829
SPI master mode 2 and 3: 2-byte transfer 829
SPI slave mode 0 and 1: 2-byte transfer 830
SPI slave mode 2 and 3: 2-byte transfer 830
IEEE 1284 timing 831
IEEE 1284 timing example 831
USB timing 832
USB differential data timing 833
USB full speed load timing 833
USB low speed load...834
Reset and hardware strapping timing 835
JTAG timing 836
Clock timing 837
USB crystal/external oscillator timing 837
LCD input clock timing..838
System PLL bypass mode timing..839

Chapter 18: Packaging 841
Product specifications 845

Using This Guide

Review this section for basic information about the guide you are using, as well as general support and contact information. This printed version of the *NS9750 Hardware Reference, Rev. E* includes two volumes (90000622_E and 90000623_E). A single PDF (90000624_E) is included on your documentation CD.

About this guide

This guide provides information about the Digi NS9750, a single chip 0.13μm CMOS network-attached processor. The NS9750 is part of the Digi NET+ARM family of devices.

The NET+ARM family is part of the NET+Works integrated product family, which includes the NET+OS network software suite.

Who should read this guide

This guide is for hardware developers, system software developers, and applications programmers who want to use the NS9750 for development.

To complete the tasks described in this guide, you must:

- Understand the basics of hardware and software design, operating systems, and microprocessor design.
- Understand the NS9750 architecture.

What's in this guide

This table shows where you can find specific information in the printed guides.

To read about	See	Vol
NS9750 key features	Chapter 1, "About the NS9750	1
NS9750 ball grid array assignments	Chapter 2, "NS9750 Pinout"	1
NS9750 CPU	Chapter 3, "Working with the CPU"	1
System functionality	Chapter 4, "System Control Module"	1
How the NS9750 works with the Multiport Memory Controller, an AMBA-compliant SoC peripheral	Chapter 5, "Memory Controller"	1
How the NS9750 works with Ethernet MAC and Ethernet front-end module	Chapter 6, "Ethernet Communication Module"	1
PCI-to-AHB bus functionality, which connects PCI-based devices to the NS9750 AHB bus	Chapter 7, "PCI-to-AHB Bridge"	1
Digi proprietary BBus	Chapter 8, "BBus Bridge	2
NS9750 BBus DMA controller subsystem	Chapter 9, "BBus DMA Controller"	2
Chip-level support for low-speed peripherals	Chapter 10, "BBus Utility"	2
Interface between the ARM CPU and the I^{2}C bus	Chapter 11, "I^{2}C Master/Slave Interface"	2
LCD controller	Chapter 12, "LCD Controller"	2
UART mode serial controller	Chapter 13, "Serial Control Module: UART"	2
SPI mode serial controller	Chapter 14, "Serial Control Module: SPI"	2
IEEE 1284 peripheral port	Chapter 15, "IEEE 1284 Peripheral Controller"	2
USB 2.0	Chapter 16, "USB Controller Module"	2
NS9750 electrical characteristics and timing diagrams and information	Chapter 17, "Timing"	2
NS9750 packaging information	Chapter 18, "Packaging"	2

Conventions used in this guide

This table describes the typographic conventions used in this guide:

This convention	Is used for
italic type	Emphasis, new terms, variables, and document titles.
monospaced type	Filenames, pathnames, and code examples.
_ (underscore)	Defines a signal as being active low.
‘b	Indicates that the number following this indicator is in binary radix
‘d	Indicates that the number following this indicator is in decimal radix
‘h	Indicates that the number following this indicator is in hexadecimal radix
RW1TC	Indicates Read/Write 1 to clear.

Related documentation

- *NS9750 Jumpers and Components* provides a hardware description of the NS9750 development board, and includes information about jumpers, components, switches, and configuration.
- *NS9750 Sample Driver Configurations* provides sample configurations that you can use to develop your drivers.

Review the documentation CD-ROM that came with your development kit for information on third-party products and other components.

See the NET+OS software documentation for information appropriate to the chip you are using.

Documentation updates

Digi occasionally provides documentation updates on the Web site.

Be aware that if you see differences between the documentation you received in your package and the documentation on the Web site, the Web site content is the latest version.

Customer support

To get help with a question or technical problem with this product, or to make comments and recommendations about our products or documentation, use the contact information listed in this table:

For	Contact information
Technical support	United States: +1 877 912-3444 Other locations: +1 952 912-3444 www.digiembedded.com

About NS9750

CHAPTER 1

The Digi NS9750 is a single chip 0.13μm CMOS network-attached processor. This chapter provides an overview of the NS9750, which is based on the standard architecture in the NET+ARM family of devices.

NS9750 Features

The NS9750 uses an ARM926EJ-S core as its CPU, with MMU, DSP extensions, Jazelle Java accelerator, and 8 kB of instruction cache and 4 kB of data cache in a Harvard architecture. The NS9750 runs up to 200 MHz, with a 100 MHz system and memory bus and 50 MHz peripheral bus. The NS9750 offers an extensive set of I/O interfaces and Ethernet high-speed performance and processing capacity. The NS9750 is designed specifically for use in high-performance intelligent networked devices and Internet appliances including high-performance, low-latency remote I/O, intelligent networked information displays, and streaming and surveillance cameras.

32-bit ARM926EJ-S RISC processor

- 125 to 200 MHz
- 5-stage pipeline with interlocking
- Harvard architecture
- 8 kB instruction cache and 4 kB data cache
- 32-bit ARM and 16-bit Thumb instruction sets. Can be mixed for performance/code density tradeoffs.
- MMU to support virtual memory-based OSs, such as Linux, VxWorks, others
- DSP instruction extensions, improved divide, single cycle MAC
- ARM Jazelle, 1200CM (coffee marks) Java accelerator
- EmbeddedICE-RT debug unit
- JTAG boundary scan, BSDL support

External system bus interface

- 32-bit data, 32-bit internal address bus, 28-bit external address bus
- Glueless interface to SDRAM, SRAM, EEPROM, buffered DIMM, Flash
- 4 static and 4 dynamic memory chip selects
- 1-32 wait states per chip select

  A shared Static Extended Wait register allows transfers to have up to 16368 wait states that can be externally terminated
- Self-refresh during system sleep mode
- Automatic dynamic bus sizing to 8 bits, 16 bits, 32 bits

- Burst mode support with automatic data width adjustment
- Two external DMA channels for external peripheral support

System Boot

- High-speed boot from 8-bit, 16-bit, or 32-bit ROM or Flash
- Hardware-supported low cost boot from serial EEPROM through SPI port (patent pending)

High performance 10/100 Ethernet MAC

- 10/100 Mbps MII/RMII PHY interfaces
- Full-duplex or half-duplex
- Station, broadcast, or multicast address filtering
- 2 kB RX FIFO
- 256-byte TX FIFO with on-chip buffer descriptor ring
  - Eliminates underruns and decreases bus traffic
- Separate TX and RX DMA channels
- Intelligent receive-side buffer size selection
- Full statistics gathering support
- External CAM filtering support

PCI/CardBus port

- PCI v2.2, 32-bit bus, up to 33 MHz bus speed
- Programmable to:
  - PCI device mode
  - PCI host mode:

    Supports up to 3 external PCI devices

    Embedded PCI arbiter or external arbiter
- CardBus host mode

Flexible LCD controller

- Supports most commercially available displays:
  - Active Matrix color TFT displays:
    Up to 24bpp direct 8:8:8 RGB; 16 colors
  - Single and dual panel color STN displays:
    Up to 16bpp 4:4:4 RGB; 3375 colors
  - Single and dual panel monochrome STN displays:
    1, 2, 4bpp palettized gray scale
- Formats image data and generates timing control signals
- Internal programmable palette LUT and grayscaler support different color techniques
- Programmable panel-clock frequency

USB ports

- USB v.2.0 full speed (12 Mbps) and low speed (1.5 Mbps)
- Configurable to device or OHCI host
  - USB host is bus master
  - USB device supports one bidirectional control endpoint and 11 unidirectional endpoints
- All endpoints supported by a dedicated DMA channel; 13 channels total
- 20 byte RX FIFO and 20 byte TX FIFO

Serial ports

- 4 serial modules, each independently configurable to UART mode, SPI master mode, or SPI slave mode
- Bit rates from 75 bps to 921.6 kbps: asynchronous x16 mode
- Bit rates from 1.2 kbps to 6.25 Mbps: synchronous mode
- UART provides:
  - High-performance hardware and software flow control
  - Odd, even, or no parity
  - 5, 6, 7, or 8 bits
  - 1 or 2 stop bits
  - Receive-side character and buffer gap timers

- Internal or external clock support, digital PLL for RX clock extraction
- 4 receive-side data match detectors
- 2 dedicated DMA channels per module, 8 channels total
- 32 byte TX FIFO and 32 byte RX FIFO per module

I^{2}C port

- I^{2}C v.1.0 configurable to master or slave mode
- Bit rates: fast (400 kHz) or normal (100 kHz) with clock stretching
- 7-bit and 10-bit address modes
- Supports I^{2}C bus arbitration

1284 parallel peripheral port

- All standard modes: ECP, byte, nibble, compatibility (also known as SPP or "Centronix")
- RLE (run length encoding) decoding of compressed data in ECP mode
- Operating clock from 100 kHz to 2 MHz

High performance multiple-master/distributed DMA system

- Intelligent bus bandwidth allocation (patent pending)
- System bus and peripheral bus

System bus

- Every system bus peripheral is a bus master with a dedicated DMA engine

Peripheral bus

- One 13-channel DMA engine supports USB device
  - 2 DMA channels support control endpoint
  - 11 DMA channels support 11 endpoints
- One 12-channel DMA engine supports:
  - 4 serial modules (8 DMA channels)
  - 1284 parallel port (4 DMA channels)
- All DMA channels support fly-by mode

External peripheral

- One 2-channel DMA engine supports external peripheral connected to memory bus

- Each DMA channel supports memory-to-memory transfers

Power management (patent pending)

- Power save during normal operation
  - Disables unused modules
- Power save during sleep mode
  - Sets memory controller to refresh
  - Disables all modules except selected wakeup modules
  - Wakeup on valid packets or characters

Vector interrupt controller

- Decreased bus traffic and rapid interrupt service
- Hardware interrupt prioritization

General purpose timers/counters

- 16 independent 16-bit or 32-bit programmable timers or counters
  - Each with an I/O pin
- Mode selectable into:
  - Internal timer mode
  - External gated timer mode
  - External event counter
- Can be concatenated
- Resolution to measure minute-range events
- Source clock selectable: internal clock or external pulse event
- Each can be individually enabled/disabled

System timers

- Watchdog timer
- System bus monitor timer
- System bus arbiter timer
- Peripheral bus monitor timer

General purpose I/O

- 50 programmable GPIO pins (muxed with other functions)
- Software-readable powerup status registers for every pin for customer-defined bootstrapping

External interrupts

- 4 external programmable interrupts
  - Rising or falling edge-sensitive
  - Low level- or high level-sensitive

Clock generator

- Low cost external crystal
- On-chip phase locked loop (PLL)
- Software programmable PLL parameters
- Optional external oscillator
- Separate PLL for USB

Operating grades/Ambient temperatures

- 200 MHz: 0 – 70^{o} C
- 162 MHz: -40 – +85^{o} C
- 125 MHz: 0 – 70^{o} C

System-level interfaces

Figure 1 shows the NS9750 system-level interfaces.

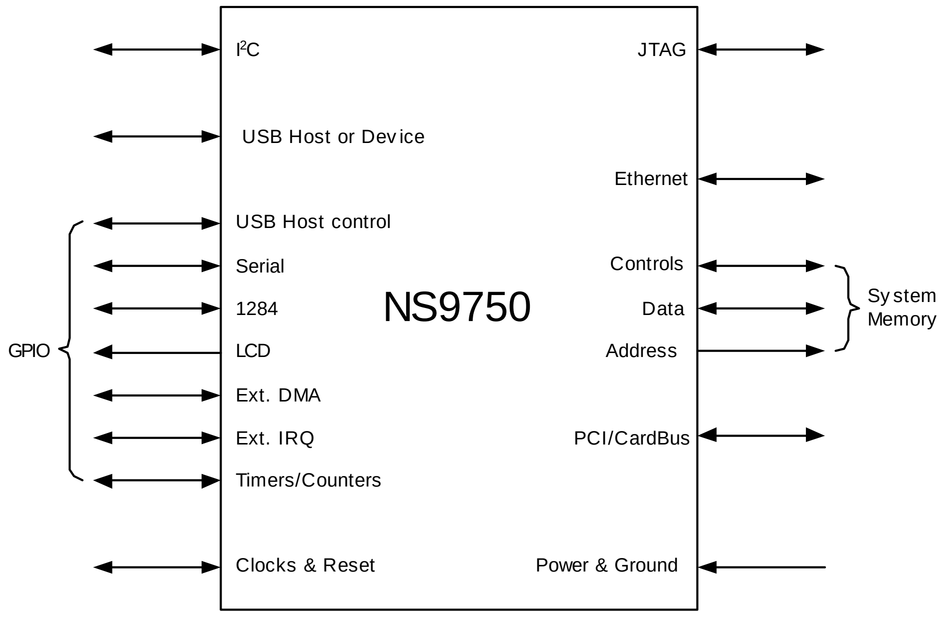


Figure 1: System-level hardware interfaces

- Ethernet MII/RMII interface to external PHY
- System memory interface
  - Glueless connection to SDRAM
  - Glueless connection to buffered PC100 DIMM
  - Glueless connection to SRAM
  - Glueless connection to Flash memory or ROM
- PCI muxed with CardBus interface
- USB host or device interface
- I^{2}C interface
- 50 GPIO pins muxed with:
  - Four 8-pin-each serial ports, each programmable to UART or SPI

- 1284 port
- Up to 24-bit TFT or STN color and monochrome LCD controller
- Two external DMA channels
- Four external interrupt pins programmed to rising or falling edge, or to high or low level
- Sixteen 16-bit or 32-bit programmable timers or counters
- Two control signals to support USB host

- JTAG development interface
- Clock interfaces for crystal or external oscillator
  - System clock
  - USB clock
- Clock interface for optional LCD external oscillator
- Power and ground

System boot

There are two ways to boot the NS9750 system (see Figure 2):

- From a fast Flash over the system memory bus
- From an inexpensive, but slower, serial EEPROM through SPI port B.

Both boot methods are glueless. The bootstrap pin, RESET_DONE, indicates where to boot on a system powerup. Flash boot can be done from 8-bit, 16-bit, or 32-bit ROM or Flash.

Serial EEPROM boot is supported by NS9750 hardware. A configuration header in the EEPROM specifies total number of words to be fetched from EEPROM, as well as a system memory configuration and a memory controller configuration. The boot engine configures the memory controller and system memory, fetches data from low-cost serial EEPROM, and writes the data to external system memory, holding the CPU in reset.

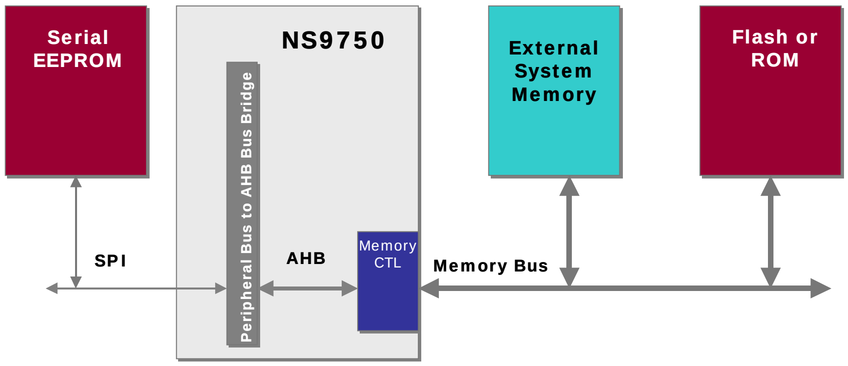


Figure 2: Two methods of booting NS9750 system

Reset

Master reset using an external reset pin resets NS9750. Only the AHB bus error status registers retain their values; software read resets these error status registers. The input reset pin can be driven by a system reset circuit or a simple power-on reset circuit.

RESET_DONE as an input

Used at bootup only:

- When set to 0, the system boots from SDRAM through the serial SPI EEPROM.
- When set to 1, the system boots from Flash/ROM. This is the default.

RESET_DONE as an output

Sets to 1, per Step 6 in the boot sequence.

If the system is booting from serial EEPROM through the SPI port, the boot program must be loaded into the SDRAM before the CPU is released from reset. The memory controller is powered up with dy_cs_n[0] enabled with a default set of SDRAM configurations. The default address range for dy_cs_n[0] is from 0x0000 0000. The other chip selects are disabled.

Boot sequence

1. When the system reset turns to inactive, the reset signal to the CPU is still held active.
2. An I/O module on the peripheral bus (BBus) reads from a serial ROM device that contains the memory controller settings and the boot program.
3. The BBus-to-AHB bridge requests and gets the system bus.
4. The memory controller settings are read from the serial EEPROM and used to initialize the memory controller.
5. The BBus-to-AHB bridge loads the boot program into the SDRAM, starting at address 0.
6. The reset signal going to the CPU is released once the boot program is loaded. RESET_DONE is now set to 1.
7. The CPU begins to execute code from address 0x0000 0000.

Figure 3 shows a sample reset circuit.

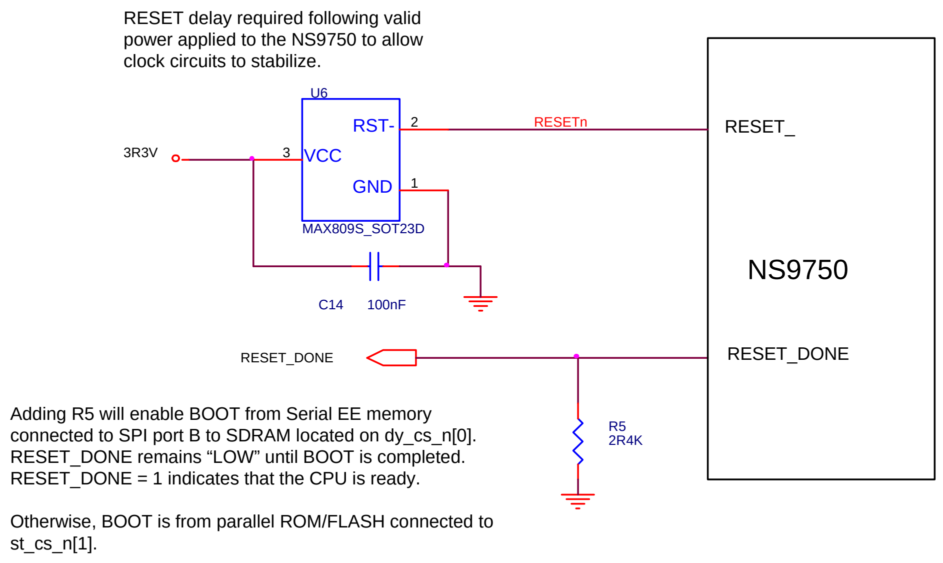

RESET delay required following valid power applied to the NS9750 to allow clock circuits to stabilize.
U6
RST-
VCC
GND
2
3
1
MAX809S_SOT23D
3R3V
RESETn
RESET_
NS9750
RESET_DONE
C14
100nF
RESET_DONE
R5
2R4K
Adding R5 will enable BOOT from Serial EE memory connected to SPI port B to SDRAM located on dy_cs_n[0]. RESET_DONE remains "LOW" until BOOT is completed. RESET_DONE = 1 indicates that the CPU is ready.
Otherwise, BOOT is from parallel ROM/FLASH connected to st_cs_n[1].

Figure 3: Sample reset circuit

You can use one of five software resets to reset the NS9750. Select the reset by setting the appropriate bit in the appropriate register.

- Watchdog timer can issue reset upon watchdog timer expiration (see "Software Watchdog Timer register" on page 293).
- AHB bus arbiter can issue reset upon AHB bus arbiter timer expiration.
- AHB bus monitor can issue reset upon AHB bus monitor timer expiration.
- Software reset can reset individual internal modules or all modules except memory and CPU (see "Reset and Sleep Control register" on page 295).
- The system is reset whenever software sets the PLL SW change bit to 1 (see "PLL Configuration register" on page 299).

Hardware reset duration is 4 ms for PLL to stabilize. Software duration depends on speed grade, as shown in Table 1.

Speed grade	CPU clock cycles	Duration
200 MHz	128	640 ns
162 MHz	128	790 ns
125 MHz	128	1024 ns

Table 1: Software reset duration

The minimum reset pulse width is 10 crystal clocks.

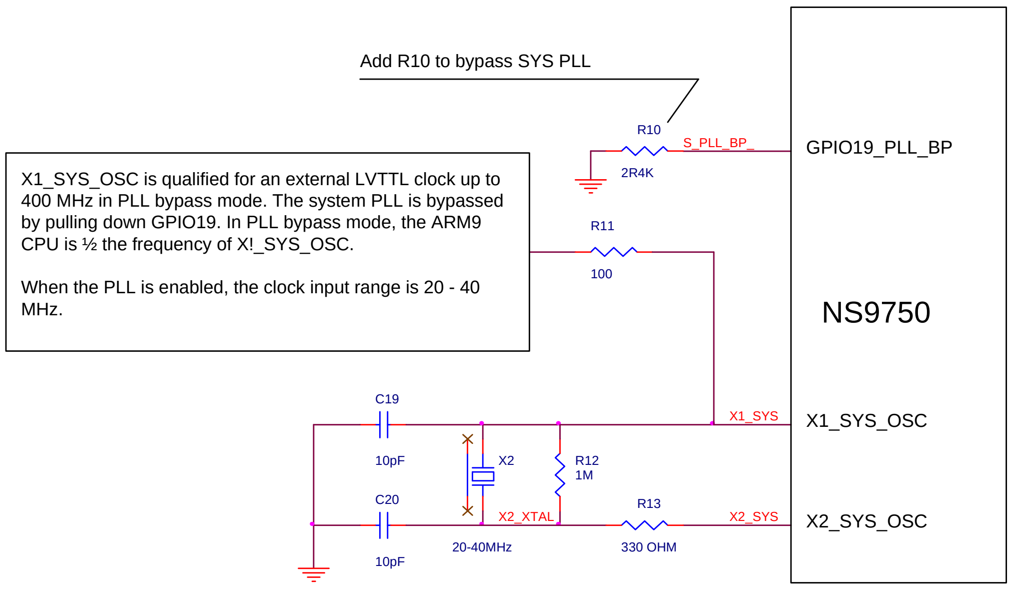
System clock

The system clock is provided to the NS9750 by either a crystal or an external oscillator. Table 2 shows sample clock frequency settings for each chip speed grade.

Speed	cpu_clk	hclk (main bus)	bbus_clk
200 MHz	200 (199.0656)	99.5328	49.7664
162 MHz	162.2016	81.1008	40.5504
125 MHz	125.3376	62.6688	31.3344

Table 2: Sample clock frequency settings with 29.4912 MHz crystal

If an oscillator is used, it must be connected to the x1_sys_osc input (C8 pin) on the NS9750. If a crystal is used, it must be connected with a circuit such as the one shown in Figure 4.



Add R10 to bypass SYS PLL
X1_SYS_OSC is qualified for an external LVTTL clock up to 400 MHz in PLL bypass mode. The system PLL is bypassed by pulling down GPIO19. In PLL bypass mode, the ARM9 CPU is ½ the frequency of X!_SYS_OSC.
When the PLL is enabled, the clock input range is 20 - 40 MHz.
R10
2R4K
S_PLL_BP_
GPIO19_PLL_BP
R11
100
NS9750
C19
10pF
X1_SYS
X1_SYS_OSC
X2
R12
1M
C20
10pF
X2_XTAL
R13
330 OHM
X2_SYS
X2_SYS_OSC
20-40MHz

Figure 4: System clock

The PLL parameters are initialized on powerup reset, and can be changed by software from f_{max} to 1/2 f_{max}. For a 200 MHz grade, then, the CPU may change from 200 MHZ to 100 MHz, the AHB system bus may change from 100 MHz to 50 MHz, and the peripheral BBus may change from 50 MHz to 25 MHz. If changed by software, the system resets automatically after the PLL stabilizes (approximately 4 ms).

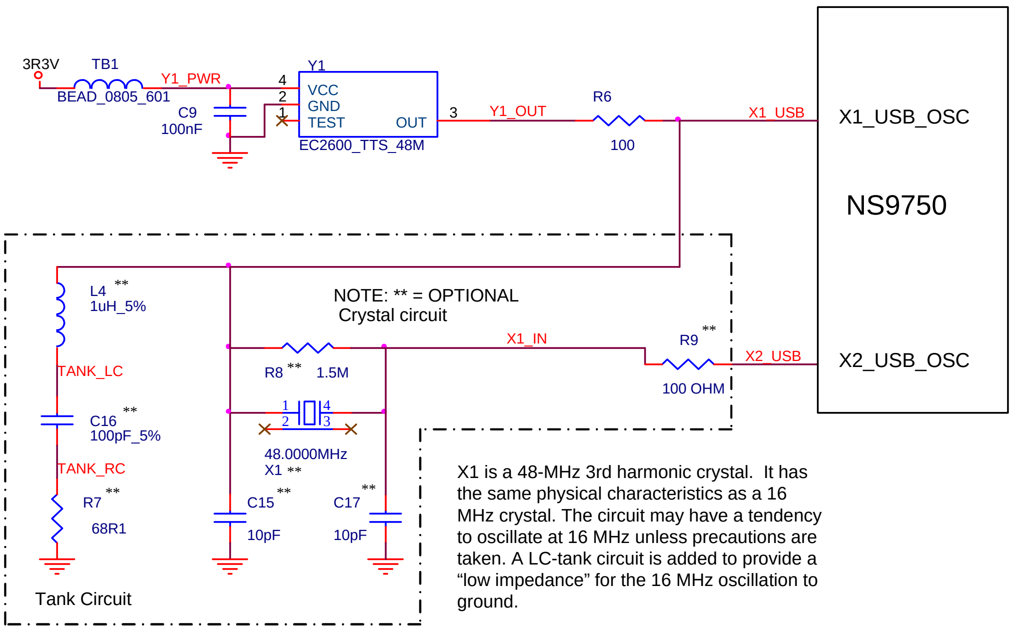
The system clock provides clocks for CPU, AHB system bus, peripheral BBus, PCI/ CardBus, LCD, timers, memory controller, and BBus modules (serial modules and 1284 parallel port).

The Ethernet MAC uses external clocks from a MII PHY or a RMII PHY. For a MII PHY, these clocks are input signals: rx_clk on pin T3 for receive clock and tx_clk on pin V3 for transmit clock. For a RMII, there is only one clock, and it connects to the rx_clk on pin T3. In this case, the transmit clock, tx_clk, should be tied low.

PCI/CardBus, LCD controller, serial modules (UART, SPI), and 1284 port can optionally use external clock signals.

USB clock

USB is clocked by a separate PLL driven by an external 48 MHz crystal, or it can be driven directly by an external 48 MHz oscillator.



X1 is a 48-MHz 3rd harmonic crystal. It has the same physical characteristics as a 16 MHz crystal. The circuit may have a tendency to oscillate at 16 MHz unless precautions are taken. A LC-tank circuit is added to provide a “low impedance” for the 16 MHz oscillation to ground.

Figure 5: USB clock

NS9750 Pinout

CHAPTER 2

The NS9750 offers a connection to an external bus expansion module, as well as a glueless connection to SDRAM, PC100 DIMM, flash, EEPROM, and SRAM memories, and an external bus expansion module. It includes a versatile embedded LCD controller, a PCI/CardBus port, a USB port, and four multi-function serial ports. The NS9750 provides up to 50 general purpose I/O (GPIO) pins and configurable power management with sleep mode.

Pinout and signal descriptions

Each pinout table applies to a specific interface, and contains the following information:

Heading	Description
Pin #	Pin number assignment for a specific I/O signal
Signal	Pin name for each I/O signal. Some signals have multiple function modes and are identified accordingly. The mode is configured through firmware using one or more configuration registers. **_n** in the signal name indicates that this signal is active *low*.
U/D	U or D indicates whether the pin is a pullup resistor or a pulldown resistor: ■ **U** — Pullup (input current source) ■ **D** — Pulldown (input current sink) If no value appears, that pin is neither a pullup nor pulldown resistor.
I/O	The type of signal: input, output, or input/output.
OD (mA)	The output drive of an output buffer. NS9750 uses one of three drivers: ■ 2 mA ■ 4 mA ■ 8 mA

More detailed signal descriptions are provided for selected modules.

System Memory interface

Pin #	Signal Name	U/D	OD (mA)	I/O	Description
A21	addr[0]		8	O	Address bus signal
B20	addr[1]		8	O	Address bus signal
C19	addr[2]		8	O	Address bus signal
A20	addr[3]		8	O	Address bus signal
B19	addr[4]		8	O	Address bus signal

Table 3: System Memory interface pinout

Pin #	Signal Name	U/D	OD (mA)	I/O	Description
C18	addr[5]		8	O	Address bus signal
A19	addr[6]		8	O	Address bus signal
A17	addr[7]		8	O	Address bus signal
C16	addr[8]		8	O	Address bus signal
B16	addr[9]		8	O	Address bus signal
A16	addr[10]		8	O	Address bus signal
D15	addr[11]		8	O	Address bus signal
C15	addr[12]		8	O	Address bus signal
B15	addr[13]		8	O	Address bus signal
A15	addr[14]		8	O	Address bus signal
C14	addr[15]		8	O	Address bus signal
B14	addr[16]		8	O	Address bus signal
A14	addr[17]		8	O	Address bus signal
A13	addr[18]		8	O	Address bus signal
B13	addr[19]		8	O	Address bus signal
C13	addr[20]		8	O	Address bus signal
A12	addr[21]		8	O	Address bus signal
B12	addr[22]		8	O	Address bus signal
C12	addr[23]		8	O	Address bus signal
D12	addr[24]		8	O	Address bus signal
A11	addr[25]		8	O	Address bus signal
B11	addr[26]		8	O	Address bus signal
C11	addr[27]		8	O	Address bus signal
G2	clk_en[0]		8	O	SDRAM clock enable
H3	clk_en[1]		8	O	SDRAM clock enable
G1	clk_en[2]		8	O	SDRAM clock enable

Table 3: System Memory interface pinout

Pin #	Signal Name	U/D	OD (mA)	I/O	Description
H2	clk_en[3]		8	O	SDRAM clock enable
A10	clk_out[0]		8	O	SDRAM reference clock. Connect to clk_in[0] using series termination.
A9	clk_out[1]		8	O	SDRAM clock
A5	clk_out[2]		8	O	SDRAM clock
A4	clk_out[3]		8	O	SDRAM clock
G26	data[0]		8	I/O	Data bus signal
H24	data[1]		8	I/O	Data bus signal
G25	data[2]		8	I/O	Data bus signal
F26	data[3]		8	I/O	Data bus signal
G24	data[4]		8	I/O	Data bus signal
F25	data[5]		8	I/O	Data bus signal
E26	data[6]		8	I/O	Data bus signal
F24	data[7]		8	I/O	Data bus signal
E25	data[8]		8	I/O	Data bus signal
D26	data[9]		8	I/O	Data bus signal
F23	data[10]		8	I/O	Data bus signal
E24	data[11]		8	I/O	Data bus signal
D25	data[12]		8	I/O	Data bus signal
C26	data[13]		8	I/O	Data bus signal
E23	data[14]		8	I/O	Data bus signal
D24	data[15]		8	I/O	Data bus signal
C25	data[16]		8	I/O	Data bus signal
B26	data[17]		8	I/O	Data bus signal
D22	data[18]		8	I/O	Data bus signal
C23	data[19]		8	I/O	Data bus signal
B24	data[20]		8	I/O	Data bus signal

Table 3: System Memory interface pinout

Pin #	Signal Name	U/D	OD (mA)	I/O	Description
A25	data[21]		8	I/O	Data bus signal
C22	data[22]		8	I/O	Data bus signal
D21	data[23]		8	I/O	Data bus signal
B23	data[24]		8	I/O	Data bus signal
A24	data[25]		8	I/O	Data bus signal
A23	data[26]		8	I/O	Data bus signal
B22	data[27]		8	I/O	Data bus signal
C21	data[28]		8	I/O	Data bus signal
A22	data[29]		8	I/O	Data bus signal
B21	data[30]		8	I/O	Data bus signal
C20	data[31]		8	I/O	Data bus signal
E1	data_mask[0]		8	O	SDRAM data mask signal
F2	data_mask[1]		8	O	SDRAM data mask signal
G3	data_mask[2]		8	O	SDRAM data mask signal
F1	data_mask[3]		8	O	SDRAM data mask signal
C5	clk_in[0]			I	SDRAM feedback clock. Connect to clk_out[0].
D2	clk_in[1]			I	Connect to GND
E3	clk_in[2]			I	Connect to GND
E2	clk_in[3]			I	Connect to GND
B4	byte_lane_sel_n[0]		8	O	Static memory byte_lane_enable[0] or write_enable_n[0] for byte-wide device signals
F4	byte_lane_sel_n[1]		8	O	Static memory byte_lane_enable[1] or write_enable_n[1] for byte-wide device signals
D1	byte_lane_sel_n[2]		8	O	Static memory byte_lane_enable[2] or write_enable_n[2] for byte-wide device signals
F3	byte_lane_sel_n[3]		8	O	Static memory byte_lane_enable[3] or write_enable_n[3] for byte-wide device signals
B5	cas_n		8	O	SDRAM column address strobe

Table 3: System Memory interface pinout

Pin #	Signal Name	U/D	OD (mA)	I/O	Description
A8	dy_cs_n[0]		8	O	SDRAM chip select signal
B8	dy_cs_n[1]		8	O	SDRAM chip select signal
A6	dy_cs_n[2]		8	O	SDRAM chip select signal
C7	dy_cs_n[3]		8	O	SDRAM chip select signal
C6	st_oe_n		8	O	Static memory output enable
D6	ras_n		8	O	SDRAM row address strobe
H1	dy_pwr_n		8	O	SyncFlash power down
B10	st_cs_n[0]		8	O	Static memory chip select signal
C10	st_cs_n[1]		8	O	Static memory chip select signal
B9	st_cs_n[2]		8	O	Static memory chip select signal
C9	st_cs_n[3]		8	O	Static memory chip select signal
B6	we_n		8	O	SDRAM write enable. Used for static and SDRAM devices.
J3	ta_strb	U		I	Slow peripheral transfer acknowledge

Table 3: System Memory interface pinout

System Memory interface signals

Table 4 describes System Memory interface signals in more detail. All signals are internal to the chip.

Name	I/O	Description
addr[27:0]	O	Address output. Used for both static and SDRAM devices. SDRAM memories use bits [14:0]; static memories use bits [25:0].

Table 4: System Memory interface signal descriptions

Name	I/O	Description
clk_en[3:0]	O	SDRAM clock enable. Used for SDRAM devices. **Note:** The clk_en signals are associated with the dy_cs_n signals. Connect SDRAM clock enables directly to a 3.3V or pullup resistor to avoid an SDRAM lockup condition during a manual or brownout condition reset. As an alternative, you can use an analog switch to connect the clock enables to the SDRAM devices to a pullup resistor until the NS9750 device reset is complete, as indicated by a high level on the reset_done output. See the sample circuit shown in Figure 7, "NS9750 clock enable configuration," on page 25.
clk_out[3:1]	O	SDRAM clocks. Used for SDRAM devices.
clk_out[0]	O	SDRAM clk_out[0] is connected to clk_in[0].
data[31:0]	I/O	Read data from memory. Used for the static memory controller and the dynamic memory controller.
data_mask[3:0]	O	Data mask output to SDRAMs. Used for SDRAM devices.
clk_in[3:1]	I	Feedback clocks. Used for SDRAM devices.
clk_in[0]	I	Feedback clock [0]. Always connects to clk_out[0].
byte_lane_sel_n[3:0]	O	Static memory byte_lane_select, active low, or write_enable_n for byte-wide devices.
cas_n	O	Column address strobe. Used for SDRAM devices.
dy_cs_n[3:0]	O	SDRAM chip selects. Used for SDRAM devices.
st_oe_n	O	Output enable for static memories. Used for static memory devices.
ras_n	O	Row address strobe. Used for SDRAM devices.
st_cs_n[3:0]	O	Static memory chip selects. Default active low. Used for static memory devices.
we_n	O	Write enable. Used for SDRAM and static memories.
ta_strb	I	*Slow peripheral transfer acknowledge* can be used to terminate static memory cycles sooner than the number of wait states programmed in the chip select setup register.

Table 4: System Memory interface signal descriptions

Figure 6 shows NS9750 SDRAM clock termination.

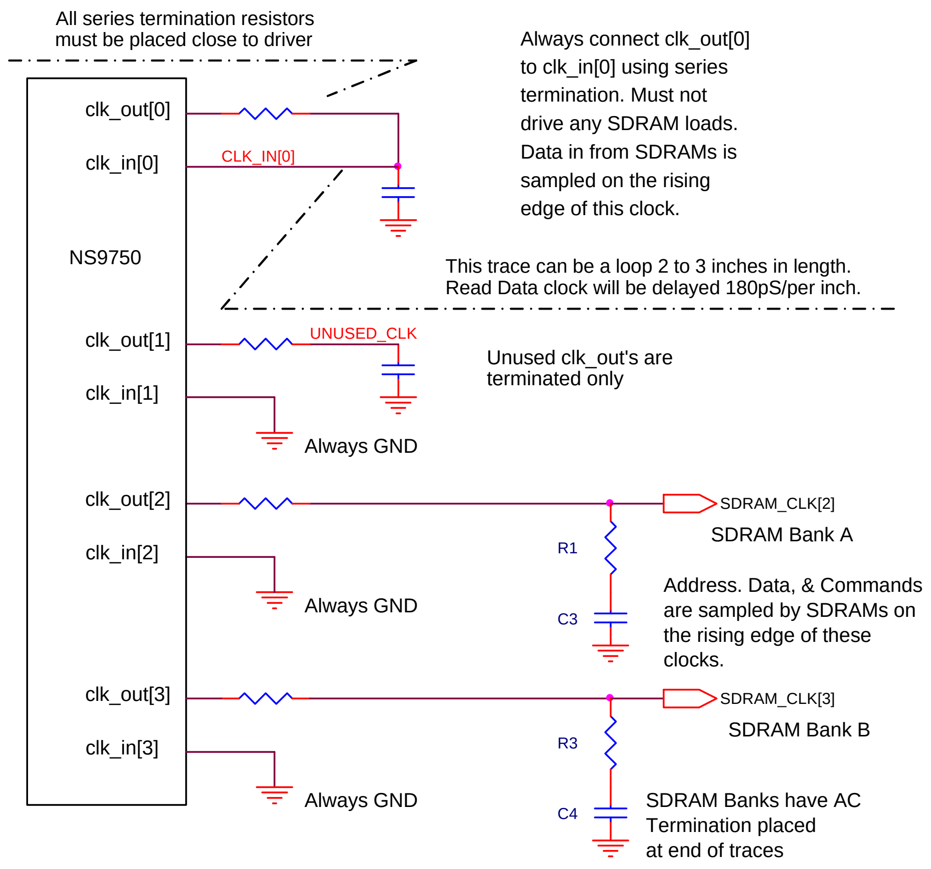


Figure 6: SDRAM clock termination

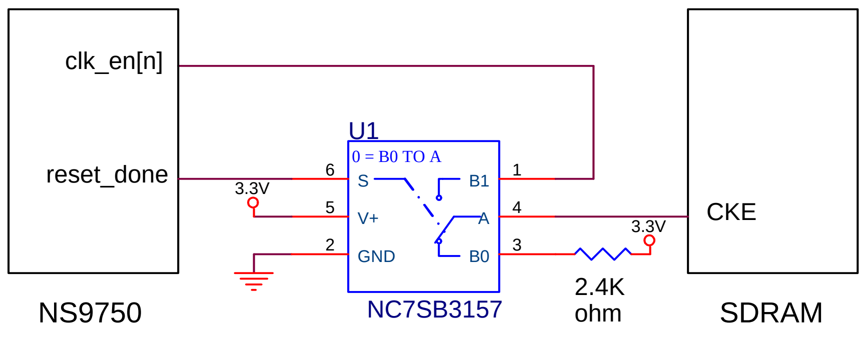


Figure 7: NS9750 clock enable configuration

Ethernet interface

Pin #	Signal name MII	Signal name RMII	U/D	OD (mA)	I/O	Description MII	Description RMII
AB1	col	N/C			I	Collision	Pull low external to NS9750
AA2	crs	crs_dv			I	Carrier sense	Carrier sense
AC1	enet_phy_int_n	enet_phy_int_n	U		I	Ethernet PHY interrupt	Ethernet PHY interrupt
AA3	mdc	mdc		4	O	MII management interface clock	MII management interface clock
AB2	mdio	mdio	U	2	I/O	MII management data	MII management data
T3	rx_clk	ref_clk			I	Receive clock	Reference clock
V2	rx_dv	N/C			I	Receive data valid	Pull low external to NS9750
W1	rx_er	rx_er			I	Receive error	Optional signal; pull low to NS9750 if not used
V1	rxd[0]	rxd[0]			I	Receive data bit 0	Receive data bit 0

Table 5: Ethernet interface pinout

Pin #	Signal name		U/D	OD (mA)	I/O	Description	
	MII	RMII				MII	RMII
U3	rxd[1]	rxd[1]			I	Receive data bit 1	Receive data bit 1
U2	rxd[2]	N/C			I	Receive data bit 2	Pull low external to NS9750
U1	rxd[3]	N/C			I	Receive data bit 3	Pull low external to NS9750
V3	tx_clk	N/C			I	Transmit clock	Pull low external to NS9750
AA1	tx_en	tx_en		2	O	Transmit enable	Transmit enable
Y3	tx_er	N/C		2	O	Transmit error	N/A
Y2	txd[0]	txd[0]		2	O	Transmit data bit 0	Transmit data bit 0
W3	txd[1]	txd[1]		2	O	Transmit data bit 1	Transmit data bit 1
Y1	txd[2]	N/C		2	O	Transmit data bit 2	N/A
W2	txd[3]	N/C		2	O	Transmit data bit 3	N/A

Table 5: Ethernet interface pinout

Clock generation/system pins

Pin #	Signal name	U/D	OD (mA)	I/O	Description
C8	x1_sys_osc			I	System clock crystal oscillator circuit input
B7	x2_sys_osc			O	System clock crystal oscillator circuit output
D9	x1_usb_osc			I	USB clock crystal oscillator circuit input. (Connect to GND if USB is not used.)
A7	x2_usb_osc			O	USB clock crystal oscillator circuit output
AC21	reset_done	U	2	I/O	CPU is enabled once the boot program is loaded. Reset_done is set to 1.
H25	reset_n	U		I	System reset input signal

Table 6: Clock generation and system pin pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description
AD20	bist_en_n			I	Enable internal BIST operation
AF21	pll_test_n			I	Enable PLL testing
AE21	scan_en_n			I	Enable internal scan testing
B18	sys_pll_dvdd				System clock PLL 1.5V digital power
A18	sys_pll_dvss				System clock PLL digital ground
B17	sys_pll_avdd				System clock PLL 3.3V analog power
C17	sys_pll_avss				System clock PLL analog ground
J2	lcdclk	U		I	External LCD clock input
T2	boot_strap[0]	U	2	I/O	Chip select 1 static memory byte_lane_enable_n, or write_enable_n for byte-wide devices bootstrap select
N3	boot_strap[1]	U	2	I/O	CardBus mode bootstrap select
P1	boot_strap[2]	U	2	I/O	Memory interface read mode bootstrap select
P2	boot_strap[3]	U	2	I/O	Chip select 1 data width bootstrap select
P3	boot_strap[4]	U	2	I/O	Chip select 1 data width bootstrap select

Table 6: Clock generation and system pin pinout

bist_en_n, pll_test_n, and scan_en_n

Table 7 is a truth/termination table for bist_en_n, pll_test_n, and scan_en_n.

	Normal operation	ARM debug	
pll_test_n	pull up	pull up	10K recommended
bist_en_n	pull down	pull up	10K pullup = debug 2.4K pulldown = normal
scan_en_n	pull down	pull down	2.4K recommended

Table 7: bist_en_n, pll_test_n, & scan_en_n truth/termination table

PCI interface

The PCI interface can be set to PCI host or PCI device (slave) using the pci_central_rsc_n pin.

Note: All output drivers for PCI meet the standard PCI driver specification.

Pin #	Signal name	U/D	OD (mA)	I/O	Description
J24	ad[0][1]		N/A	I/O	PCI time-multiplexed address/data bus
H26	ad[1][1]		N/A	I/O	PCI time-multiplexed address/data bus
J25	ad[2][1]		N/A	I/O	PCI time-multiplexed address/data bus
J26	ad[3][1]		N/A	I/O	PCI time-multiplexed address/data bus
K24	ad[4][1]		N/A	I/O	PCI time-multiplexed address/data bus
K25	ad[5][1]		N/A	I/O	PCI time-multiplexed address/data bus
K26	ad[6][1]		N/A	I/O	PCI time-multiplexed address/data bus
L24	ad[7][1]		N/A	I/O	PCI time-multiplexed address/data bus
L26	ad[8][1]		N/A	I/O	PCI time-multiplexed address/data bus
M24	ad[9][1]		N/A	I/O	PCI time-multiplexed address/data bus
M25	ad[10][1]		N/A	I/O	PCI time-multiplexed address/data bus
M26	ad[11][1]		N/A	I/O	PCI time-multiplexed address/data bus

Table 8: PCI interface pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description
N24	ad[12][1]		N/A	I/O	PCI time-multiplexed address/data bus
N25	ad[13][1]		N/A	I/O	PCI time-multiplexed address/data bus
N26	ad[14][1]		N/A	I/O	PCI time-multiplexed address/data bus
P26	ad[15][1]		N/A	I/O	PCI time-multiplexed address/data bus
U24	ad[16][1]		N/A	I/O	PCI time-multiplexed address/data bus
V26	ad[17][1]		N/A	I/O	PCI time-multiplexed address/data bus
V25	ad[18][1]		N/A	I/O	PCI time-multiplexed address/data bus
W26	ad[19][1]		N/A	I/O	PCI time-multiplexed address/data bus
V24	ad[20][1]		N/A	I/O	PCI time-multiplexed address/data bus
W25	ad[21][1]		N/A	I/O	PCI time-multiplexed address/data bus
Y26	ad[22][1]		N/A	I/O	PCI time-multiplexed address/data bus
W24	ad[23][1]		N/A	I/O	PCI time-multiplexed address/data bus
Y24	ad[24][1]		N/A	I/O	PCI time-multiplexed address/data bus
AA25	ad[25][1]		N/A	I/O	PCI time-multiplexed address/data bus
AB26	ad[26][1]		N/A	I/O	PCI time-multiplexed address/data bus
AA24	ad[27][1]		N/A	I/O	PCI time-multiplexed address/data bus
AB25	ad[28][1]		N/A	I/O	PCI time-multiplexed address/data bus
AC26	ad[29][1]		N/A	I/O	PCI time-multiplexed address/data bus
AD26	ad[30][1]		N/A	I/O	PCI time-multiplexed address/data bus
AC25	ad[31][1]		N/A	I/O	PCI time-multiplexed address/data bus
L25	cbe_n[0][1]		N/A	I/O	Command/byte enable
P25	cbe_n[1][1]		N/A	I/O	Command/byte enable
U25	cbe_n[2][1]		N/A	I/O	Command/byte enable
AA26	cbe_n[3][1]		N/A	I/O	Command/byte enable
T26	devsel_n[2]		N/A	I/O	Device select
U26	frame_n[2]		N/A	I/O	Cycle frame

Table 8: PCI interface pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description
Y25	idsel[3, 4]		N/A	I	Initialization device select: ■ For PCI host applications, connect to AD11. ■ For PCI device applications, connection is determined by the PCI device number assigned to the NS9750. ■ For CardBus applications, connect to the external pullup resistor. ■ Do not allow input to float in any application.
T24	irdy_n[2]		N/A	I/O	Initiator ready
P24	par[1]		N/A	I/O	Parity signal
R25	perr_n[2]		N/A	I/O	Parity error
R26	serr_n[2]		N/A	I/O	System error Input: pci_central_resource_n = 0 Output: pci_central_resource_n = 1
R24	stop_n[2]		N/A	I/O	Stop signal
T25	trdy_n[2]		N/A	I/O	Target ready
AC24	pci_arb_gnt_1_n[6]		N/A	O	PCI channel 1 grant
AD23	pci_arb_gnt_2_n[6]		N/A	O	PCI channel 2 grant
AE24	pci_arb_gnt_3_n[6]		N/A	O	PCI channel 3 grant
AD25	pci_arb_req_1_n[2]		N/A	I	PCI channel 1 request
AB23	pci_arb_req_2_n[2]		N/A	I	PCI channel 2 request
AC22	pci_arb_req_3_n[2]		N/A	I	PCI channel 3 request
AF23	pci_central_resource_n	D	N/A	I	PCI internal central resource enable
AF25	pci_int_a_n[2]		N/A	I/O	PCI interrupt request A, output if external central resource used
AF24	pci_int_b_n[2]		N/A	I/O	PCI interrupt request B, CCLKRUN# for CardBus applications
AE23	pci_int_c_n[2]		N/A	I	PCI interrupt request C

Table 8: PCI interface pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description
AD22	pci_int_d_n[2]		N/A	I	PCI interrupt request D
AE26	pci_reset_n[3]		N/A	I/O	PCI reset, output if internal central resource enabled
AB24	pci_clk_in	U	N/A	I	PCI clock in. (Connected to pci_clk_out or an externally generated PCI reference clock.)
AA23	pci_clk_out		N/A	O	PCI clock out

Table 8: PCI interface pinout

PCI/CardBus signals

Most of the CardBus signals are the same as the PCI signals. Other CardBus signals are unique and multiplexed with PCI signals for the NS9750. Table 9 shows these unique signals. Figure 8 illustrates how to terminate an unused PCI.

PCI signal	CardBus signal	CardBus type	Description
INTA#	CINT#[4]	Input	CardBus interrupt pin. The INTA2PCI pin in the PCI Miscellaneous Support register must be set to 0.
INTB#	CCLKRUN#[4]	Bidir	CardBus pin used to negotiate with the external CardBus device before stopping the clock. Allows external CardBus device to request that the clock be restarted.
INTC#	CSTSCHG[5]	Input	CardBus status change interrupt signal.
GNT1#	CGNT#[4]	Output	Grant to external CardBus device from NS9750's internal arbiter.

Table 9: CardBus IO multiplexed signals

PCI signal	CardBus signal	CardBus type	Description
GNT2#	CVS1	Output	Voltage sense pin. Normally driven low by NS9750, but toggled during the interrogation of the external CardBus device to find voltage requirements. **Note:** Do not connect directly to the CardBus connector. See the diagram "CardBus system connections to NS9750" on page 462 for a suggested connection scheme.
GNT3#	CVS2	Output	Voltage sense pin. Normally driven low by NS9750, but toggled during the interrogation of the external CardBus device to find voltage requirements.
REQ1#	CREQ#[4]	Input	Request from external CardBus device to NS9750's internal arbiter.
REQ2#	CCD1[4]	Input	Card detect pin. Pulled up when the socket is empty and pulled low when the external CardBus device is in the socket.
REQ3#	CCD2[4]	Input	Card detect pin. Pulled up when the socket is empty and pulled low when the external CardBus device is in the socket.

Table 9: CardBus IO multiplexed signals

Notes:

1 Add external pulldown resistor *only* if the PCI interface is not being used. See the discussion of PCI bridge configuration in *Sample Driver Configurations* for information about eliminating the pulldown resistor.

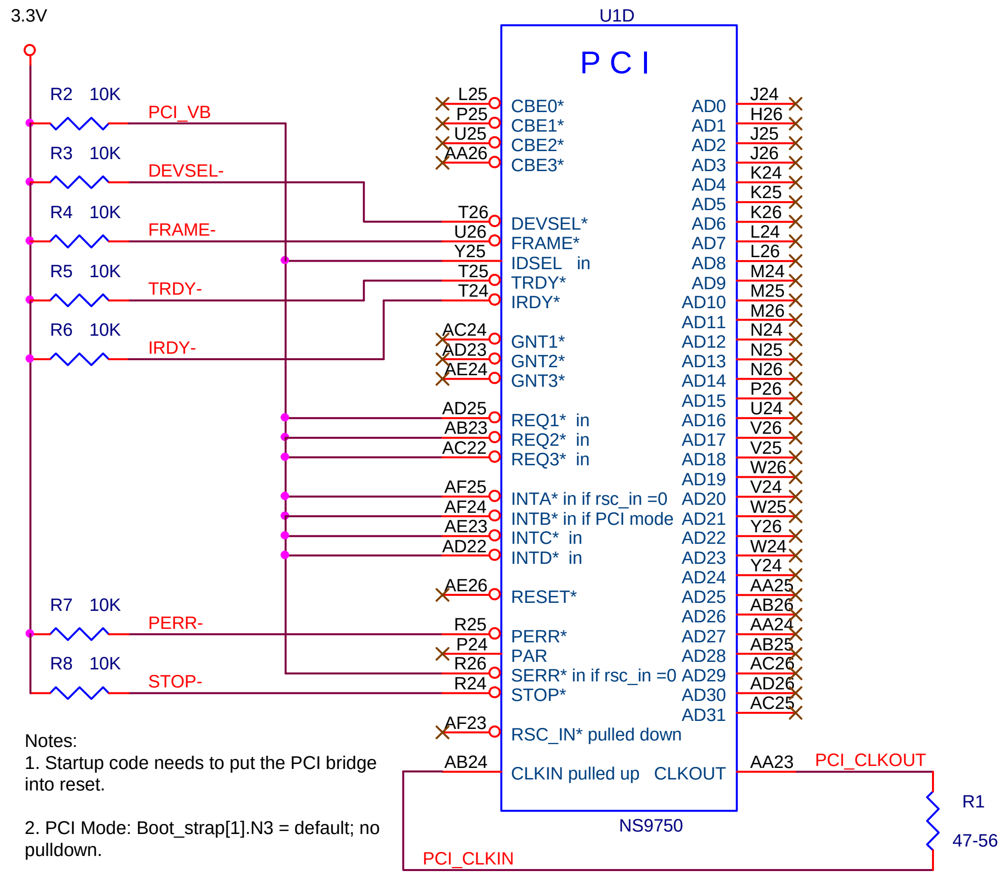
2 Add external pullup resistors *regardless* of whether the PCI interface is being used.

3 Add external pullup resistor *only* if the PCI interface is not being used.

4 Add external pullup resistor in CardBus mode.

5 Add external pulldown resistor in CardBus mode.

6 Add external pullup only if the PCI interface is being used and this signal is also being used.



Notes:
1. Startup code needs to put the PCI bridge into reset.

2. PCI Mode: Boot_strap[1].N3 = default; no pulldown.

3. NS9750 is current PCI bus master. Signals that it can drive should have individual pullups.

Figure 8: NS9750 unused PCI termination

GPIO MUX

- The BBus utility contains the control pins for each GPIO MUX bit. Each pin can be selected individually; that is, you can select any option (00, 01, 02, 03) for any pin, by setting the appropriate bit in the appropriate register.
- Some signals are muxed to two different GPIO pins, to maximize the number of possible applications. These duplicate signals are marked as such in the Descriptions column in the table. Selecting the primary GPIO pin and the duplicate GPIO pin for the same function is not recommended. If both the primary GPIO pin and the duplicate GPIO pin are programmed for the same function, however, the primary GPIO pin has precedence and will be used.
- The 00 option for the serial ports (B, A, C, and D) is configured for UART and SPI mode, respectively; that is, the UART option is shown first, followed by the SPI option if there is one. If only one value appears, it is the UART value. SPI options all begin with *SPI*.

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
AF19	gpio[0][1]	U	2	I/O	00	Ser port B TxData / SPI port B dout
					01	DMA ch 1 done (duplicate)
					02	Timer 1 (duplicate)
					03	GPIO 0
AE18	gpio[1]	U	2	I/O	00	Ser port B RxData / SPI port B din
					01	DMA ch 1 req (duplicate)
					02	Ext IRQ 0
					03	GPIO 1
AF18	gpio[2][1]	U	2	I/O	00	Ser port B RTS
					01	Timer 0
					02	DMA ch 2 read enable
					03	GPIO 2
AD17	gpio[3]	U	2	I/O	00	Ser port B CTS
					01	1284 nACK (peripheral-driven)
					02	DMA ch 1 req
					03	GPIO 3

Table 10: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
AE17	gpio[4][1]	U	2	I/O	00	Ser port B DTR
					01	1284 busy (peripheral-driven)
					02	DMA ch 1 done
					03	GPIO 4
AF17	gpio[5]	U	2	I/O	00	Ser port B DSR
					01	1284 PError (peripheral-driven)
					02	DMA ch 1 read enable
					03	GPIO 5
AD16	gpio[6]	U	2	I/O	00	Ser port B RI / SPI port B clk
					01	1284 nFault (peripheral-driven)[3]
					02	Timer 7 (duplicate)
					03	GPIO 6
AE16	gpio[7]	U	2	I/O	00	Ser port B DCD / SPI port B enable
					01	DMA ch 1 read enable (duplicate)
					02	Ext IRQ 1
					03	GPIO 7
AD15	gpio[8][1]	U	2	I/O	00	Ser port A TxData / SPI port A dout
					01	Reserved
					02	Reserved
					03	GPIO 8
AE15	gpio[9]	U	2	I/O	00	Ser port A RxData / SPI port A din
					01	Reserved
					02	Timer 8 (duplicate)
					03	GPIO 9
AF15	gpio[10][1]	U	2	I/O	00	Ser port A RTS
					01	Reserved
					02	Reserved
					03	GPIO 10
AD14	gpio[11]	U	2	I/O	00	Ser port A CTS
					01	Ext IRQ2 (duplicate)
					02	Timer 0 (duplicate)
					03	GPIO 11

Table 10: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
AE14	gpio[12][1]	U	2	I/O	00	Ser port A DTR
					01	Reserved
					02	Reserved
					03	GPIO 12
AF14	gpio[13]	U	2	I/O	00	Ser port A DSR
					01	Ext IRQ 0 (duplicate)
					02	Timer 10 (duplicate)
					03	GPIO 13
AF13	gpio[14]	U	2	I/O	00	Ser port A RI / SPI port A clk
					01	Timer 1
					02	Reserved
					03	GPIO 14
AE13	gpio[15]	U	2	I/O	00	Ser port A DCD / SPI port A enable
					01	Timer 2
					02	Reserved
					03	GPIO 15
AD13	gpio[16][2]	U	2	I/O	00	Reserved
					01	1284 nFault (peripheral-driven, duplicate)[3]
					02	Timer 11 (duplicate)
					03	GPIO 16
AF12	gpio[17][1,2]	U	2	I/O	00	USB power relay
					01	Reserved
					02	Reserved
					03	GPIO 17
AE12	gpio[18]	U	4	I/O	00	Ethernet CAM reject
					01	LCD power enable
					02	Ext IRQ 3 (duplicate)
					03	GPIO 18
AD12	gpio[19][1]	U	4	I/O	00	Ethernet CAM req
					01	LCD line-horz sync
					02	DMA ch 2 read enable (duplicate)
					03	GPIO 19

Table 10: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
AC12	gpio[20][1]	U	8	I/O	00	Ser port C DTR
					01	LCD clock
					02	Reserved
					03	GPIO 20
AF11	gpio[21]	U	4	I/O	00	Ser port C DSR
					01	LCD frame pulse-vert
					02	Reserved
					03	GPIO 21
AE11	gpio[22]	U	4	I/O	00	Ser port C RI / SPI port C clk
					01	LCD AC bias-data enable
					02	Reserved
					03	GPIO 22
AD11	gpio[23]	U	4	I/O	00	Ser port C DCD / SPI port C enable
					01	LCD line end
					02	Timer 14 (duplicate)
					03	GPIO 23
AF10	gpio[24][1]	U	4	I/O	00	Ser port D DTR
					01	LCD data bit 0
					02	Reserved
					03	GPIO 24
AE10	gpio[25]	U	4	I/O	00	Ser port D DSR
					01	LCD data bit 1
					02	Timer 15 (duplicate)
					03	GPIO 25
AD10	gpio[26]	U	4	I/O	00	Ser port D RI / SPI port D clk
					01	LCD data bit 2
					02	Timer 3
					03	GPI0 26
AF9	gpio[27]	U	4	I/O	00	Ser port D DCD / SPI port D enable
					01	LCD data bit 3
					02	Timer 4
					03	GPIO 27

Table 10: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
AE9	gpio[28]	U	4	I/O	00	Ext IRQ 1 (duplicate)
					01	LCD data bit 4
					02	LDC data bit 8 (duplicate)
					03	GPIO 28
AF8	gpio[29]	U	4	I/O	00	Timer 5
					01	LCD data bit 5
					02	LCD data bit 9 (duplicate)
					03	GPIO 29
AD9	gpio[30]	U	4	I/O	00	Timer 6
					01	LCD data bit 6
					02	LCD data bit 10 (duplicate)
					03	GPIO 30
AE8	gpio[31]	U	4	I/O	00	Timer 7
					01	LCD data bit 7
					02	LCD data bit 11 (duplicate)
					03	GPIO 31
AF7	gpio[32]	U	4	I/O	00	Ext IRQ 2
					01	1284 Data 1 (bidirectional)
					02	LCD data bit 8
					03	GPIO 32
AD8	gpio[33]	U	4	I/O	00	Timer 8
					01	1284 Data 2 (bidirectional)
					02	LCD data bit 9
					03	GPIO 33
AD7	gpio[34]	U	4	I/O	00	Timer 9
					01	1284 Data 3 (bidirectional)
					02	LCD data bit 10
					03	GPIO 34
AE6	gpio[35]	U	4	I/O	00	Timer 10
					01	1284 Data 4 (bidirectional)
					02	LCD data bit 11
					03	GPIO 35

Table 10: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
AF5	gpio[36]	U	4	I/O	00	Reserved
					01	1284 Data 5 (bidirectional)
					02	LCD data bit 12
					03	GPIO 36
AD6	gpio[37]	U	4	I/O	00	Reserved
					01	1284 Data 6 (bidirectional)
					02	LCD data bit 13
					03	GPIO 37
AE5	gpio[38]	U	4	I/O	00	Reserved
					01	1284 Data 7 (bidirectional)
					02	LCD data bit 14
					03	GPIO 38
AF4	gpio[39]	U	4	I/O	00	Reserved
					01	1284 Data 8 (bidirectional)
					02	LCD data bit 15
					03	GPIO 39
AC6	gpio[40]	U	4	I/O	00	Ser port C TxData / SPI port C dout
					01	Ext IRQ 3
					02	LCD data bit 16
					03	GPIO 40
AD5	gpio[41]	U	4	I/O	00	Ser port C RxData / SPI port C din
					01	Timer 11
					02	LCD data bit 17
					03	GPIO 41
AE4	gpio[42]	U	4	I/O	00	Ser port C RTS
					01	Timer 12
					02	LCD data bit 18
					03	GPIO 42
AF3	gpio[43]	U	4	I/O	00	Ser port C CTS
					01	Timer 13
					02	LCD data bit 19
					03	GPIO 43

Table 10: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
AD2	gpio[44][1]	U	4	I/O	00	Ser port D TxData / SPI port D dout
					01	1284 Select (peripheral-driven)
					02	LCD data bit 20
					03	GPIO 44
AE1	gpio[45]	U	4	I/O	00	Ser port D RxData / SPI port D din
					01	1284 nStrobe (host-driven)
					02	LCD data bit 21
					03	GPIO 45
AB3	gpio[46]	U	4	I/O	00	Ser port D RTS
					01	1284 nAutoFd (host-driven)
					02	LCD data bit 22
					03	GPIO 46
AA4	gpio[47]	U	4	I/O	00	Ser port D CTS
					01	1284 nInit (host-driven)
					02	LCD data bit 23
					03	GPIO 47
AC2	gpio[48]	U	2	I/O	00	Timer 14
					01	1284 nSelectIn (host-driven)
					02	DMA ch 2 req
					03	GPIO 48
AD1	gpio[49][1]	U	2	I/O	00	Timer 15
					01	1284 peripheral logic high (peripheral-driven)
					02	DMA ch 2 done
					03	GPIO 49

1 This pin is used for bootstrap initialization (see Table 168, "Configuration pins — Bootstrap initialization," on page 273). Note that the GPIO pins used as bootstrap pins have a defined powerup state that is required for the appropriate NS9750 configuration. If these GPIO pins are also used to control external devices (for example, power switch enable), the powerup state for the external device should be compatible with the boostrap state. If the powerup state is not compatible with the bootstrap state, either select a different GPIO pin to control the external device or add additional circuitry to reach the proper powerup state to the external device.

Table 10: GPIO MUX pinout

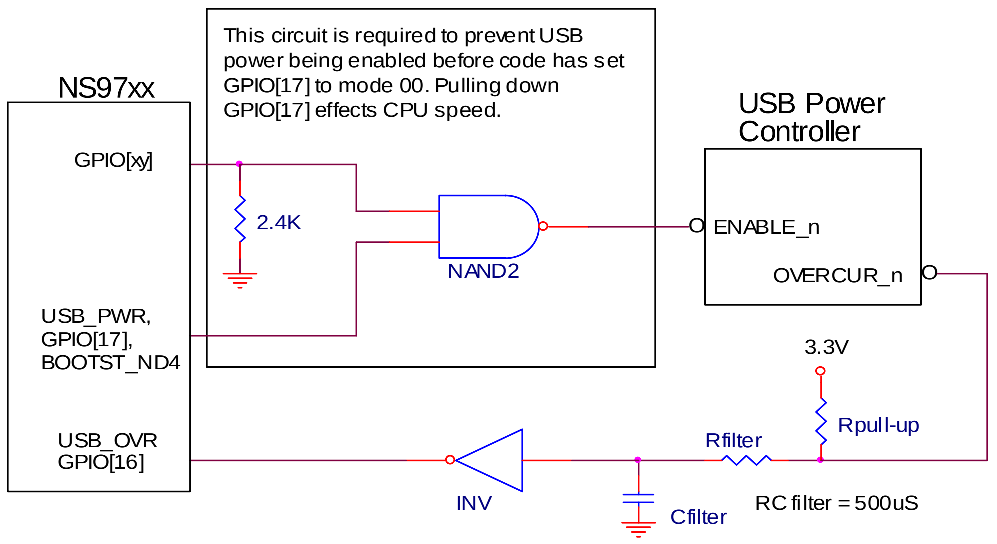
Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)

2 gpio[17] is used as both a bootstrap input pin for PLL_ND and an output that controls a power switch for USB Host power. If the power switch needs to powerup in the inactive state, the enable to the power switch must be the same value as the bootstrap value for PLL_ND; for example, if PLL_ND requires high on gpio[17], a high true power switch must be selected. gpio[16] is used for USB_OVR and should have a noise filter to prevent false indications of overcurrent, unless the USB power IC has this filter built in. See "Example: Implementing gpio[16] and gpio[17]" on page 41 for an illustration.

3 The nFault signal GPIO6 or GPIO16 can be used as a code-controlled direction pin for the transceiver. The polarity cannot be altered inside the NS9750; an inverter will be required.

Table 10: GPIO MUX pinout

Example: Implementing gpio[16] and gpio[17]


This circuit is required to prevent USB power being enabled before code has set GPIO[17] to mode 00. Pulling down GPIO[17] effects CPU speed.
NS97xx
GPIO[xy]
USB_PWR, GPIO[17], BOOTST_ND4
USB_OVR GPIO[16]
2.4K
NAND2
USB Power Controller
ENABLE_n
OVERCUR_n
3.3V
Rpull-up
Rfilter
INV
Cfilter
RC filter = 500uS

LCD module signals

The LCD module signals are multiplexed with GPIO pins. They include seven control signals and up to 24 data signals. Table 11 describes the control signals.

Signal name	Type	Description
CLPOWER	Output	LCD panel power enable
CLLP	Output	Line synchronization pulse (STN) / horizontal synchronization pulse (TFT)
CLCP	Output	LCD panel clock
CLFP	Output	Frame pulse (STN) / vertical synchronization pulse (TFT)
CLAC	Output	STN AC bias drive or TFT data enable output
CLD[23:0]	Output	LCD panel data
CLLE	Output	Line end signal

Table 11: LCD module signal descriptions

The CLD[23:0] signal has eight modes of operation:

- TFT 24-bit interface
- TFT 18-bit interface
- Color STN single panel
- Color STN dual panel
- 4-bit mono STN single panel
- 4-bit mono STN dual panel
- 8-bit mono STN single panel
- 8-bit mono STN dual panel

See the discussion of LCD panel signal multiplexing details for information about the CLD signals used with STN and TFT displays.

I^{2}C interface

Bits	Signal name	U/D	OD (mA)	I/O	Description
AC15	iic_scl		4	I/O	I^{2}C serial clock line. Add a 10K resistor to VDDA(3.3V) if not used.
AF16	iic_sda		4	I/O	I^{2}C serial data line. Add a 10K resistor to VDDA(3.3V) if not used.

Table 12: I^{2}C interface pinout

USB interface

Notes:

- If not using the USB interface, these pins should be pulled down to ground through a 15K ohm resistor.
- All output drivers for USB meet the standard USB driver specification.

Bits	Signal name	U/D	OD (mA)	I/O	Description
AB4	usb_dm			I/O	USB data -
AC3	usb_dp			I/O	USB data +

Table 13: USB interface pinout

JTAG interface for ARM core/boundary scan

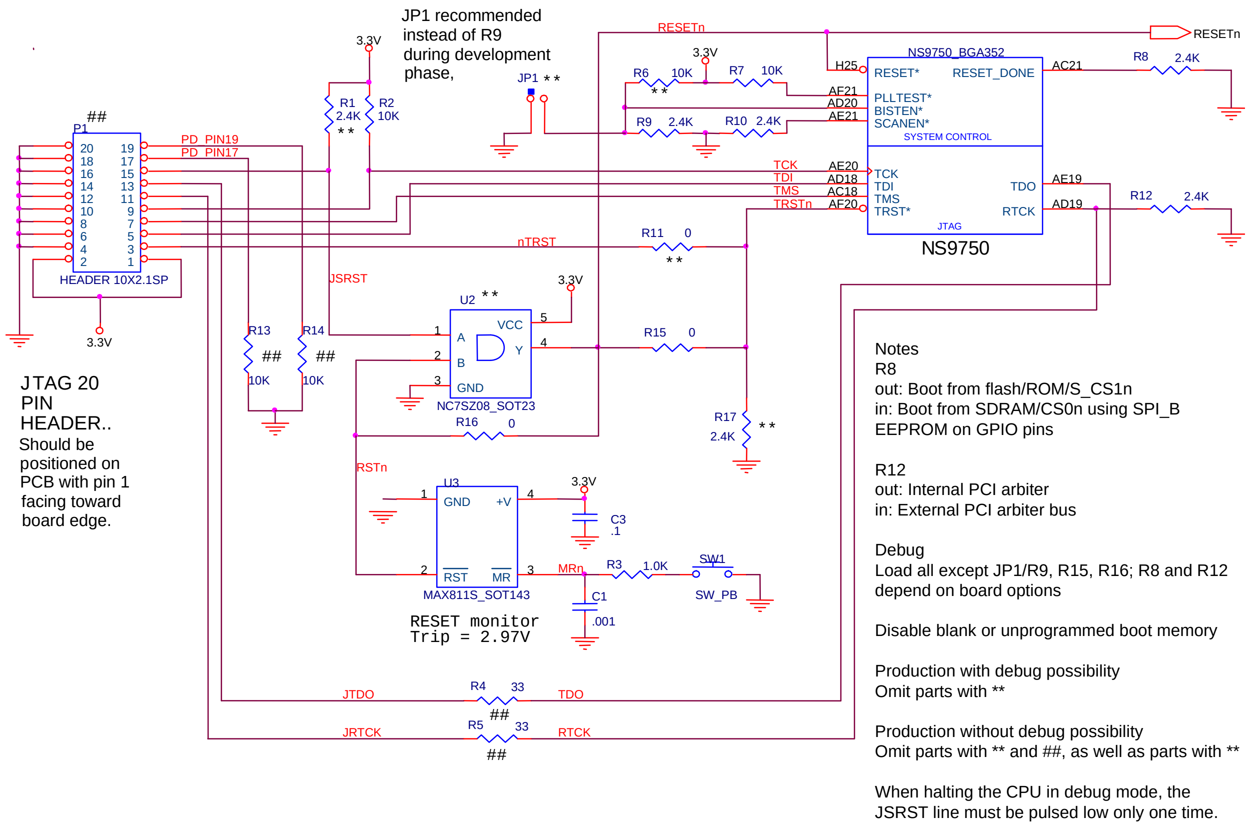
Note: trst_n must be pulsed low to initialize JTAG when a debugger is not attached. See Figure 9, "JTAG interface," on page 44.

Bits	Signal name	U/D	OD (mA)	I/O	Description
AE20	tck			I	Test clock
AD18	tdi	U		I	Test data in

Table 14: JTAG interface/boundary scan pinout

Bits	Signal name	U/D	OD (mA)	I/O	Description
AE19	tdo		2	O	Test data out
AC18	tms	U		I	Test mode select
AF20	trst_n	U		I	Test mode reset
AD19	rtck	U	2	I/O	Returned test clock, ARM core only

Table 14: JTAG interface/boundary scan pinout



JTAG 20 PIN HEADER..
Should be positioned on PCB with pin 1 facing toward board edge.

Notes
R8
out: Boot from flash/ROM/S_CS1n
in: Boot from SDRAM/CS0n using SPI_B EEPROM on GPIO pins

R12
out: Internal PCI arbiter
in: External PCI arbiter bus

Debug
Load all except JP1/R9, R15, R16; R8 and R12 depend on board options

Disable blank or unprogrammed boot memory

Production with debug possibility
Omit parts with **

Production without debug possibility
Omit parts with ** and ##, as well as parts with **

When halting the CPU in debug mode, the JSRST line must be pulsed low only one time.

Figure 9: JTAG interface

Reserved

Pin#	Description
J1	Tie to ground directly
K3	Tie to ground directly
K2	Tie to ground directly
K1	Tie to ground directly
R1	Tie to ground directly
R2	Tie to ground directly
R3	Tie to ground directly
T1	Tie to ground directly
AF6	Tie to ground directly
AE3	Tie to ground directly
AC5	Tie to ground directly
AD4	Tie to 1.5V core power
AF2	Tie to 3.3V I/O power
AE7	No connect
L3	No connect
L2	No connect
L1	No connect
M3	No connect
M2	Tie to ground directly
M1	Tie to ground directly
N1	Tie to ground directly
N2	Tie to ground directly
AF22	No connect
AD21	No connect
AE22	No connect

Table 15: Reserved pins

Power ground

Pin #	Signal name	Description
J23, L23, K23, U23, T23, V23, D18, D17, AC17, D16, AC16, D11, D10, AC11, AC10, AC9, J4, L4, K4, U4, T4, V4	VDDC	Core power, 1.5V
G23, H23, M23, R23, P23, N23, Y23, W23, D20, AC20, D19, AC19, D14, D13, AC14, AC13, D8, D7, AC8, AC7, G4, H4, M4, R4, P4, N4, Y4, W4	VDDS	I/O power, 3.3V
A26, B25, AE25, AF26, D23, C24, AD24, AC23, D5, D4, C4, E4, AC4, A3, A2, D3, C3, C2, B3, B2, AE2, AD3, A1, C1, B1, AF1	VSS2	Ground

Table 16: Power ground pins

Working with the CPU

C H A P T E R 3

The NS9750 core is based on the ARM926EJ-S processor. The ARM926EJ-S processor belongs to the ARM9 family of general-purpose microprocessors. The ARM926EJ-S processor is targeted at multi-tasking applications in which full memory management, high performance, low die size, and low power are important.

About the processor

The ARM926EJ-S processor supports the 32-bit ARM and 16-bit Thumb instructions sets, allowing you to trade off between high performance and high code density. The processor includes features for efficient execution of Java byte codes, providing Java performance similar to JIT but without the associated overhead.

The ARM926EJ-S supports the ARM debug architecture, and includes logic to assist in both hardware and software debug. The processor has a Harvard-cached architecture and provides a complete high-performance processor subsystem, including:

- ARM926EJ-S integer core
- Memory Management Unit (MMU) (see "Memory Management Unit (MMU)," beginning on page 78, for information)
- Separate instruction and data AMBA AHB bus interfaces

Figure 10 shows the main blocks in the ARM926EJ-S processor.

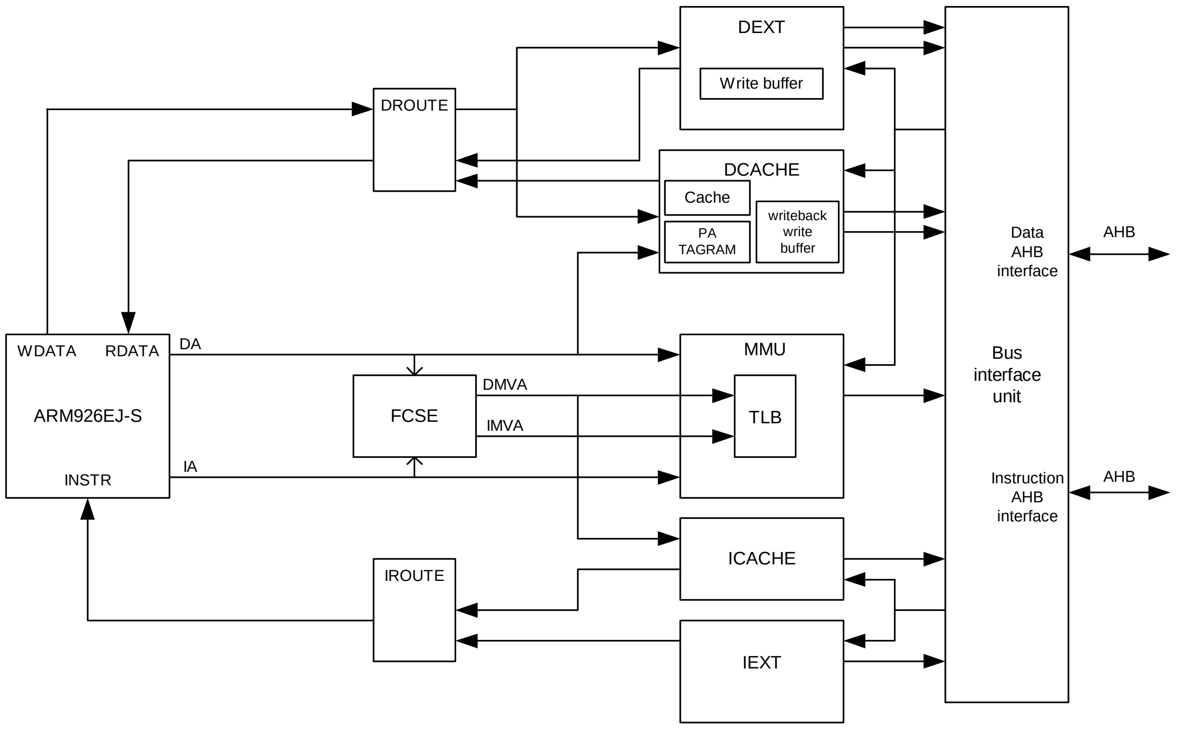


Figure 10: ARM926EJ-S processor block diagram

Instruction sets

The processor executes three instruction sets:

- 32-bit ARM instruction set
- 16-bit Thumb instruction set
- 8-bit Java instruction set

ARM instruction set

The ARM instruction set allows a program to achieve maximum performance with the minimum number of instructions. The majority of instructions are executed in a single cycle.

Thumb instruction set

The Thumb instruction set is simpler than the ARM instruction set, and offers increased code density for code that does not require maximum performance. Code can switch between ARM and Thumb instruction sets on any procedure call.

Java instruction set

In Java state, the processor core executes a majority of Java bytecodes naturally. Bytecodes are decoded in two states, compared to a single decode stage when in ARM/Thumb mode. See "Jazelle (Java)" on page 77 for more information about Java.

System control processor (CP15) registers

The system control processor (CP15) registers configure and control most of the options in the ARM926EJ-S processor. Access the CP15 registers using only the MRC and MCR instructions in a privileged mode; the instructions are provided in the explanation of each applicable register. Using other instructions, or MRC and MCR in unprivileged mode, results in an UNDEFINED instruction exception.

ARM926EJ-S system addresses

The ARM926EJ-S has three distinct types of addresses:

- In the ARM926EJ-S domain: Virtual address (VA)
- In the Cache and MMU domain: Modified virtual address (MVA)
- In the AMBA domain: Physical address (PA)

Example

This is an example of the address manipulation that occurs when the ARM926EJ-S core requests an instruction:

1. The ARM926EJ-S core issues the virtual address of the instruction.
2. The virtual address is translated using the FCSE PID (fast context switch extension process ID) value to the modified virtual address. The instruction cache (ICache) and memory management unit (MMU) find the modified virtual address (see "R13: Process ID register" on page 75).
3. If the protection check carried out by the MMU on the modified virtual address does not abort and the modified virtual address tag is in the ICache, the instruction data is returned to the ARM926EJ-S core.

   If the protection check carried out by the MMU on the modified virtual address does not abort but the cache misses (the MVA tag is not in the cache), the MMU translates the modified virtual address to produce the physical address. This address is given to the AMBA bus interface to perform an external access.

Accessing CP15 registers

Use only MRC and MCR instructions, only in privileged mode, to access CP15 registers. Figure 11 shows the MRC and MCR instruction bit pattern.

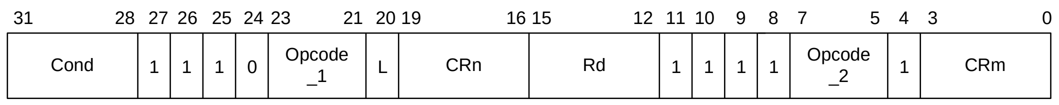


Figure 11: CP15 MRC and MCR bit pattern

The mnemonics for these instructions are:

MCR{cond} p15,opcode_1,Rd,CRn,CRm,opcode_2

MRC{cond} p15,opcode_1,Rd,CRn,CRm,opcode_2

If you try to read from a write-only register or write to a read-only register, you will have UNPREDICTABLE results. In all instructions that access CP15:

- The opcode_1 field SHOULD BE ZERO, except when the values specified are used to select the operations you want. Using other values results in unpredictable behavior.
- The opcode_2 and CRm fields SHOULD BE ZERO, except when the values specified are used to select the behavior you want. Using other values results in unpredictable behavior.

Terms and abbreviations

Table 17 lists the terms and abbreviations used in the CP15 registers and explanations.

Term	Abbreviation	Description
UNPREDICTABLE	UNP	**For reads:** The data returned when reading from this location is unpredictable, and can have any value. **For writes:** Writing to this location causes unpredictable behavior, or an unpredictable change in device configuration.

Table 17: CP15 terms and abbreviations

Term	Abbreviation	Description
UNDEFINED	UND	An instruction that accesses CP15 in the manner indicated takes the UNDEFINED instruction exception.
SHOULD BE ZERO	SBZ	When writing to this field, all bits of the field SHOULD BE ZERO.
SHOULD BE ONE	SBO	When writing to this location, all bits in this field SHOULD BE ONE.
SHOULD BE ZERO or PRESERVED	SBZP	When writing to this location, all bits of this field SHOULD BE ZERO or PRESERVED by writing the same value that has been read previously from the same field.

Table 17: CP15 terms and abbreviations

Note: In all cases, reading from or writing any data values to any CP15 registers, including those fields specified as UNPREDICTABLE, SHOULD BE ONE, or SHOULD BE ZERO, does not cause any physical damage to the chip.

Register summary

CP15 uses 16 registers.

- Register locations 0, 5, and 13 each provide access to more than one register. The register accessed depends on the value of the opcode_2 field in the CP15 MRC/MCR instructions (see "Accessing CP15 registers" on page 52).
- Register location 9 provides access to more than one register. The register accessed depends on the value of the CRm field (see "Accessing CP15 registers" on page 52).

Register	Reads	Writes
0	ID code (based on opcode_2 value)	Unpredictable
0	Cache type (based on opcode_2 value)	Unpredictable
1	Control	Control
2	Translation table base	Translation table base
3	Domain access control	Domain access control

Table 18: CP15 register summary

Register	Reads	Writes
4	Reserved	Reserved
5	Data fault status (based on opcode_2 value)	Data fault status (based on opcode_2 value)
6	Instruction fault status (based on opcode_2 value)	Instruction fault status (based on opcode_2 value)
7	Cache operations	Cache operations
8	Unpredictable	TLB
9	Cache lockdown (based on CRm value)	Cache lockdown
10	TLB lockdown	TLB lockdown
11 and 12	Reserved	Reserved
13	FCSE PID (based on opcode_2 value) FCSE = Fast context switch extension PID = Process identifier	FCSE PID (based on opcode_2 value) FCSE = Fast context switch extension PID = Process identifier
13	Context ID (based on opcode_2 value)	Context ID (based on opcode_2 value)
14	Reserved	Reserved
15	Test configuration	Test configuration

Table 18: CP15 register summary

All CP15 register bits that are defined and contain state are set to 0 by reset, with these exceptions:

- The V bit is set to 0 at reset if the VINITHI signal is low, and set to 1 if the VINITHI signal is high.
- The B bit is set to 0 at reset if the BIGENDINIT signal is low, and set to 1 if the BIGENDINIT signal is high.

R0: ID code and cache type status registers

Register R0 access the ID register, and cache type register. Reading from R0 returns the device ID, and the cache type, depending on the opcode_2 value:

opcode_2=0	ID value
opcode_2=1	instruction and data cache type

The CRm field SHOULD BE ZERO when reading from these registers. Table 19 shows the instructions you can use to read register R0.

Function	Instruction
Read ID code	MRC p15,0,Rd,c0,c0,{0, 3-7}
Read cache type	MRC p15,0,Rd,c0,c0,1

Table 19: Reading from register R0

Writing to register R0 is UNPREDICTABLE.

R0: ID code

R0: ID code is a read-only register that returns the 32-bit device ID code. You can access the ID code register by reading CP15 register R0 with the opcode_2 field set to any value other than 1 or 2. Note this example:

```
MRC p15, 0, Rd, c0, c0, {0, 3-7}; returns ID
```

Table 20 shows the contents of the ID code register.

Bits	Function	Value
[31:24]	ASCII code of implementer trademark	0x41
[23:20]	Specification revision	0x0
[19:16]	Architecture (ARMv5TEJ)	0x6
[15:4]	Part number	0x926
[3:0]	Layout revision	0x0

Table 20: R0: ID code

R0: Cache type register

R0: Cache type is a read-only register that contains information about the size and architecture of the instruction cache (ICache) and data cache (DCache) enabling operating systems to establish how to perform operations such as cache cleaning and lockdown. See "Cache features" on page 105 for more information about cache.

You can access the cache type register by reading CP15 register R0 with the opcode_2 field set to 1. Note this example:

```
MRC p15, 0, Rd, c0, c0, 1; returns cache details
```

Figure 12 shows the format of the cache type register. Table 21 describes the fields in the register.

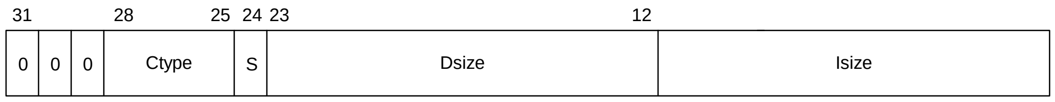


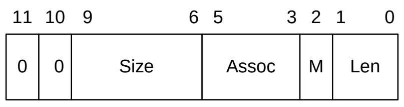
Figure 12: Cache type register format

Field	Description
Ctype	Determines the cache type, and specifies whether the cache supports lockdown and how it is cleaned. Ctype encoding is shown below; all unused values are reserved. Value: 0b1110 Method: Writeback Cache cleaning: Register 7 operations (see "R7: Cache Operations register" on page 64) Cache lockdown: Format C (see "R9: Cache Lockdown register" on page 69)
S bit	Specifies whether the cache is a unified cache (S=0) or separate ICache and DCache (S=1). Will always report separate ICache and DCache for NS9750.
Dsize	Specifies the size, line length, and associativity of the DCache.
Isize	Species the size, length and associativity of the ICache.

Table 21: Cache type register field definition

Dsize and Isize fields

The Dsize and Isize fields in the cache type register have the same format, as shown:



The field contains these bits:

Field	Description
Size	Determines the cache size in conjunction with the M bit. ■ The M bit is 0 for DCache and ICache. ■ The size field is bits [21:18] for the DCache and bits [9:6] for the ICache. ■ The minimum size of each cache is 4 KB; the maximum size is 128 KB. ■ Cache size encoding with M=0: **Size field** — **Cache size** 0b0011 — 4 KB 0b0100 — 8 KB **Note:** The NS9750 always reports 4KB for DCache and 8KB for ICache.
Assoc	Determines the cache associativity in conjunction with the M bit. ■ The M bit is 0 for both DCache and ICache. ■ The assoc field is bits [17:15 for the DCache and bits [5:3] for the ICache. ■ Cache associativity with encoding: **Assoc field** — **Associativity** 0b010 — 4-way Other values — Reserved
M bit	Multiplier bit. Determines the cache size and cache associativity values in conjunction with the size and assoc fields. **Note:** This field must be set to 0 for the ARM926EJ-S processor.
Len	Determines the line length of the cache. ■ The len field is bits [13:12] for the DCache and bits [1:0] for the ICache. ■ Line length encoding: **Len field** — **Cache line length** 10 — 8 words (32 bytes) Other values — Reserved

R1: Control register

Register R1 is the control register for the ARM926EJ-S processor. This register specifies the configuration used to enable and disable the caches and MMU (memory management unit). It is recommended that you access this register using a read-modify-write sequence.

For both reading and writing, the CRm and opcode_2 fields SHOULD BE ZERO. Use these instructions to read and write this register:

```
MRC p15, 0, Rd, c1, c0, 0 ; read control register
MCR p15, Rd, c1, c0, 0 ; write control register
```

All defined control bits are set to zero on reset except the V bit and B bit.

- The V bit is set to zero at reset if the VINITHI signal is low.
- The B bit is set to zero at reset if the BIGENDINIT signal is low, and set to one if the BIGENDINIT signal is high.

Figure 13 shows the Control register format. Table 22 describes the Control register bit functionality.

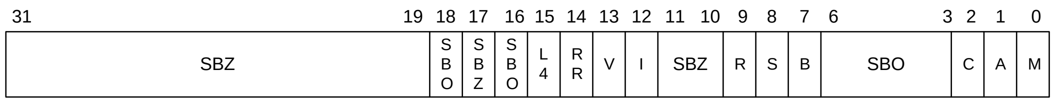


Figure 13: Control register format

Bits	Name	Function
[31:19]	N/A	Reserved: ■ When read, returns an UNPREDICTABLE value. ■ When written, SHOULD BE ZERO, or a value read from bits [31:19] on the same processor. ■ Use a read-modify-write sequence when modifying this register to provide the greatest future compatibility.
[18]	N/A	Reserved, SBO. Read = 1, write =1.
[17]	N/A	Reserved, SBZ. read = 0, write = 0.
[16]	N/A	Reserved, SBO. Read = 1, write = 1.

Table 22: R1: Control register bit definition

Bits	Name	Function
[15]	L4	Determines whether the T is set when load instructions change the PC. 0 Loads to PC set the T bit 1 Loads to PC do not set the T bit
[14]	RR bit	**Replacement strategy for ICache and DCache** 0 Random replacement 1 Round-robin replacement
[13]	V bit	**Location of exception vectors** 0 Normal exception vectors selected; address range=0x0000 0000 to 0x0000 001C 1 High exception vectors selected; address range=0xFFFF 0000 to 0xFFFF 001C Set to the value of VINITHI on reset.
[12]	I bit	**ICache enable/disable** 0 ICache disabled 1 ICache enabled
[11:10]	N/A	SHOULD BE ZERO
[9]	R bit	**ROM protection** Modifies the ROM protection system.
[8]	S bit	**System protection** Modifies the MMU protection system. See "Memory Management Unit (MMU)," beginning on page 78.
[7]	B bit	Endianness 0 Little endian operation 1 Big endian operation Set to the value of BIGENDINIT on reset.
[6:3]	N/A	Reserved. SHOULD BE ONE.
[2]	C bit	**DCache enable/disable** 0 Cache disabled 1 Cache enabled
[1]	A bit	**Alignment fault enable/disable** 0 Data address alignment fault checking disabled 1 Data address alignment fault checking enabled

Table 22: R1: Control register bit definition

Bits	Name	Function
[0]	M bit	**MMU enable/disable** 0 Disabled 1 Enabled

Table 22: R1: Control register bit definition

The M, C, I, and RR bits directly affect ICache and DCache behavior, as shown:

Cache	MMU	Behavior
ICache disabled	Enabled or disabled	All instruction fetches are from external memory (AHB).
ICache enabled	Disabled	All instruction fetches are cachable, with no protection checking. All addresses are flat-mapped; that is: VA=MVA=PA.
ICache enabled	Enabled	Instruction fetches are cachable or noncachable, and protection checks are performed. All addresses are remapped from VA to PA, depending on the MMU page table entry; that is, VA translated to MVA, MVA remapped to PA.
DCache disabled	Enabled or disabled	All data accesses are to external memory (AHB).
DCache enabled	Disabled	All data accesses are noncachable nonbufferable. All addresses are flat-mapped; that is, VA=MVA=PA.
DCache enabled	Enabled	All data accesses are cachable or noncachable, and protection checks are performed. All addresses are remapped from VA to PA, depending on the MMU page table entry; that is, VA translated to MVA, MVA remapped to PA.

Table 23: Effects of Control register on caches

If either the DCache or ICache is disabled, the contents of that cache are not accessed. If the cache subsequently is re-enabled, the contents will not have changed. To guarantee that memory coherency is maintained, the DCache must be cleaned of dirty data before it is disabled.

R2: Translation Table Base register

Register R2 is the Translation Table Base register (TTBR), for the base address of the first-level translation table.

- Reading from R2 returns the pointer to the currently active first-level translation table in bits [31:14] and an UNPREDICTABLE value in bits [13:0].
- Writing to R2 updates the pointer to the first-level translation table from the value in bits[31:14] of the written value. Bits [13:0] SHOULD BE ZERO.

Use these instructions to access the Translation Table Base register:

```
MRC p15, 0, Rd, c2, c0, 0 ; read TTBR
MCR p15, 0, Rd, c2, c0, 0 ; write TTBR
```

The CRm and opcode_2 fields SHOULD BE ZERO when writing to R2.

Figure 14 shows the format of the Translation Table Base register.

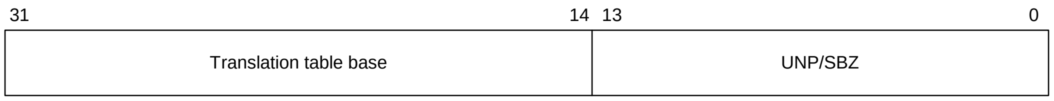


Figure 14: R2: Translation Table Base register

R3: Domain Access Control register

Register R3 is the Domain Access Control register and consists of 16 two-bit fields, as shown in Figure 15.

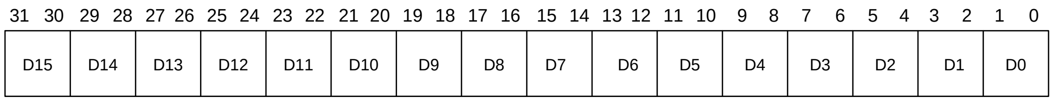


Figure 15: R3: Domain Access Control register

- Reading from R3 returns the value of the Domain Access Control register.
- Writing to R3 writes the value of the Domain Access Control register.

Each two-bit field defines the access permissions for one of the 16 domains (D15–D0):

00	No access: Any access generates a domain fault
01	Client: Accesses are checked against the access permission bits in the section or page descriptor
10	Reserved: Currently behaves like no access mode (00)
11	Manager: Accesses are not checked against the access permission bits, so a permission fault cannot be generated.

Use these instructions to access the Domain Access Control register:

```
MRC p15, 0, Rd, c3, c0, 0 ; read domain access permissions
MCR p15, 0, Rd, c3, c0, 0 ; write domain access permissions
```

R4 register

Accessing (reading or writing) this register causes UNPREDICTABLE behavior.

R5: Fault Status registers

Register R5 accesses the Fault Status registers (FSRs). The Fault Status registers contain the source of the last instruction or data fault. The instruction-side FSR is intended for debug purposes only.

The FSR is updated for alignment faults and for external aborts that occur while the MMU is disabled. The FSR accessed is determined by the opcode_2 value:

opcode_2=0	Data Fault Status register (DFSR)
opcode_2=1	Instruction Fault Status register (IFSR)

See "Memory Management Unit (MMU)," beginning on page 78, for the fault type encoding.

Access the FSRs using these instructions:

```
MRC p15, 0, Rd, c5, c0, 0 ; read DFSR
MCR p15, 0, Rd, c5, c0, 0 ; write DFSR
MRC p15, 0, Rd, c5, c0, 1 ; read IFSR
MCR p15, 0, Rd, c5, c0, 1 ; write IFSR
```

Figure 16 shows the format of the Fault Status registers. Table 24 describes the Fault Status register bits.

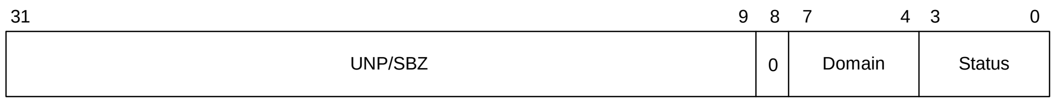


Figure 16: Fault Status registers format

Bits	Description
[31:9]	UNPREDICTABLE/SHOULD BE ZERO
[8]	Always reads as zero. Writes are ignored.
[7:4]	Specifies which of the 16 domains (D15–D0) was being accessed when a data fault occurred.
[3:0]	Type of fault generated. (See "Memory Management Unit (MMU)," beginning on page 78.)

Table 24: Fault Status register bit description

Table 25 shows the encodings used for the status field in the Fault Status register, and indicates whether the domain field contains valid information. See "MMU faults and CPU aborts" on page 95 for information about MMU aborts in Fault Address and Fault Status registers.

Priority	Source	Size	Status	Domain
Highest	Alignment	N/A	0b00x1	Invalid
	External abort on translation	First level Second level	0b1100 0b1110	Invalid Valid
	Translation	Section page	0b0101 0b0111	Invalid Valid
	Domain	Section page	0b1001 0b1011	Valid Valid
	Permission	Section page	0b1101 0b1111	Valid Valid

Table 25: Fault Status register status field encoding

Priority	Source	Size	Status	Domain
Lowest	External abort	Section page	0b1000 0b1010	Valid Valid

Table 25: Fault Status register status field encoding

R6: Fault Address register

Register R6 accesses the Fault Address register (FAR). The Fault Address register contains the modified virtual address of the access attempted when a data abort occurred. This register is updated only for data aborts, not for prefetch aborts; it is updated also for alignment faults and external aborts that occur while the MMU is disabled.

Use these instructions to access the Fault Address register:

```
MRC p15, 0, Rd, c6, c0, 0 ; read FAR
MCR p15, 0, Rd, c6, c0, 0 ; write FAR
```

Writing R6 sets the Fault Address register to the value of the data written. This is useful for debugging, to restore the value of a Fault Address register to a previous state.

The CRm and opcode_2 fields SHOULD BE ZERO when reading or writing R6.

R7: Cache Operations register

Register R7 controls the caches and write buffer. The function of each cache operation is selected by the opcode_2 and CRm fields in the MCR instruction that writes to CP15 R7. Writing other opcode_2 or CRm values is UNPREDICTABLE.

Reading from R7 is UNPREDICTABLE, with the exception of the two test and clean operations (see Table 27, "R7: Cache operations," on page 66 and "Test and clean operations" on page 67).

Use this instruction to write to the Cache Operations register:

```
MCR p15, opcode_1, Rd, CRn, CRm, opcode_2
```

Table 26 describes the cache functions provided by register R7. Table 27 lists the cache operation functions and associated data and instruction formats for R7.

Function	Description
Invalidate cache	Invalidates all cache data, including any dirty data.
Invalidate single entry using either index or modified virtual address	Invalidates a single cache line, discarding any dirty data.
Clean single data entry using either index or modified virtual address	Writes the specified DCache line to main memory if the line is marked valid and dirty. The line is marked as not dirty, and the valid bit is unchanged.
Clean and invalidate single data entry using wither index or modified virtual address.	Writes the specified DCache line to main memory if the line is marked valid and dirty. The line is marked not valid.
Test and clean DCache	Tests a number of cache lines, and cleans one of them if any are dirty. Returns the overall dirty state of the cache in bit 30. (See "Test and clean operations" on page 67).
Test, clean, and invalidate DCache	Tests a number of cache lines, and cleans one of them if any are dirty. When the entire cache has been tested and cleaned, it is invalidated. (See "Test and clean operations" on page 67).
Prefetch ICache line	Performs an ICache lookup of the specified modified virtual address. If the cache misses and the region is cachable, a linefill is performed.
Drain write buffer	Acts as an explicit memory barrier. This instruction drains the contents of the write buffers of all memory stores occurring in program order before the instruction is completed. No instructions occurring in program order after this instruction are executed until the instruction completes. Use this instruction when timing of specific stores to the level two memory system has to be controlled (for example, when a store to an interrupt acknowledge location has to complete before interrupts are enabled).

Table 26: Cache Operations register function descriptions

Function	Description
Wait for interrupt	Drains the contents of the write buffers, puts the processor into low-power state, and stops the processor from executing further instructions until an interrupt (or debug request) occurs. When an interrupt does occur, the MCR instruction completes, and the IRQ or FIRQ handler is entered as normal. The return link in R14_irq or R14_fiq contains the address of the MCR instruction plus eight, so the typical instruction used for interrupt return (SUBS PC,R14,#4) returns to the instruction following the MCR.

Table 26: Cache Operations register function descriptions

Function/operation	Data format	Instruction
Invalidate ICache and DCache	SBZ	MCR p15, 0, Rd, c7, c7, 0
Invalidate ICache	SBZ	MCR p15, 0, Rd, c7, c5, 0
Invalidate ICache single entry (MVA)	MVA	MCR p15, 0, Rd, c7, c5, 1
Invalidate ICache single entry (set/way)	Set/Way	MCR p15, 0, Rd, c7, c5, 2
Prefetch ICache line (MVA)	MVA	MCR p15, 0, Rd, c7, c13, 1
Invalidate DCache	SBZ	MCR p15, 0, Rd, c7, c6, 0
Invalidate DCache single entry (MVA)	MVA	MCR p15, 0, Rd, c7, c6, 1
Invalidate DCache single entry (set/way)	Set/Way	MCR p15, 0, Rd, c7, c6, 2
Clean DCache single entry (MVA)	MVA	MCR p15, 0, Rd, c7, c10, 1
Clean DCache single entry (set/way)	Set/Way	MCR p15, 0, Rd, c7, C10, 2
Test and clean DCache	n/a	MRC p15, 0, Rd, c7, c10, 3
Clean and invalidate DCache entry (MVA)	MVA	MCR p15, 0, Rd, c7, c14, 1
Clean and invalidate DCache entry (set/way)	Set/Way	MCR p15, 0, Rd, c7, c14, 2
Test, clean, and invalidate DCache	n/a	MRC p15, 0, Rd, c7, c14, 3
Drain write buffer	SBZ	MCR p15, 0, Rd, c7, c10, 4
Wait for interrupt	SBZ	MCR p15, 0, Rd, c7, c0, 4

Table 27: R7: Cache operations

Figure 17 shows the modified virtual address format for Rd for the CP15 R7 MCR operations.

- The tag, set, and word fields define the MVA.
- For all cache operations, the word field SHOULD BE ZERO.

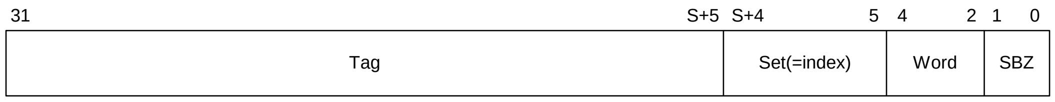


Figure 17: R7: MVA format

Figure 18 shows the Set/Way format for Rd for the CP15 R7 MCR operations.

- A and S are the base-two logarithms of the associativity and the number of sets.
- The set, way, and word files define the format.
- For all of the cache operations, word SHOULD BE ZERO.

For example, a 16 KB cache, 4-way set associative, 8-word line results in the following:

- A = $\log_2$ associativity = $\log_2 4 = 2$
- S = $\log_2$ NSETS where

  NSETS = cache size in bytes/associativity/line length in bytes:

  NSETS = 16384/4/32 = 128

  Result: S = $\log_2 128 = 7$

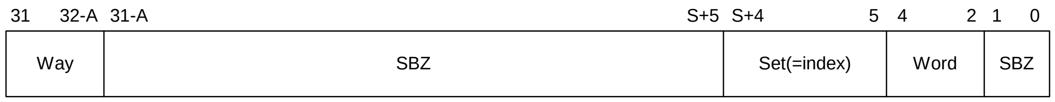


Figure 18: R7: Set/Way format

Test and clean operations

Test and clean DCache instruction

The test and clean DCache instruction provides an efficient way to clean the entire DCache, using a simple loop. The test and clean DCache instruction tests a number of lines in the DCache to determine whether any of them are dirty. If any dirty lines are found, one of those lines is cleaned. The test and clean DCache instruction also returns the status of the entire DCache in bit 30.

Note: The test and clean DCache instruction MRC p15, 0, r15, c7, c10, 3 is a special encoding that uses r15 as a destination operand. The PC is not changed by using this instruction, however. This MRC instruction also sets the condition code flags.

If the cache contains any dirty lines, bit 30 is set to 0. If the cache contains no dirty lines, bit 30 is set to 1. Use the following loop to clean the entire cache:

```
tc_loop:        MRC p15, 0, r15, c7, c10, 3 ; test and clean
                BNE tc_loop
```

Test, clean, and invalidate DCache instruction

The test, clean, and invalidate DCache instruction is the same as the test and clean DCache instruction except that when the entire cache has been cleaned, it is invalidated. Use the following loop to test, clean, and invalidate the entire DCache:

```
tci_loop:       MRC p15, 0, r15, c7, c14, 3 ; test clean and invalidate
                BNE tci_loop
```

R8:TLB Operations register

Register R8 is a write-only register that controls the translation lookaside buffer (TLB). There is a single TLB used to hold entries for both data and instructions. The TLB is divided into two parts:

- Set-associative
- Fully-associative

The *fully-associative* part (also referred to as the *lockdown* part of the TLB) stores entries to be locked down. Entries held in the lockdown part of the register are preserved during an invalidate-TLB operation. Entries can be removed from the lockdown TLB using an invalidate TLB single entry operation.

There are six TLB operations; the function to be performed is selected by the opcode_2 and CRm fields in the MCR instruction used to write register R8. Writing other opcode_2 or CRm values is UNPREDICTABLE. Reading from this register is UNPREDICTABLE.

Use the instruction shown in Table 28 to perform TLB operations.

Operation	Data	Instruction
Invalidate set-associative TLB	SBZ	MCR p15, 0, Rd, c8, c7, 0
Invalidate single entry	SBZ	MCR p15, 0, Rd, c8, c7. 1
Invalidate set-associative TLB	SBZ	MCR p15, 0, Rd, c8, c5, 0
Invalidate single entry	MVA	MCR p15, 0, Rd, c8, c5, 1
Invalidate set-associative TLB	SBZ	MCR p15, 0, Rd, c8, c6, 0
Invalidate single entry	MVA	MCR p15, 0, Rd, c8, c6, 1

Table 28: R8: Translation Lookaside Buffer operations

- The *invalidate TLB operations* invalidate all the unpreserved entries in the TLB.
- The *invalidate TLB single entry operations* invalidate any TLB entry corresponding to the modified virtual address given in Rd, regardless of its preserved state. See "R10: TLB Lockdown register," beginning on page 73, for an explanation of how to preserve TLB entries.

Figure 19 shows the modified virtual address format used for invalid TLB single entry operations.

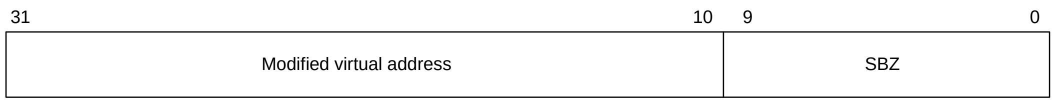


Figure 19: R8: TLB Operations, MVA format

Note: If either small or large pages are used, and these pages contain subpage access permissions that are different, you must use four invalidate TLB single entry operations, with the MVA set to each subpage, to invalidate all information related to that page held in a TLB.

R9: Cache Lockdown register

Register R9 access the cache lockdown registers. Access this register using CRm = 0.

The Cache Lockdown register uses a cache-way-based locking scheme (format C) that allows you to control each cache way independently.

These registers allow you to control which cache-ways of the four-way cache are used for the allocation on a linefill. When the registers are defined, subsequent linefills are placed only in the specified target cache way. This gives you some control over the cache pollution cause by particular applications, and provides a traditional lockdown operation for locking critical code into the cache.

A locking bit for each cache way determines whether the normal cache allocation is allowed to access that cache way (see Table 30, “Cache Lockdown register L bits,” on page 71). A maximum of three cache ways of the four-way associative cache can be locked, ensuring that normal cache line replacement is performed.

Note: If no cache ways have the L bit set to 0, cache way 3 is used for all linefills.

The first four bits of this register determine the L bit for the associated cache way. The opcode_2 field of the MRC or MCR instruction determines whether the instruction or data lockdown register is accessed:

opcode_2=0	Selects the DCache Lockdown register, or the Unified Cache Lockdown register if a unified cache is implemented. The ARM926EJ-S processor has separate DCache and ICache.
opcode_2=1	Selects the ICache Lockdown register.

Use the instructions shown in Table 29 to access the CacheLockdown register.

Function	Data	Instruction
Read DCache Lockdown register	L bits	MRC p15, 0, Rd, c9, c0, 0
Write DCache Lockdown register	L bits	MCR p15, 0, Rd, c9, c0, 0
Read ICache Lockdown register	L bits	MRC p15, 0, Rd, c9, c0, 1
Write ICache Lockdown register	L bits	MCR p15, 0, Rd, c9, c0, 1

Table 29: Cache Lockdown register instructions

You must modify the Cache Lockdown register using a modify-read-write sequence; for example:

```
MRC p15, 0, Rn, c9, c0, 1 ;
ORR Rn, Rn, 0x01 ;
MCR p15, 0, Rn, c9, c0, 1 ;
```

This sequence sets the L bit to 1 for way 0 of the ICache. Figure 20 shows the format for the Cache Lockdown register.

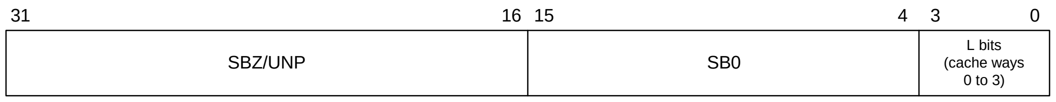


Figure 20: R9: Cache Lockdown register format

Table 30 shows the format of the Cache Lockdown register L bits. All cache ways are available for allocation from reset.

Bits	4-way associative	Notes
[31:16]	UNP/SBZ	Reserved
[15:4]	0xFFF	SBO
[3]	L bit for way 3	Bits [3:0] are the L bits for each cache way: 0 Allocation to the cache way is determined by the standard replacement algorithm (reset state) 1 No allocation is performed to this way
[2]	L bit for way 2	
[1]	L bit for way 1	
[0]	L bit for way 0	

Table 30: Cache Lockdown register L bits

Use one of these procedures to lockdown and unlock cache:

- Specific loading of addresses into a cache way
- Cache unlock procedure

Specific loading of addresses into a cache-way

The procedure to lock down code and data into way *i* of cache, with *N* ways, using format C, makes it impossible to allocate to any cache way other than the target cache way:

1. Be sure that no processor exceptions can occur during the execution of this procedure; for example, disable interrupts. If this is not possible, all code and data used by any exception handlers must be treated as code and data as in Steps 2 and 3.
2. If an ICache way is being locked down, be sure that all the code executed by the lockdown procedure is in an uncachable area of memory or in an already locked cache way.
3. If a DCache way is being locked down, be sure that all data used by the lockdown procedure is in an uncachable area of memory or is in an already locked cache way.
4. Ensure that the data/instructions that are to be locked down are in a cachable area of memory.
5. Be sure that the data/instructions that are to be locked down are not already in the cache. Use the Cache Operations register (R7) clean and/or invalidate functions to ensure this.
6. Write these settings to the Cache Lockdown register (R9), to enable allocation to the target cache way:

   CRm = 0

   Set L == 0 for bit i

   Set L == 1 for all other bits
7. For each of the cache lines to be locked down in cache way *i*:
   – If a DCache is being locked down, use an LDR instruction to load a word from the memory cache line to ensure that the memory cache line is loaded into the cache.
   – If an ICache is being locked down, use the Cache Operations register (R7) MCR prefetch ICache line (<CRm>==c13, <opcode2>==1) to fetch the memory cache line into the cache.
8. Write <CRm>==0 to Cache Lockdown register (R9), setting L==1 for bit *i* and restoring all other bits to the values they had before the lockdown routine was started.

Cache unlock procedure

To unlock the locked down portion of the cache, write to Cache Lockdown register (R9) setting L==0 for the appropriate bit. The following sequence, for example, sets the L bit to 0 for way 0 of the ICache, unlocking way 0:

```
MRC p15, 0, Rn, c9, c0, 1;
BIC Rn, Rn, 0x01 ;
MCR p15, 0, Rn, c9, c0, 1;
```

R10: TLB Lockdown register

The TLB Lockdown register controls where hardware page table walks place the TLB entry — in the set associative region or the lockdown region of the TLB. If the TLB entry is put in the lockdown region, the register indicates which entry is written. The TLB lockdown region contains eight entries (see the discussion of the TLB structure in "TLB structure," beginning on page 104, for more information).

Figure 21 shows the TLB lockdown format.

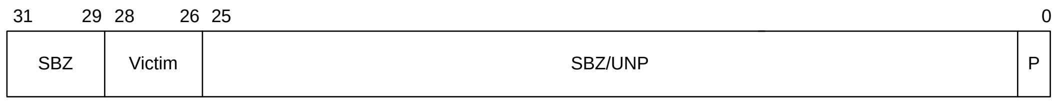


Figure 21: TLB Lockdown register format

When writing the TLB Lockdown register, the value in the P bit (D0) determines in which region the TLB entry is placed:

P=0	Subsequent hardware page table walks place the TLNB entry in the set associative region of the TLB.
P=1	Subsequent hardware page table walks place the TLB entry in the lockdown region at the entry specified by the victim, in the range 0–7.

TLB entries in the lockdown region are preserved so invalidate-TLB operations only invalidate the unpreserved entries in the TLB; that is, those entries in the set-associative region. Invalidate-TLB single entry operations invalidate any TLB entry corresponding to the modified virtual address given in Rd, regardless of the entry's preserved state; that is, whether they are in lockdown or set-associative TLB regions.

See "R8:TLB Operations register" on page 68 for a description of the TLB-invalidate operations.

Use these instructions to program the TLB Lockdown register:

Function	Instruction
Read data TLB lockdown victim	MRC p15, 0, Rd, c10, c0, 0
Write data TLB lockdown victim	MCR p15, 0, Rd, c10, c0, 0

The victim automatically increments after any table walk that results in an entry being written into the lockdown part of the TLB.

Note: It is not possible for a lockdown entry to map entirely either small or large pages, unless all subpage access permissions are the same. Entries can still be written into the lockdown region, but the address range that is mapped covers only the subpage corresponding to the address that was used to perform the page table walk.

Sample code sequence

This example shows the code sequence that locks down an entry to the current victim.

```
ADR r1,LockAddr           ;   set R1 to the value of the address to be locked
                              down
MCR p15,0,r1,c8,c7,1      ;   invalidate TLB single entry to ensure that
                              LockAddr is not already in the TLB
MRC p15,0,r0,c10,c0,0     ;   read the lockdown register
ORR r0,r0,#1              ;   set the preserve bit
MCR p15,0,r0,c10,c0,0     ;   write to the lockdown register
LDR r1,[r1]               ;   TLB will miss, and entry will be loaded
MRC p15,0,r0,c10,c0,0     ;   read the lockdown register (victim will have
                          ;   incremented
BIC r0,r0,#1              ;   clear preserve bit
MCR p15,0,r0,c10,c0,0     ;   write to the lockdown register
```

R11 and R12 registers

Accessing (reading or writing) these registers causes UNPREDICTABLE behavior.

R13: Process ID register

The Process ID register accesses the process identifier registers. The register accessed depends on the value on the opcode_2 field:

opcode_2=0	Selects the *Fast Context Switch Extension* (FCSE) *Process Identifier* (PID) register.
opcode_2=1	Selects the context ID register.

Use the Process ID register to determine the process that is currently running. The process identifier is set to 0 at reset.

FCSE PID register

Addresses issued by the ARM926EJ-S core, in the range 0 to 32 MB, are translated according to the value contained in the FCSE PID register. Address A becomes A + (FCSE PID x 32 MB); it is this modified address that the MMU and caches see. Addresses above 32 MB are not modified. The FCSE PID is a 7-bit field, which allows 128 x 32 MB processes to be mapped.

If the FCSE PID is 0, there is a flat mapping between the virtual addresses output by the ARM926EJ-S core and the modified virtual addresses used by the caches and MMU. The FCSE PID is set to 0 at system reset.

If the MMU is disabled, there is no FCSE address translation.

FCSE translation is not applied for addresses used for entry-based cache or TLB maintenance operations. For these operations, VA=MVA.

Use these instructions to access the FCSE PID register:

Function	Data	ARM instruction
Read FCSE PID	FCSE PID	MRC p15,0,Rd,c13,c0,0
Write FCSE PID	FCSE PID	MCR p15,0,Rd,c13,c0,0

Figure 22 shows the format of the FCSE PID register.

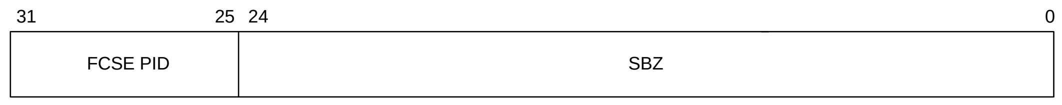


Figure 22: Process ID register format

Performing a fast context switch

You can perform a fast context switch by writing to the Process ID register (R13) with opcode_2 set to 0. The contents of the caches and the TLB do not have to be flushed after a fast context switch because they still hold address tags. The two instructions after the FCSE PID has been written have been fetched with the old FCSE PID, as shown in this code example:

```
{FCSE PID = 0}
    MOV r0, #1:SHL:25              ;Fetched with FCSE PID = 0
    MCR p15,0,r0,c13,c0,0          ;Fetched with FCSE PID = 0
    A1                             ;Fetched with FCSE PID = 0
    A2                             ;Fetched with FCSE PID = 0
    A3                             ;Fetched with FCSE PID = 1
```

A1, A2, and A3 are the three instructions following the fast context switch.

Context ID register

The Context ID register provides a mechanism that allows real-time trace tools to identify the currently executing process in multi-tasking environments.

Use these instructions to access the Context ID register:

Function	**Data**	**ARM instruction**
Read context ID	Context ID	MRC p15,0,Rd,c13,c0,1
Write context ID	Context ID	MCR p15,0,Rd,c13,c0,1

Figure 23 shows the format of the Context ID register (Rd) transferred during this operation.

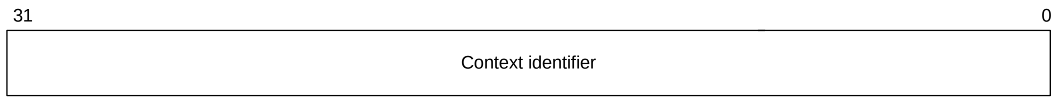


Figure 23: Context ID register format

R14 register

Accessing (reading or writing) this register is reserved.

R15: Test and debug register

Register R15 to provides device-specific test and debug operations in ARM926EJ-S processors. Use of this register currently is reserved.

Jazelle (Java)

The ARM926EJ-S processor has ARM's embedded Jazelle Java acceleration hardware in the core. Java offers rapid application development to software engineers.

The ARM926EJ-S processor core executes an extended ARMv5TE instruction set, which includes support for Java byte code execution (ARMv5TEJ). An ARM optimized *Java Virtual Machine (JVM)* software layer has been written to work with the Jazelle hardware. The Java byte code acceleration is accomplished by the following:

- Hardware, which directly executes 80% of simple Java byte codes.
- Software emulation within the ARM-optimized JVM, which addresses the remaining 20% of the Java byte codes.

DSP

The ARM926EJ-S processor core provides enhanced DSP capability. Multiply instructions are processed using a single cycle 32x16 implementation. There are 32x32, 32x16, and 16x16 multiply instructions, or *Multiply Accumulate (MAC)*, and the pipeline allows one multiply to start each cycle. Saturating arithmetic improves efficiency by automatically selecting saturating behavior during execution, and is used to set limits on signal processing calculations to minimize the effect of noise or signal errors. All of these instructions are beneficial for algorithms that implement the following:

- GSM protocols
- FFT
- State space servo control

Memory Management Unit (MMU)

The MMU provides virtual memory features required by systems operating on platforms such as WindowsCE or Linux. A single set of two-level page tables stored in main memory control the address translation, permission checks, and memory region attributes for both data and instruction accesses. The MMU uses a single, unified *Translation Lookaside Buffer (TLB)* to cache the information held in the page tables. TLB entries can be locked down to ensure that a memory access to a given region never incurs the penalty of a page table walk.

MMU Features

- Standard ARM926EJ-S architecture MMU mapping sizes, domains, and access protection scheme.
- Mapping sizes, as follows:
  - 1 MB for sections
  - 64 KB for large pages
  - 4 KB for small pages
  - 1 KB for tiny pages

- Access permissions for large pages and small pages can be specified separately for each quarter of the page (subpage permissions).
- Hardware page table walks.
- Invalidate entire TLB using R8: TLB Operations register (see "R8:TLB Operations register" on page 68).
- Invalidate TLB entry selected by MVA, using R8: TLB Operations register (see "R8:TLB Operations register" on page 68).
- Lockdown of TLB entries using R10: TLB Lockdown register (see "R10: TLB Lockdown register" on page 73).

Access permissions and domains

For large and small pages, access permissions are defined for each subpage (1 KB for small pages, 16 KB for large pages). Sections and tiny pages have a single set of access permissions.

All regions of memory have an associated domain. A domain is the primary access control mechanism for a region of memory. It defines the conditions necessary for an access to proceed. The domain determines whether:

- Access permissions are used to qualify the access.
- The access is unconditionally allowed to proceed.
- The access is unconditionally aborted.

In the latter two cases, the access permission attributes are ignored.

There are 16 domains, which are configured using R3: Domain Access Control register (see "R3: Domain Access Control register" on page 61).

Translated entries

The TLB caches translated entries. During CPU memory accesses, the TLB provides the protection information to the access control logic.

When the TLB contains a translated entry for the modified virtual address (MVA), the access control logic determines whether:

- Access is permitted and an off-chip access is required — the MMU outputs the appropriate physical address corresponding to the MVA.

- Access is permitted and an off-chip access is not required — the cache services the access.
- Access is not permitted — the MMU signals the CPU core to abort.

If the TLB misses (it does not contain an entry for the MVA), the translation table walk hardware is invoked to retrieve the translation information from a translation table in physical memory. When retrieved, the translation information is written into the TLB, possible overwriting an existing value.

At reset, the MMU is turned off, no address mapping occurs, and all regions are marked as noncachable and nonbufferable.

MMU program accessible registers

Table 31 shows the CP15 registers that are used in conjunction with page table descriptors stored in memory to determine MMU operation.

Register	Bits	Description
R1: Control register	M, A, S, R	Contains bits to enable the MMU (M bit), enable data address alignment checks (A bit), and to control the access protection scheme (S bit and R bit).
R2: Translation Table Base register	[31:14]	Holds the physical address of the base of the translation table maintained in main memory. This base address must be on a 16 KB boundary.
R3: Domain Access Control register	[31:0]	Comprises 16 two-bit fields. Each field defines the access control attributes for one of 16 domains (D15 to D00).
R5: Fault Status registers, IFSR and DFSR	[7:0]	Indicates the cause of a data or prefetch abort, and the domain number of the aborted access when an abort occurs. Bits [7:4] specify which of the 16 domains (D15 to D00) was being accessed when a fault occurred. Bits [3:0] indicate the type of access being attempted. The value of all other bits is UNPREDICTABLE. The encoding of these bits is shown in Table 32, "Priority encoding of fault status," on page 85).
R6: Fault Address register	[31:0]	Holds the MVA associated with the access that caused the data abort. See Table 32, "Priority encoding of fault status," on page 85 for details of the address stored for each type of fault.

Table 31: MMU program-accessible CP15 registers

Register	Bits	Description
R8: TLB Operations register	[31:0]	Performs TLB maintenance operations. These are either invalidating all the (unpreserved) entries in the TLB, or invalidating a specific entry.
R10: TLB Lockdown register	[28:26] and 0	Enables specific page table entries to be locked into the TLB. Locking entries in the TLB guarantees that accesses to the locked page or section can proceed without incurring the time penalty of a TLB miss. This enables the execution latency for time-critical pieces of code, such as interrupt handlers, to be minimized.

Table 31: MMU program-accessible CP15 registers

All CP15 MMU registers, except R8: TLB Operations, contain state that can be read using MRC instructions, and can be written using MCR instructions. Registers R5 (Fault Status) and R6 (Fault Address) are also written by the MMU during an abort.

Writing to R8: TLB Operations causes the MMU to perform a TLB operation, to manipulate TLB entries. This register is write-only.

Address translation

The virtual address (VA) generated by the CPU core is converted to a modified virtual address (MVA) by the FCSE (fast context switch extension) using the value held in CP15 R13: Process ID register. The MMU translates MVAs into physical addresses to access external memory, and also performs access permission checking.

The MMU table-walking hardware adds entries to the TLB. The translation information that comprises both the address translation data and the access permission data resides in a translation table located in physical memory. The MMU provides the logic for automatically traversing this translation table and loading entries into the TLB.

The number of stages in the hardware table walking and permission checking process is one or two. depending on whether the address is marked as a section-mapped access or a page-mapped access.

There are three sizes of page-mapped accesses and one size of section-mapped access. Page-mapped accesses are for large pages, small pages, and tiny pages.

The translation process always begins in the same way — with a level-one fetch. A section-mapped access requires only a level-one fetch, but a page-mapped access requires an additional level-two fetch.

Translation table base

The hardware translation process is initiated when the TLB does not contain a translation for the requested MVA. R2: Translation Table Base (TTB) register points to the base address of a table in physical memory that contains section or page descriptors, or both. The 14 low-order bits [13:0] of the TTB register are UNPREDICTABLE on a read, and the table must reside on a 16 KB boundary.

Figure 24 shows the format of the TTB register.

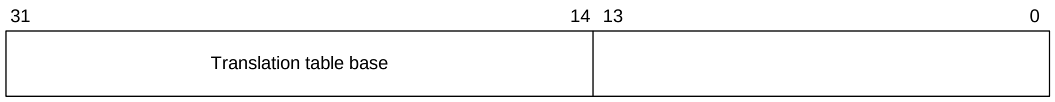


Figure 24: R2: Translation Table base register

The translation table has up to 4096 x 32-bit entries, each describing 1 MB of virtual memory. This allows up to 4 GB of virtual memory to be addressed.

Figure 25 shows the table walk process.

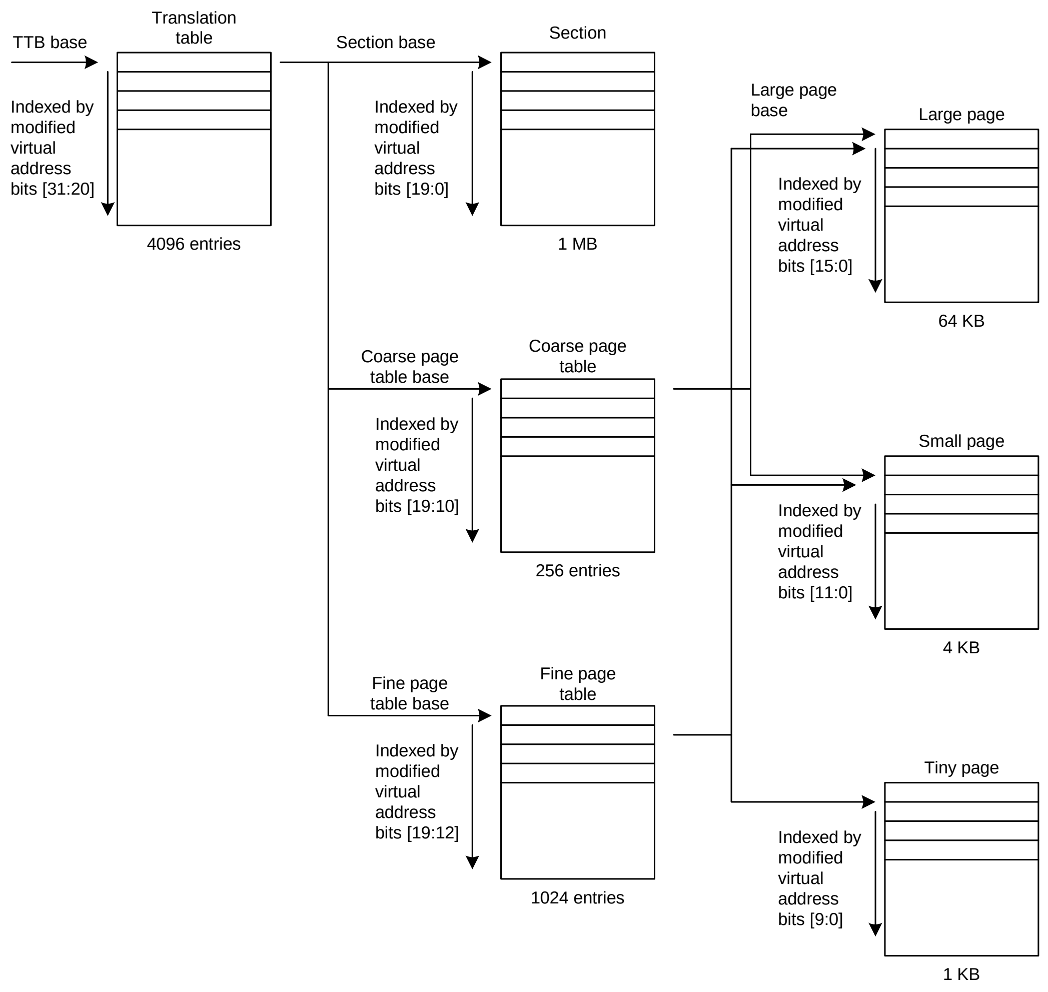

TTB base
Translation table
Indexed by modified virtual address bits [31:20]
4096 entries
Section base
Section
Indexed by modified virtual address bits [19:0]
1 MB
Large page base
Large page
Indexed by modified virtual address bits [15:0]
64 KB
Coarse page table base
Coarse page table
Indexed by modified virtual address bits [19:10]
256 entries
Small page
Indexed by modified virtual address bits [11:0]
4 KB
Fine page table base
Fine page table
Indexed by modified virtual address bits [19:12]
1024 entries
Tiny page
Indexed by modified virtual address bits [9:0]
1 KB

Figure 25: Translating page tables

First-level fetch

Bits [31:14] of the TTB register are concatenated with bits [31:20] of the MVA to produce a 30-bit address.

Figure 26 shows the concatenation and address:

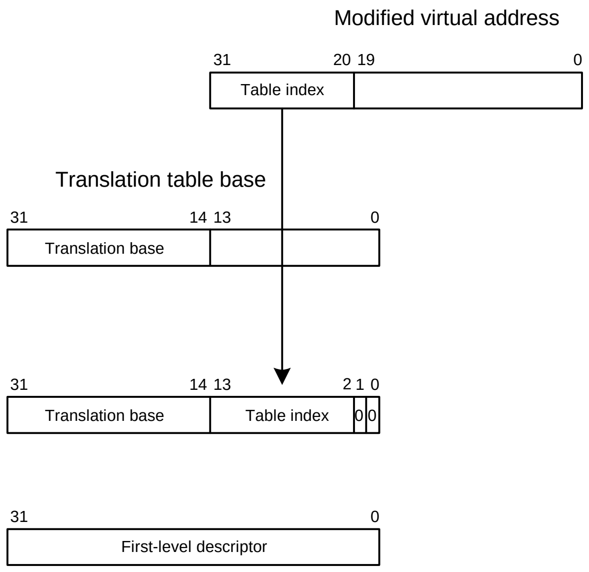


Figure 26: Accessing translation table first-level descriptors

This address selects a 4-byte translation table entry. This is a first-level descriptor for either a section or a page.

First-level descriptor

The first-level descriptor returned is a section description, a coarse page table descriptor, a fine page table descriptor, or is invalid. Figure 27 shows the format of a first-level descriptor.

A section descriptor provides the base address of a 1 MB block of memory.

The page table descriptors provide the base address of a page table that contains second-level descriptors. There are two page-table sizes:

- **Coarse page tables,** which have 256 entries and split the 1 MB that the table describes into 4 KB blocks.

- **Fine page tables,** which have 1024 entries and split the 1 MB that the table describes into 1 KB blocks.

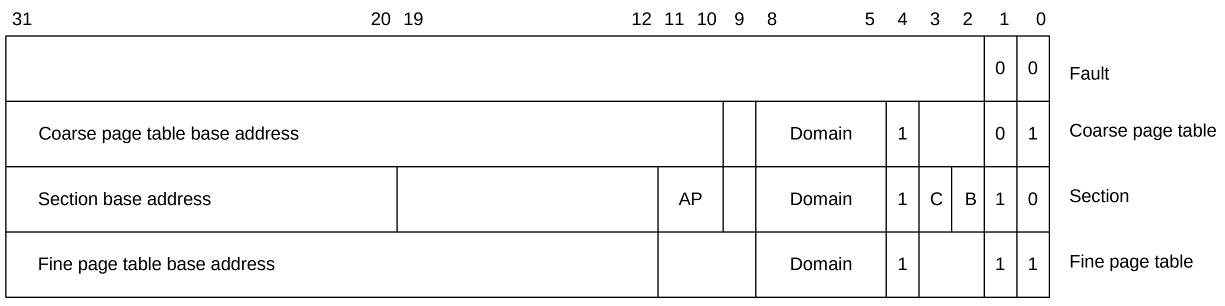


Figure 27: First-level descriptor

Table 32 shows first-level descriptor bit assignments.

Bits			
Section	**Coarse**	**Fine**	**Description**
[31:20]	[31:10]	[31:12]	Forms the corresponding bits of the physical address.
[19:12]	----	---	SHOULD BE ZERO
[11:10]	---	---	Access permission bits. See "Access permissions and domains" on page 79 and "Fault Address and Fault Status registers" on page 96 for information about interpreting the access permission bits.
9	9	[11:9]	SHOULD BE ZERO
[8:5]	[8:5]	[8:5]	Domain control bits
4	4	4	Must be 1.
[3:2]	---	---	Bits C and B indicate whether the area of memory mapped by this page is treated as write-back cachable, write-through cachable, noncached buffered, or noncached nonbuffered.
---	[3:2]	[3:2]	SHOULD BE ZERO

Table 32: Priority encoding of fault status

Bits			
Section	**Coarse**	**Fine**	**Description**
[1:0]	[1:0]	[1:0]	These bits indicate the page size and validity, and are interpreted as shown in Table 33, "Interpreting first-level descriptor bits [1:0]," on page 86.

Table 32: Priority encoding of fault status

Value	Meaning	Description
0 0	Invalid	Generates a section translation fault.
0 1	Coarse page table	Indicates that this is a coarse page table descriptor.
1 0	Section	Indicates that this is a section descriptor.
1 1	Fine page table	Indicates that this is a fine page table descriptor.

Table 33: Interpreting first-level descriptor bits [1:0]

Section descriptor

A section descriptor provides the base address of a 1 MB block of memory. Figure 28 shows the section descriptor format. Table 34 describes the section descriptor bits.

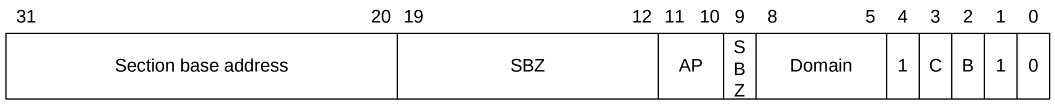


Figure 28: Section descriptor

Bits	Description
[31:20]	Forms the corresponding bits of the physical address for a section.
[19:12]	Always written as 0.
[11:10]	Specify the access permissions for this section.
[09	Always written as 0.

Table 34: Section descriptor bits

Bits	Description
[8:5]	Specifies one of the 16 possible domains (held in the Domain and Access Control register) that contain the primary access controls.
4	Should be written as 1, for backwards compatibility.
[3:2]	Indicate if the area of memory mapped by this section is treated as writeback cachable, write-through cachable, noncached buffered, or noncached nonbuffered.
[1:0]	Must be *10* to indicate a section descriptor.

Table 34: Section descriptor bits

Coarse page table descriptor

A coarse page table descriptor provides the base address of a page table that contains second-level descriptors for either large page or small page accesses. Coarse page tables have 256 entries, splitting the 1 MB that the table describes into 4 KB blocks. Figure 29 shows the coarse page table descriptor format; Table 35 describes the coarse page table descriptor bit assignments.

Note: If a coarse page table descriptor is returned from the first-level fetch, a second-level fetch is initiated.

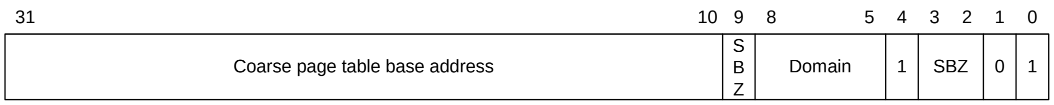


Figure 29: Coarse page table descriptor

Bits	Description
[31:10]	Forms the base for referencing the second-level descriptor (the coarse page table index for the entry derived from the MVA).
9	Always written as 0.
[8:5]	Specifies one of the 16 possible domains (held in the Domain Access Control registers) that contain the primary access controls.
4	Always written as 1.
[3:2]	Always written as 0.

Table 35: Coarse page table descriptor bits

Bits	Description
[1:0]	Must be *01* to indicate a coarse page descriptor.

Table 35: Coarse page table descriptor bits

Fine page table descriptor

A fine page table descriptor provides the base address of a page table that contains second-level descriptors for large page, small page, or tiny page accesses. Fine page tables have 1024 entries, splitting the 1 MB that the table describes into 1 KB blocks. Figure 30 shows the format of a fine page table descriptor. Table 36 describes the fine page table descriptor bit assignments.

Note: If a fine page table descriptor is returned from the first-level fetch, a second-level fetch is initiated.

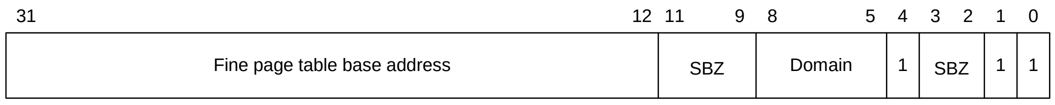


Figure 30: Fine page table descriptor

Bits	Description
[31:12]	Forms the base for referencing the second-level descriptor (the fine page table index for the entry is derived from the MVA).
[11:9]	Always written as 0.
[8:5]	Specifies one of the 16 possible domains (held in the Domain Access Control register) that contain primary access controls.
4	Always written as 1.
[3:2}	Always written as 0.
[1:0]	Must be *11* to indicate a fine page table descriptor.

Table 36: Fine page table descriptor bits

Translating section references

Figure 31 shows the complete section translation sequence.

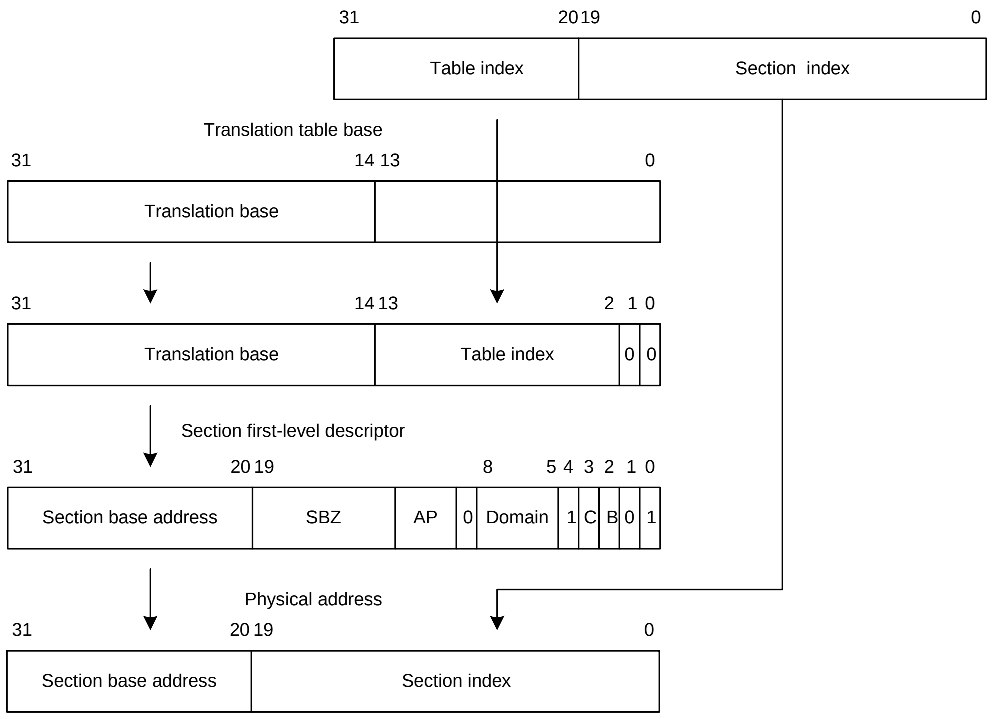


Figure 31: Section translation

Second-level descriptor

The base address of the page table to be used is determined by the descriptor returned (if any) from a first-level fetch — either a coarse page table descriptor or a fine page table descriptor. The page table is then accessed and a second-level descriptor returned.

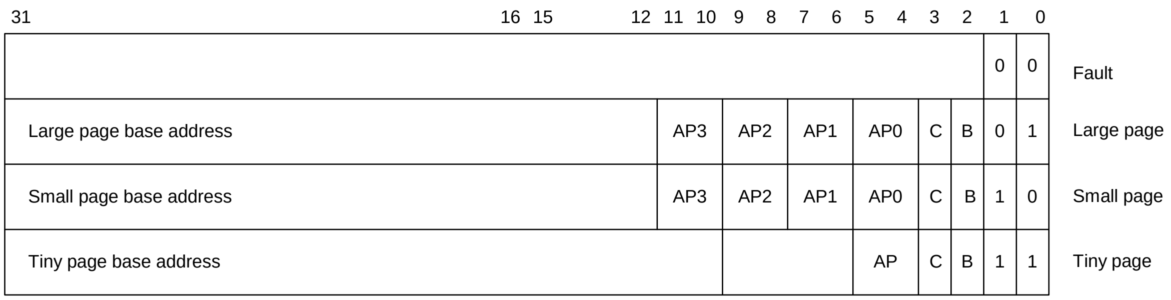


Figure 32: Second-level descriptor

A second-level descriptor defines a tiny, small, or large page descriptor, or is invalid:

- A large page descriptor provides the base address of a 64 KB block of memory.
- A small page descriptor provides the base address of a 4 KB block of memory.
- A tiny page descriptor provides the base address of a 1 KB block of memory.

Coarse page tables provide base addresses for either small or large pages. Large page descriptors must be repeated in 16 consecutive entries. Small page descriptors must be repeated in each consecutive entry.

Fine page tables provide base addresses for large, small, or tiny pages. Large page descriptors must be repeated in 64 consecutive entries. Small page descriptors must be repeated in four consecutive entries. Tiny page descriptors must be repeated in each consecutive entry.

Table 37 describes the second-level descriptor bit assignments.

Bits			
Large	**Small**	**Tiny**	**Description**
[31:16]	[31:12]	[31:10]	Form the corresponding bits of the physical address.
[15:12]	---	[9:6]	SHOULD BE ZERO

Table 37: Second-level descriptor bits

Bits			
Large	Small	Tiny	Description
[11:4]	[11:4]	[5:4]	Access permission bits. See "Domain access control" on page 98 and "Fault checking sequence" on page 99 for information about interpreting the access permission bits.
[3:2]	[3:2]	[3:2]	Indicate whether the area of memory mapped by this page is treated as write-back cachable, write-through cachable, noncached buffered, and noncached nonbuffered.
[1:0]	[1:0]	[1:0]	Indicate the page size and validity, and are interpreted as shown in Table 38, “Interpreting page table entry bits [1:0],” on page 91.

Table 37: Second-level descriptor bits

The two least significant bits of the second-level descriptor indicate the descriptor type; see Table 38.

Value	Meaning	Description
0 0	Invalid	Generates a page translation fault.
0 1	Large page	Indicates that this is a 64 KB page.
1 0	Small page	Indicates that this is a 4 KB page.
1 1	Tiny page	Indicates that this is a 1 KB page.

Table 38: Interpreting page table entry bits [1:0]

Note: Tiny pages do not support subpage permissions and therefore have only one set of access permission bits.

Translating large page references

Figure 33 shows the complete translation sequence for a 64 KB large page.

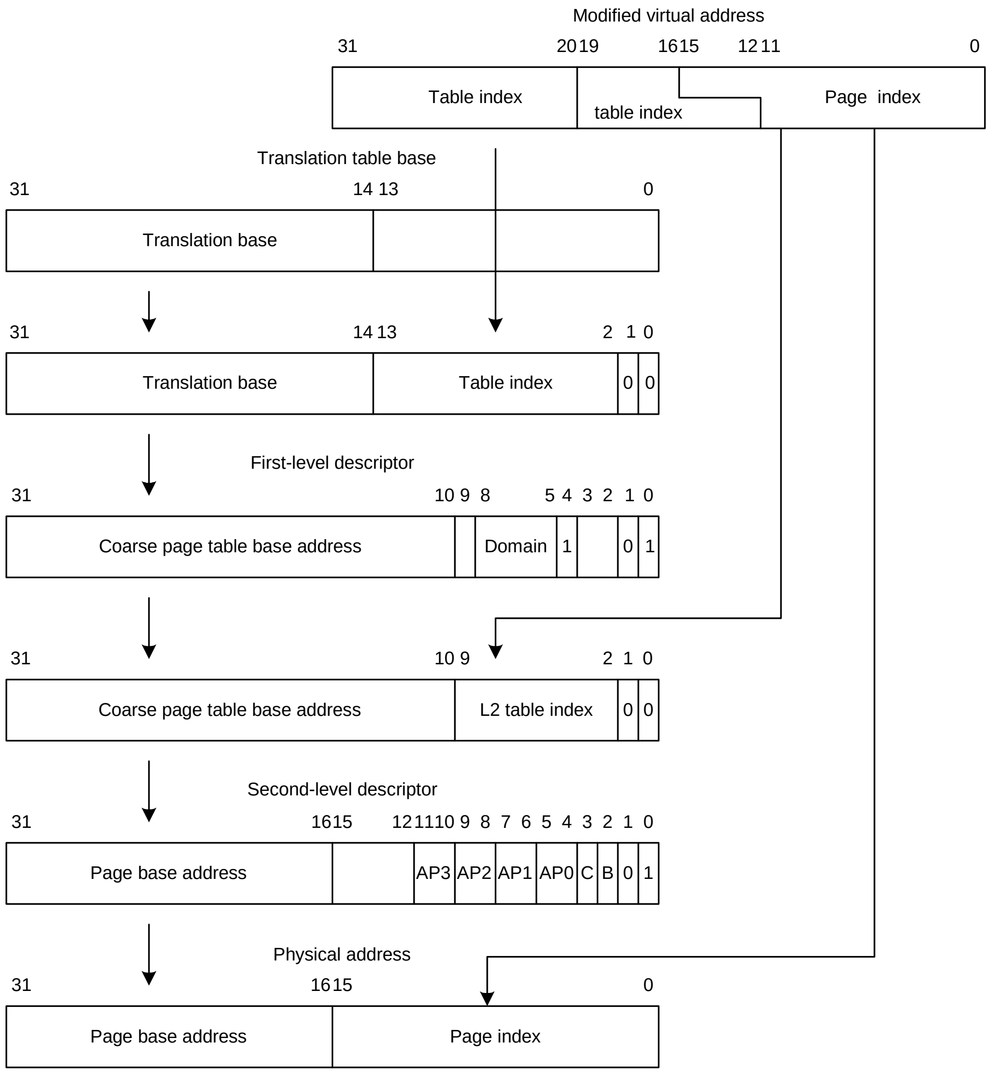


Figure 33: Large page translation from a coarse page table

Because the upper four bits of the page index and low-order four bits of the coarse page table index overlap, each coarse page table entry for a large page must be duplicated 16 times (in consecutive memory locations) in the coarse page table.

If the large page descriptor is included in a fine page table, the high-order six bits of the page index and low-order six bits of the fine page table overlap. Each fine page table entry for a large page must be duplicated 64 times.

Translating small page references

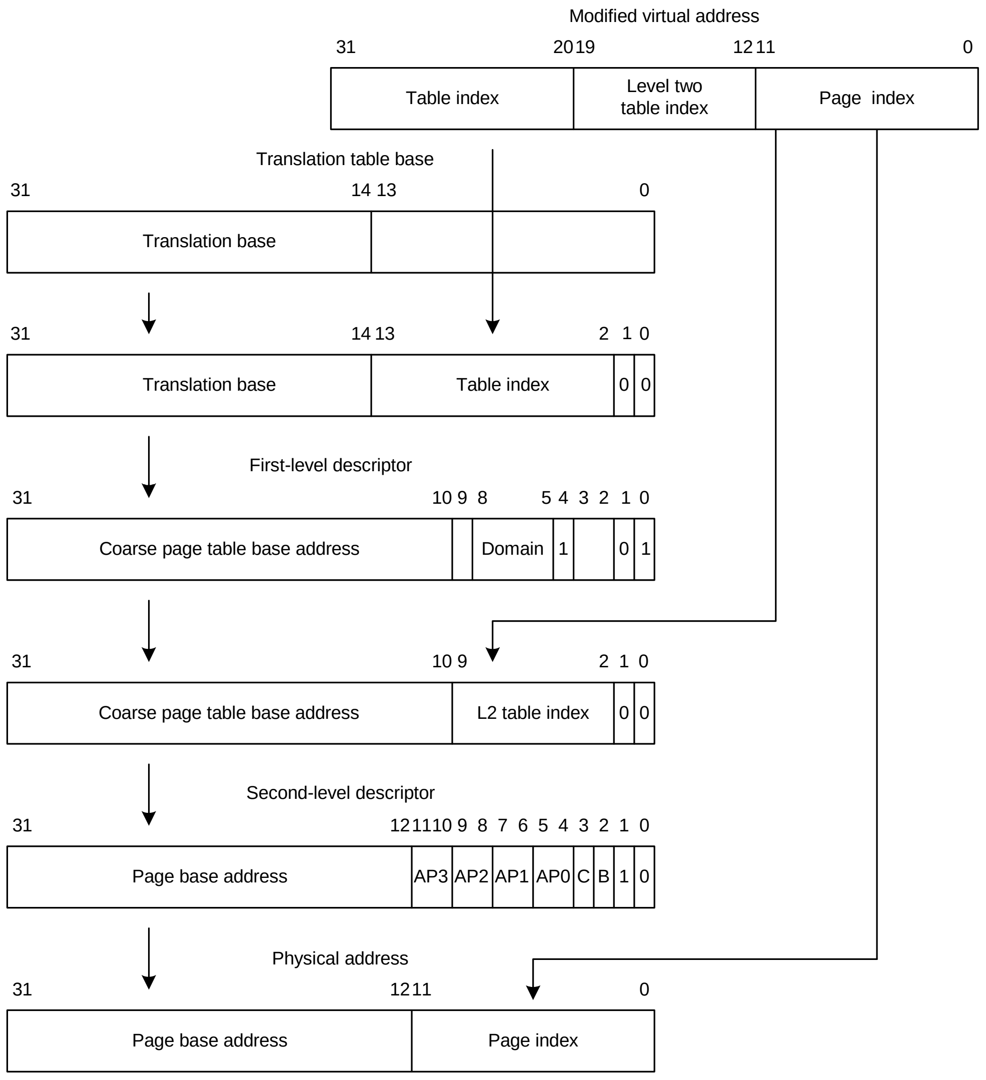


Figure 34: Small page translation from a coarse page table

If a small page descriptor is included in a fine page table, the upper two bits of the page index and low-order two bits of the fine page table index overlap. Each fine page table entry for a small page must be duplicated four times.

Translating tiny page references

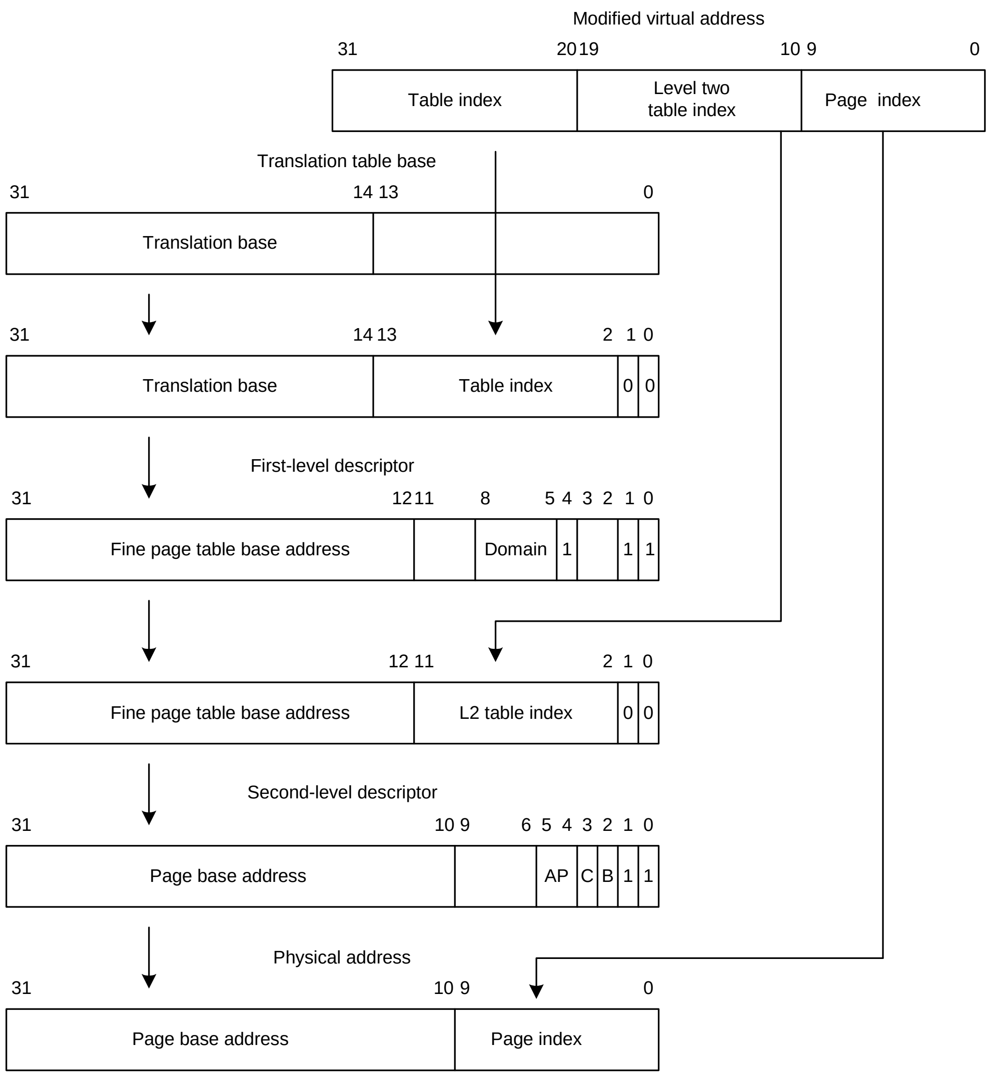


Figure 35: Tiny page translation from a fine page table

Page translation involves one additional step beyond that of a section translation. The first-level descriptor is the fine page table descriptor; this points to the first-level descriptor.

Note: The domain specified in the first-level description and access permissions specified in the first-level description together determine whether the access has permissions to proceed. See "Domain access control" on page 98 for more information.

Subpages

You can define access permissions for subpages of small and large pages. If, during a page table walk, a small or large page has a different subpage permission, only the subpage being accessed is written into the TLB. For example, a 16 KB (large page) subpage entry is written into the TLB if the subpage permission differs, and a 64 KB entry is put in the TLB if the subpage permissions are identical.

When you use subpage permissions and the page entry has to be invalidated, you must invalidate all four subpages separately.

MMU faults and CPU aborts

The MMU generates an abort on these types of faults:

- Alignment faults (data accesses only)
- Translation faults
- Domain faults
- Permission faults

In addition, an external abort can be raised by the external system. This can happen only for access types that have the core synchronized to the external system:

- Page walks
- Noncached reads
- Nonbuffered writes
- Noncached read-lock-write sequence (SWP)

Alignment fault checking is enabled by the A bit in the R1: Control register. Alignment fault checking is not affected by whether the MMU is enabled. Translation, domain, and permission faults are generated only when the MMU is enabled.

The access control mechanisms of the MMU detect the conditions that produce these faults. If a fault is detected as a result of a memory access, the MMU aborts the access and signals the fault condition to the CPU core. The MMU retains status and address information about faults generated by the data accesses in the Data Fault Status register and Fault Address register (see "Fault Address and Fault Status registers" on page 96).

The MMU also retains status about faults generated by instruction fetches in the Instruction Fault Status register.

An access violation for a given memory access inhibits any corresponding external access to the AHB interface, with an abort returned to the CPU core.

Fault Address and Fault Status registers

On a data abort, the MMU places an encoded four-bit value — the *fault status* — along with the four-bit encoded domain number in the Data Fault Status register. Similarly, on a prefetch abort, the MMU places an encoded four-bit value along with the four-bit encoded domain number in the Instruction Fault Status register. In addition, the MVA associated with the data abort is latched into the Fault Address register. If an access violation simultaneously generates more than one source of abort, the aborts are encoded in the priority stated in Table 39. The Fault Address register is not updated by faults caused by instruction prefetches.

Priority	Source	Size	Status	Domain
Highest	Alignment	---	0b00x1	Invalid
	External abort on transmission	First level Second level	0b1100 0b1110	Invalid Valid
	Translation	Section page	0b0101 0b0111	Invalid Valid
	Domain	Section page	0b1001 0b1011	Valid Valid
	Permission	Section page	0b1101 0b1111	Valid Valid
Lowest	External abort	Section page	0b1000 0b1010	Valid Valid

Table 39: Priority encoding of fault status

Notes:

- Alignment faults can write either 0b0001 or 0b0011 into Fault Status register [3:0].
- Invalid values can occur in the status bit encoding for domain faults. This happens when the fault is raised before a valid domain field has been read from a page table description.
- Aborts masked by a higher priority abort can be regenerated by fixing the cause of the higher priority abort, and repeating the access.
- Alignment faults are not possible for instruction fetches.
- The Instruction Fault Status register can be updated for instruction prefetch operations (MCR p15,0,Rd,c7,c13,1).

Fault Address register (FAR)

For load and store instructions that can involve the transfer of more than one word (LDM/STM, STRD, and STC/LDC), the value written into the Fault Address register depends on the type of access and, for external aborts, on whether the access crosses a 1 KB boundary. Table 40 shows the Fault Address register values for multi-word transfers.

Domain	Fault Address register
Alignment	MVA of first aborted address in transfer
External abort on translation	MVA of first aborted address in transfer
Translation	MVA of first aborted address in transfer
Domain	MVA of first aborted address in transfer
Permission	MVA of first aborted address in transfer
External about for noncached reads, or nonbuffered writes	MVA of last address before 1KB boundary, if any word of the transfer before 1 KB boundary is externally aborted. MVA of last address in transfer if the first externally aborted word is after the 1 KB boundary.

Table 40: Fault Address register values for multi-word transfers

Compatibility issues

- To enable code to be ported easily to future architectures, it is recommended that no reliance is made on external abort behavior.
- The Instruction Fault Status register is intended for debugging purposes only.

Domain access control

MMU accesses are controlled primarily through the use of domains. There are 16 domains, and each has a two-bit field to define access to it. Client users and Manager users are supported.

The domains are defined in the R3: Domain Access Control register. Figure 15, "R3: Domain Access Control register," on page 61 shows how the 32 bits of the register are allocated to define the 16 two-bit domains.

Table 41 shows how the bits within each domain are defined to specify access permissions.

Value	Meaning	Description
0 0	No access	Any access generates a domain fault.
0 1	Client	Accesses are checked against the access permission bits in the section or page descriptor.
1 0	Reserved	Reserved. Currently behaves like *no access* mode.
1 1	Manager	Accesses are not checked against the access permission bits, so a permission fault cannot be generated.

Table 41: Domain Access Control register, access control bits

Table 42 shows how to interpret the *access permission (AP)* bits, and how the interpretation depends on the R and S bits in the R1: Control register (see "R1: Control register," beginning on page 58).

AP	S	R	Privileged permissions	User permissions
0 0	0	0	No access	No access

Table 42: Interpreting access permission (AP) bits

AP	S	R	Privileged permissions	User permissions
0 0	1	0	Read only	Read only
0 0	0	1	Read only	Read only
0 0	1	1	UNPREDICTABLE	UNPREDICTABLE
0 1	x	x	Read/write	No access
1 0	x	x	Read/write	Read only
1 1	x	x	Read/write	Read/write

Table 42: Interpreting access permission (AP) bits

Fault checking sequence

The sequence the MMU uses to check for access faults is different for sections and pages. Figure 36 shows the sequence for both types of access.

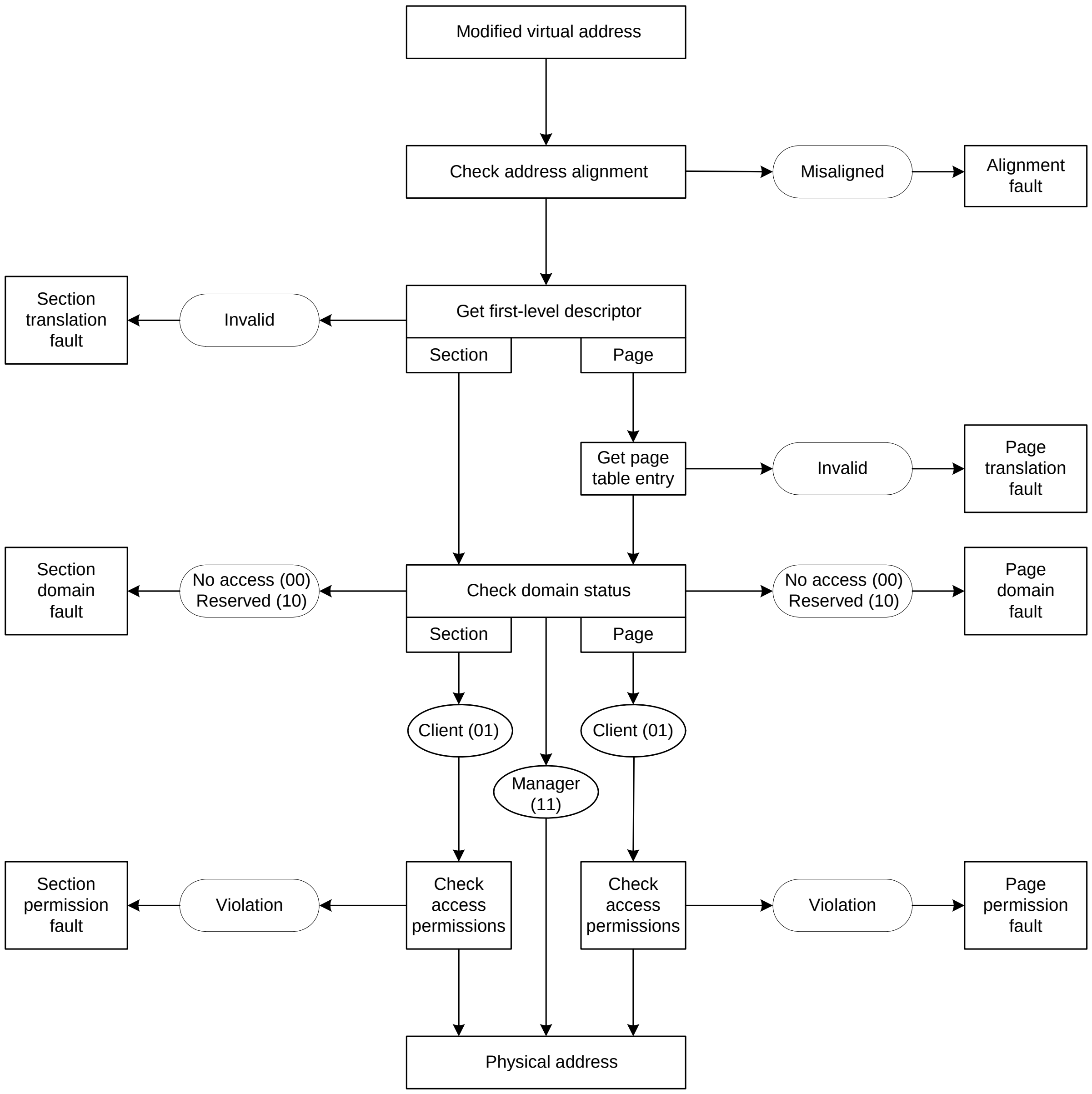


Figure 36: Sequence for checking faults

The conditions that generate each of the faults are discussed in the following sections.

Alignment faults

If alignment fault checking is enabled (the A bit in the R1: Control register is set; see "R1: Control register," beginning on page 58), the MMU generates an alignment fault on any data word access if the address is not word-aligned, or on any halfword access if the address is not halfword-aligned — irrespective of whether the MMU is enabled. An alignment fault is not generated on any instruction fetch or byte access.

Note: If an access generates an alignment fault, the access sequence aborts without reference to other permission checks.

Translation faults

There are two types of translation fault: section and page.

- A section translation fault is generated if the level one descriptor is marked as invalid. This happens if bits [1:0] of the descriptor are both 0.
- A page translation fault is generated if the level one descriptor is marked as invalid. This happens if bits [1:0] of the descriptor are both 0.

Domain faults

There are two types of domain faults: section and page.

- **Section:** The level one descriptor holds the four-bit domain field, which selects one of the 16 two-bit domains in the Domain Access Control register. The two bits of the specified domain are then checked for access permissions as described in Table 42: "Interpreting access permission (AP) bits" on page 98. The domain is checked when the level one descriptor is returned.
- **Page:** The level one descriptor holds the four-bit domain field, which selects one of the 16 two-bit domains in the Domain Access Control register. The two bits of the specified domain are then checked for access permissions as described in Table 42: "Interpreting access permission (AP) bits" on page 98. The domain is checked when the level one descriptor is returned.

If the specified access is either *no access* (00) or *reserved* (10), either a section domain fault or a page domain fault occurs.

Permission faults

If the two-bit domain field returns *client* (01), access permissions are checked as follows:

- **Section:** If the level one descriptor defines a section-mapped access, the AP bits of the descriptor define whether the access is allowed, per Table 42: "Interpreting access permission (AP) bits" on page 98. The interpretation depends on the setting of the S and R bits (see "R1: Control register," beginning on page 58). If the access is not allowed, a section permission fault is generated.
- **Large page or small page:** If the level one descriptor defines a page-mapped access and the level two descriptor is for a large or small page, four access permission fields (AP3 to AP0) are specified, each corresponding to one quarter of the page.

  For small pages, AP3 is selected by the top 1 KB of the page and AP0 is selected by the bottom 1 KB of the page.

  For large pages, AP3 is selected by the top 16 KB of the page and AP0 is selected by the bottom 16 KB of the page. The selected AP bits are then interpreted in the same way as for a section (see Table 42: "Interpreting access permission (AP) bits" on page 98).

  The only difference is that the fault generated is a page permission fault.
- **Tiny page:** If the level one descriptor defines a page-mapped access and the level two descriptor is for a tiny page, the AP bits of the level one descriptor define whether the access is allowed in the same way as for a section. The fault generated is a page permission fault.

External aborts

In addition to MMU-generated aborts, external aborts cam be generated for certain types of access that involve transfers over the AHB bus. These aborts can be used to flag errors on external memory accesses. Not all accesses can be aborted in this way, however.

These accesses can be aborted externally:

- Page walks
- Noncached reads

- Nonbuffered writes
- Noncached read-lock-write (SWP) sequence

For a read-lock-write (SWP) sequence, the write is always attempted if the read externally aborts.

A swap to an NCB region is forced to have precisely the same behavior as a swap to an NCNB region. This means that the write part of a swap to an NCB region can be aborted externally.

Enabling the MMU

Before enabling the MMU using the R1: Control register (see page 58), you must perform these steps:

1. Program the R2: Translation Table Base register (see page 61) and the R3: Domain Access Control register (see page 61).
2. Program first-level and second-level page tables as required, ensuring that a valid translation table is placed in memory at the location specified by the Translation Table Base register.

When these steps have been performed, you can enable the MMU by setting R1: Control register bit 0 (the M bit) to high.

Care must be taken if the translated address differs from the untranslated address, because several instructions following the enabling of the MMU might have been prefetched with MMU off (VA=MVA=PA). If this happens, enabling the MMU can be considered as a branch with delayed execution. A similar situation occurs when the MMU is disabled. Consider this code sequence:

```
MRC p15, 0, R1, c1, C0, 0          ; Read control register
ORR R1, #0x1                       ; Set M bit
MCR p15, 0,R1,C1, C0,0             ; Write control register and enable MMU
Fetch Flat
Fetch Flat
Fetch Translated
```

Note: Because the same register (R1: Control register) controls the enabling of ICache, DCache, and the MMU, all three can be enabled using a single MCR instruction.

Disabling the MMU

Clear bit 0 (the M bit) in the R1: Control register to disable the MMU.

Note: If the MMU is enabled, then disabled, then subsequently re-enabled, the contents of the TLB are preserved. If these are now invalid, the TLB must be invalidated before re-enabling the MMU (see "R8:TLB Operations register" on page 68).

TLB structure

The MMU runs a single unified TLB used for both data accesses and instruction fetches. The TLB is divided into two parts:

- An eight-entry fully-associative part used exclusively for holding locked down TLB entries.
- A set-associative part for all other entries.

Whether an entry is placed in the set-associative part or lockdown part of the TLB depends on the state of the TLB Lockdown register when the entry is written into the TLB (see "R10: TLB Lockdown register" on page 73).

When an entry has been written into the lockdown part of the TLB, it can be removed only by being overwritten explicitly or, when the MVA matches the locked down entry, by an MVA-based TLB invalidate operation.

The structure of the set-associative part of the TLB does not form part of the programmer's model for the ARM926EJ-S processor. No assumptions must be made about the structure, replacement algorithm, or persistence of entries in the set-associative part — specifically:

- Any entry written into the set-associative part of the TLB can be removed at any time. The set-associative part of the TLB must be considered as a temporary cache of translation/page table information. No reliance must be placed on an entry residing or not residing in the set-associative TLB unless that entry already exists in the lockdown TLB. The set-associative part of the TLB can contain entries that are defined in the page tables but do not correspond to address values that have been accessed since the TLB was invalidated.
- The set-associative part of the TLB must be considered as a cache of the underlying page table, where memory coherency must be maintained at all

times. To guarantee coherency if a level one descriptor is modified in main memory, either an invalidate-TLB or Invalidate-TLB-by-entry operation must be used to remove any cached copies of the level one descriptor. This is required regardless of the type of level one descriptor (section, level two page reference, or fault).

- If any of the subpage permissions for a given page are different, each of the subpages are treated separately. To invalidate all entries associated with a page with subpage permissions, four MVA-based invalidate operations are required — one for each subpage.

Caches and write buffer

The ARM926EJ-S processor includes an instruction cache (ICache), data cache (DCache), and write buffer. The size of the caches can be from 4 KB to 128 KB, in power of two increments.

Cache features

- The caches are virtual index, virtual tag, addressed using the modified virtual address (MVA). This avoids cache cleaning and/or invalidating on context switch.
- The caches are four-way set associative, with a cache line length of eight words per line (32 bytes per line), and with two dirty bits in the DCache.
- The DCache supports write-through and write-back (copyback) cache operations, selected by memory region using the C and B bits in the MMU translation tables.
- The caches support *allocate on read-miss*. The caches perform critical-word first cache refilling.
- The caches use pseudo-random or round-robin replacement, selected by the RR bit in R1: Control register.
- Cache lockdown registers enable control over which cache ways are used for allocation on a linefill, providing a mechanism for both lockdown and controlling cache pollution.

- The DCache stores the *Physical Address Tag* (PA tag) corresponding to each DCache entry in the tag RAM for use during cache line write-backs, in addition to the virtual address tag stored in the tag RAM. This means that the MMU is not involved in DCache write-back operations, which removes the possibility of TLB misses to the write-back address.
- Cache maintenance operations provide efficient invalidation of:
  - The entire DCache or ICache
  - Regions of the DCache or ICache
  - Regions of virtual memory

  Cache maintenance operations also provide for efficient cleaning and invalidation of:
  - The entire DCache
  - Regions of the DCache
  - Regions of virtual memory

  The latter allows DCache coherency to be efficiently maintained when small code changes occur; for example, for self-modifying code and changes to exception vectors.

Write buffer

The write buffer is used for all writes to a noncachable bufferable region, write-through region, and write misses to a write-back region. A separate buffer is incorporated in the DCache for holding write-back data for cache line evictions or cleaning of dirty cache lines.

- The main write buffer has a 16-word data buffer and a four-address buffer.
- The DCache write-back buffer has eight data word entries and a single address entry.

The MCR drain write buffer instruction enables both write buffers to be drained under software control.

The MCR wait -for-interrupt causes both write buffers to be drained, and the ARM926EJ-S processor to be put into low-power state until an interrupt occurs.

Enabling the caches

On reset, the ICache and DCache entries all are invalidated and the caches disabled. The caches are not accessed for reads or writes. The caches are enabled using the I, C, and M bits from the R1: Control register, and can be enabled independently of one another. Table 43 gives the I and M bit settings for the ICache, and the associated behavior.

R1 I bit	R1 M bit	ARM926EJ-S behavior
0	-----	ICache disabled. All instruction fetches are fetched from external memory (AHB).
1	0	ICache enabled, MMU disabled. All instruction fetches are cachable, with no protection checks. All addresses are flat-mapped; that is, VA=MVA=PA.
1	1	ICache enabled, MMU enabled. Instruction fetches are cachable or noncachable, depending on the page descriptor C bit (see Table 44: "Page table C bit settings for ICache"), and protection checks are performed. All addresses are remapped from VA to PA, depending on the page entry; that is, the VA is translated to MVA and the MVA is remapped to a PA.

Table 43: R1:Control register I and M bit settings for ICache

Table 44 shows the page table C bit settings for the ICache (R1 I bit = M bit = 1).

Page table C bit	Description	ARM926EJ-S behavior	
0	Noncachable	ICache disabled. All instruction fetches are fetched from external memory.	
1	Cachable	**Cache hit**	Read from the ICache.
		Cache miss	Linefill from external memory.

Table 44: Page table C bit settings for ICache

Table 45 gives the R1: Control register C and M bit settings for DCache, and the associated behavior.

R1 C bit	R1 M bit	ARM926EJ-S behavior
0	0	DCache disabled. All data accesses are to the external memory.

Table 45: R1: Control register I and M bit settings for DCache

R1 C bit	R1 M bit	ARM926EJ-S behavior
1	0	DCache enabled, MMU disabled. All data accesses are noncachable, nonbufferable, with no protection checks. All addresses are flat-mapped; that is, VA=MVA=PA.
1	1	DCache enabled, MMU enabled. All data accesses are cachable or noncachable, depending on the page descriptor C bit and B bit (see Table 46: "Page table C and B bit settings for DCache"), and protection checks are performed. All addresses are remapped from VA to PA, depending on the MMU page table entry; that is, the VA is translated to an MVA and the MVA is remapped to a PA.

Table 45: R1: Control register I and M bit settings for DCache

Table 46 gives the page table C and B bit settings for the DCache (R1: Control register C bit = M bit = 1), and the associated behavior.

Page table C bit	Page table B bit	Description	ARM926EJ-S behavior	
0	0	Noncachable, nonbufferable	DCache disabled. Read from external memory. Write as a nonbuffered store(s) to external memory. DCache is not updated.	
0	1	Noncachable, bufferable	DCache disabled. Read from external memory. Write as a buffered store(s) to external memory. DCache is not updated.	
1	0	Write-through	DCache enabled:	
			Read hit	Read from DCache.
			Read miss	Linefill.
			Write hit	Write to the DCache, and buffered store to external memory.
			Write miss	Buffered store to external memory.
1	1	Write-back	DCache enabled:	
			Read hit	Read from DCache.
			Read miss	Linefill.
			Write hit	Write to the DCache only.
			Write miss	Buffered store to external memory.

Table 46: Page table C and B bit settings for DCache

Cache MVA and Set/Way formats

This section shows how the MVA and set/way formats of ARM926EJ-S caches map to a generic virtually indexed, virtually addressed cache. Figure 37 shows a generic, virtually indexed, virtually addressed cache.

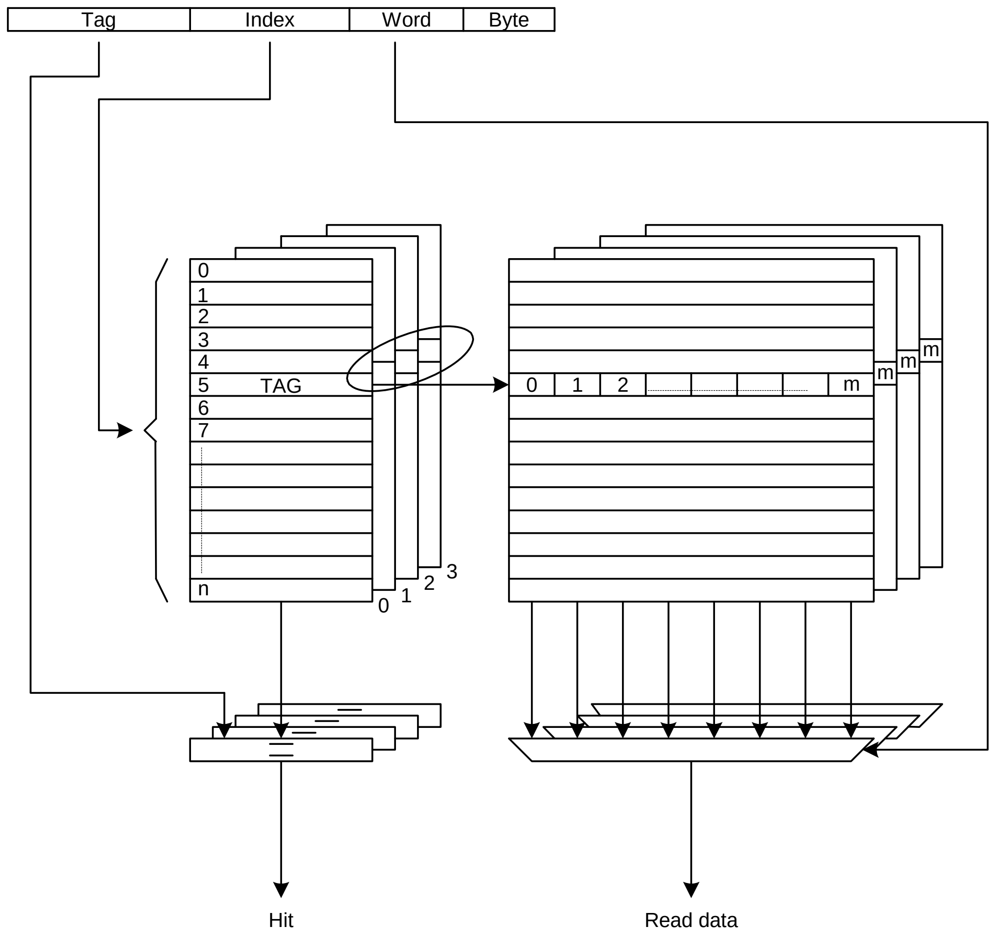


Figure 37: Generic virtually indexed, virtually addressed cache

Figure 38 shows the ARM926EJ-S cache format.

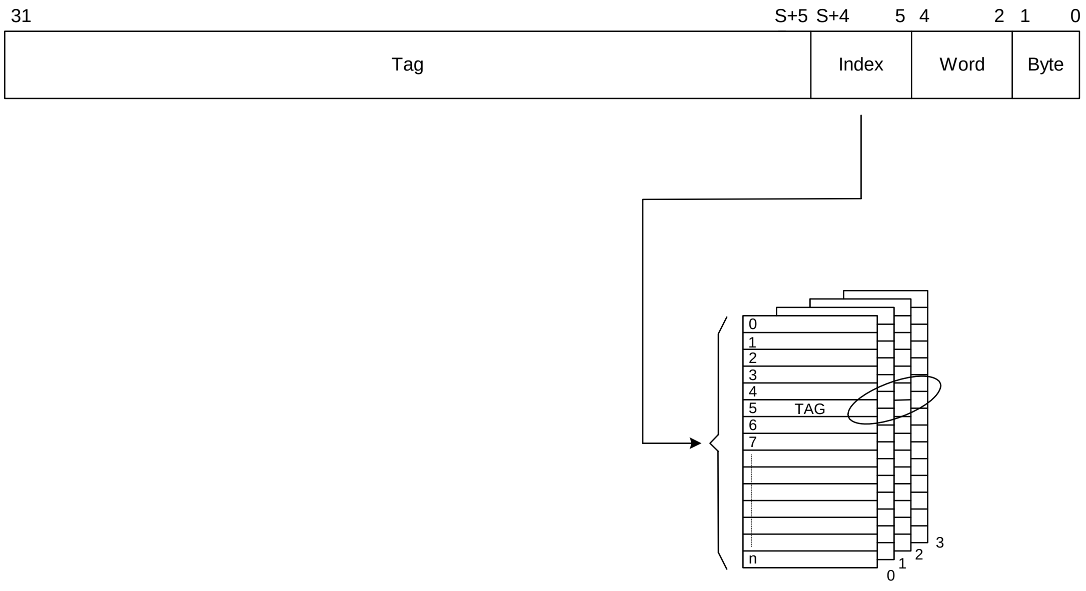


Figure 38: ARM926EJ-S cache associativity

The following points apply to the ARM926EJ-S cache associativity:

- The group of tags of the same index defines a *set.*
- The number of tags in a set is the *associativity*.
- The ARM926EJ-S caches are 4-way associative.
- The range of tags addressed by the index defines a *way.*
- The number of tags is a way is the number of sets, *NSETS*.

Table 47 shows values of S and NSETS for an ARM926EJ-S cache.

ARM926EJ-S	S	NSETS
4 KB	5	32
8 KB	6	64
16 KB	7	128

Table 47: Values of S and NSETS

ARM926EJ-S	S	NSETS
32 KB	8	256
64 KB	9	512
128 KB	10	1024

Table 47: Values of S and NSETS

Figure 39 shows the set/way/word format for ARM926EJ-S caches.

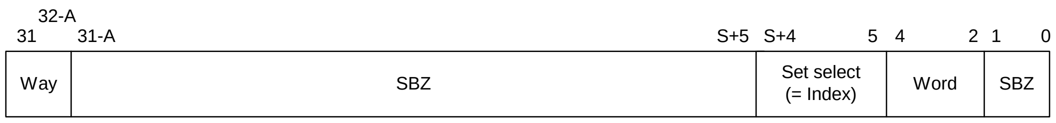


Figure 39: ARM926EJ-S cache set/way/word format

In this figure:

$A = \log_2$ associativity

For example, with a 4-way cache A = 2:

$S = \log_2$ NSETS

Noncachable instruction fetches

The ARM926EJ-S processor performs speculative noncachable instruction fetches to increase performance. Speculative instruction fetching is enabled at reset.

Note: It is recommended that you use ICache rather than noncachable code, when possible. Noncachable code previously has been used for operating system boot loaders and for preventing cache pollution. ICache, however, can be enabled without the MMU being enabled, and cache pollution can be controlled using the cache lockdown register.

Self-modifying code

A four-word buffer holds speculatively fetched instructions. Only sequential instructions are fetched speculatively; if the ARM926EJ-S issues a nonsequential instruction fetch, the contents of the buffer are discarded (flushed). In situations on which the contents of the prefetch buffer might become invalid during a sequence of sequential instruction fetches by the processor core (for example, turning the MMU on or off, or turning on the ICache), the prefetch buffer also is flushed. This avoids the necessity of performing an explicit *Instruction Memory Barrier* (IMB) operation, except when self-modifying code is used. Because the prefetch buffer is flushed when the ARM926EJ-S core issues a nonsequential instruction fetch, a branch instruction (or equivalent) can be used to implement the required IMB behavior, as shown in this code sequence:

```
LDMIA       R0,{R1-R5}              ; load code sequence into R1-R5
ADR         R0,self_mod_code
STMIA       R0,{R1-R5}              ; store code sequence (nonbuffered region)
B           self_mod_code           ; branch to modified code
self_mod_code:
```

This IMB application applies only to the ARM926EJ-S processor running code from a noncachable region of memory. If code is run from a cachable region of memory, or a different device is used, a different IMB implementation is required. IMBs are discussed in "Instruction Memory Barrier," beginning on page 113.

AHB behavior

If instruction prefetching is disabled, all instruction fetches appear on the AHB interface as single, nonsequential fetches.

If prefetching is enabled, instruction fetches appear either as bursts of four instructions or as single, nonsequential fetches. No speculative instruction fetching is done across a 1 KB boundary.

All instruction fetches, including those made in Thumb state, are word transfers (32 bits). In Thumb state, a single-word instruction fetch reads two Thumb instructions and a four-word burst reads eight instructions.

Instruction Memory Barrier

Whenever code is treated as data — for example, self-modifying code or loading code into memory — a sequence of instructions called an *instruction memory barrier (IMB)* operation must be used to ensure consistency between the data and instruction streams processed by the ARM926EJ-S processor.

Usually the instruction and data streams are considered to be completely independent by the ARM926EJ-S processor memory system, and any changes in the data side are not automatically reflected in the instruction side. For example, if code is modified in main memory, ICache may contain stale entries. To remove these stale entries, part of all of the ICache must be invalidated.

IMB operation

Use this procedure to ensure consistency between data and instruction sides:

1 **Clean the DCache.** If the cache contains cache lines corresponding to write-back regions of memory, it might contain dirty entries. These entries must be cleaned to make external memory consistent with the DCache. If only a small part of the cache has to be cleaned, it can be done by using a sequence of clean DCache single entry instructions. If the entire cache has to be cleaned, you can use the test and clean operation (see "R7: Cache Operations register," beginning on page 64).

2 **Drain the write buffer.** Executing a drain write buffer causes the ARM926EJ-S core to wait until outstanding buffered writes have completed on the AHB interface. This includes writes that occur as a result of data being written back to main memory because of clean operations, and data for store instructions.

3 **Synchronize data and instruction streams in level two AHB systems.** The level two AHB subsystem might require synchronization between data and instruction sides. It is possible for the data and instruction AHB masters to be attached to different AHB subsystems. Even if both masters are present on the same bus, some form of separate ICache might exist for performance reasons; this must be invalidated to ensure consistency.

The process of synchronizing instructions and data in level two memory must be invoked using some form of fully blocking operation, to ensure that the end of the operation can be determined using software. It is recommended that either a nonbuffered store (STR) or a noncached load (LDR) be used to trigger external synchronization.

4 **Invalidate the cache.** The ICache must be invalidated to remove any stale copies of instructions that are no longer valid. If the ICache is not being used, or the modified regions are not in cachable areas of memory, this step might not be required.

5 **Flush the prefetch buffer.** To ensure consistency, the prefetch buffer should be flushed before self-modifying code is executed (see "Self-modifying code" on page 112).

Sample IMB sequences

These sequences correspond to steps 1–4 in "IMB operation."

```
clean loop
        MRC p15, 0, r15, c7, c10, 3         ; clean entire dcache using test and clean
        BNE clean_loop

        MRC p15, 0, r0, c7, c10, 4          ; drain write buffer
        STR rx,[ry]                                ; nonbuffered store to signal L2 world to
                                                   ; synchronize
        MCR p15, 0, r0, c7, c5, 0                  ; invalidate icache
```

This next sequence illustrates an IMB sequence used after modifying a single instruction (for example, setting a software breakpoint), with no external synchronization required:

```
STR rx,[ry]                         ; store that modifies instruction at address ry
MCR p15, 0, ry, c7, c10, 1          ; clean dcache single entry (MVA)
MCR p15, 0, r0, c7, c10, 4          ; drain write buffer
MCR p15, 0, ry, c7, c5, 1           ; invalidate icache single entry (MVA)
```

Memory Controller

C H A P T E R 4

The Multiport Memory Controller is an AMBA-compliant system-on-chip (SoC) peripheral that connects to the Advanced High-performance Bus (AHB). The remainder of this chapter refers to this controller as the *memory controller*.

Features

The memory controller provides these features:

- AMBA 32-bit AHB compliancy.
- Dynamic memory interface support including SDRAM and JEDEC low-power SDRAM.
- Asynchronous static memory device support including RAM, ROM, and Flash, with and without asynchronous page mode.
- Can operate with cached processors with copyback caches.
- Can operate with uncached processors.
- Low transaction latency.
- Read and write buffers to reduce latency and improve performance, particularly for uncached processors.
- 8-bit, 16-bit, and 32-bit wide static memory support.
- 16-bit and 32-bit wide chip select SDRAM memory support.
- Static memory features, such as:
  - Asynchronous page mode read
  - Programmable wait states
  - Bus turnaround delay
  - Output enable and write enable delays
  - Extended wait
- Power-saving modes that dynamically control SDRAM clk_en.
- Dynamic memory self-refresh mode supported by a power management unit (PMU) interface or by software.
- Controller supports 2K, 4K, and 8K row address synchronous memory parts; that is, typical 512 MB, 256 MB, and 16 Mb parts with 8, 16, or 32 DQ bits per device.
- A separate AHB interface to program the memory controller. This enables the memory controller registers to be situated in memory with other system peripheral registers.
- Locked AHB transaction support.

- Support for all AHB burst types.
- Little and big endian support.

Note: Synchronous static memory devices (synchronous burst mode) are not supported.

System overview

Figure 40 shows the NS9750 memory controller in a sample system.

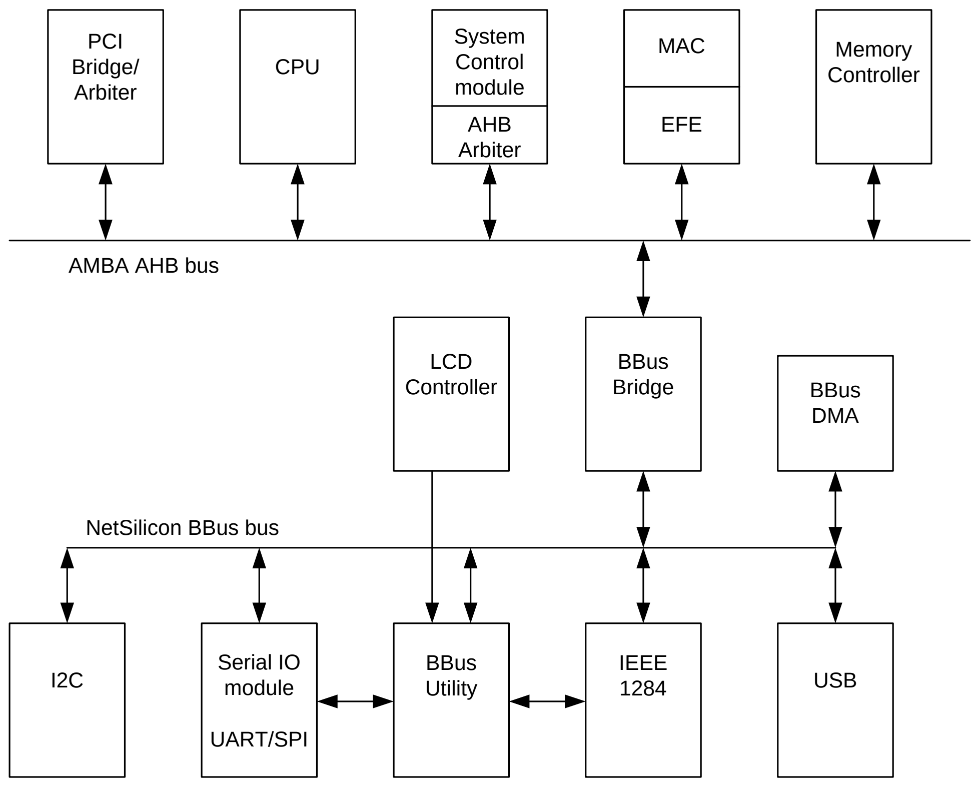


Figure 40: NS9750 sample system

Note: The largest amount of memory allowed for a single chip select is 256 MB.

Low-power operation

In many systems, the contents of the memory system have to be maintained during low-power sleep modes. NS9750 provides two features to enable this:

- Dynamic memory refresh over soft reset
- A mechanism to place the dynamic memories into self-refresh mode

Self-refresh mode can be entered as follows:

1. Set the SREFREQ bit in the Dynamic Memory Control register (see page 208).
2. Poll the SREFACK bit in the Status register (see page 207).

Note: Static memory can be accessed as normal when the SDRAM memory is in self-refresh mode.

Low-power SDRAM partial array refresh

The memory controller supports JEDEC low-power SDRAM partial array refresh. Partial array refresh can be programmed by initializing the SDRAM memory device appropriately. When the memory device is put into self-refresh mode, only the memory banks specified are refreshed. The memory banks that are not refreshed lose their data contents.

Memory map

The memory controller provides hardware support for booting from external nonvolatile memory. During booting, the nonvolatile memory must be located at address 0x00000000 in memory. When the system is booted, the SRAM or SDRAM memory can be remapped to address 0x00000000 by modifying the address map in the AHB decoder.

Power-on reset memory map

On power-on reset, memory chip select 1 is mirrored onto memory chip select 0 and chip select 4. Any transactions to memory chip select 0 or chip select 4 (or chip select 1), then, access memory chip select 1. Clearing the address mirror bit (M) in the Control register (see page 205) disables address mirroring, and memory chip select 0, chip select 4, and memory chip select 1 can be accessed as normal.

Chip select 1 memory configuration

You can configure the memory width and chip select polarity of static memory chip select 1 by using selected input signals. This allows you to boot from chip select 1.

These are the bootstrap signals:

- boot_strap[4:3]: Memory width select
- gpio[49]: Chip select polarity
- boot_strap[0]: Byte lane enable_n/write_enable_n for byte-wide devices

Example: Boot from flash, SRAM mapped after boot

The system is set up as:

- Chip select 1 is connected to the boot flash device.
- Chip select 0 is connected to the SRAM to be remapped to 0x00000000 after boot.

The boot sequence is as follows:

1. At power-on, the reset chip select 1 is mirrored into chip select 0 (and chip select 4). The following signals are configured so the nonvolatile memory device can be accessed:
   - boot_strap[4:3]
   - gpio[49]
2. When the power-on reset (reset_n) and AHB reset (HRESETn) go inactive, the processor starts booting from 0x00000000 in memory.
3. The software programs the optimum delay values in the flash memory so the boot code can run at full speed.
4. The code branches to chip select 1 so the code can continue executing from the non-remapped memory location.
5. The appropriate values are programmed into the memory controller to configure chip select 0.
6. The address mirroring is disabled by clearing the address mirror (M) field in the Control register (see page 205).
7. The ARM reset and interrupt vectors are copied from flash memory to SRAM that can then be accessed at address 0x00000000.

8 More boot, initialization, or application code is executed.

Example: Boot from flash, SDRAM remapped after boot

The system is set up as:

- Chip select 1 is connected to the boot flash device.
- Chip select 4 is connected to the SDRAM to be remapped to 0x00000000 after boot.

The boot sequence is as follows:

1 At power-on, the reset chip select 1 is mirrored into chip select 4 (and chip select 0). The following signals are configured so the nonvolatile memory device can be accessed:
   - boot_strap[4:3]
   - gpio[49]

2 When the power-on reset (reset_n) and AHB reset (HRESETn) go inactive, the processor starts booting from 0x00000000 in memory.

3 The software programs the optimum delay values in flash memory so the boot code can run at full speed.

4 The code branches to chip select 1 so the code can continue executing from the non-remapped memory location.

5 The appropriate values are programmed into the memory controller to configure chip select 4, and the memory device is initialized.

6 The address mirroring is disabled by clearing the address mirror (M) field in the Control register (see page 205).

7 The ARM reset and interrupt vectors are copied from flash memory to SDRAM that can then be accessed at address 0x00000000.

8 More boot, initialization, or application code is executed.

Static memory controller

Table 48 shows configurations for the static memory controller with different types of memory devices. See "Static Memory Configuration 0–3 registers" on page 230 for more information.

Device	Write protect	Page mode	Buffer
ROM	Enabled	Disabled	Disabled [a]
Page mode ROM	Enabled	Enabled	Enabled [a]
Extended wait ROM	Enabled	Disabled	Disabled [a]
SRAM	Disabled (or enabled) [b]	Disabled	Disabled [a]
Page mode SRAM	Disabled (or enabled) [b]	Enabled	Enabled [a]
Extended wait SRAM	Disabled (or enabled) [b]	Disabled	Disabled [a]
Flash	Disabled or (enabled) [b]	Disabled	Disabled [c]
Page mode flash	Disabled or (enabled) [b]	Enabled	Enabled [c]
Extended wait flash	Disabled or (enabled) [b]	Disabled	Disabled [a]
Memory mapped peripheral	Disabled (or enabled) [b]	Disabled	Disabled

a Enabling the buffers means that any access causes the buffer to be used. Depending on the application, this can provide performance improvements. Devices without async-page-mode support generally work better with the buffer disabled. Again, depending on the application, this can provide performance improvements.

b SRAM and Flash memory devices can be write-protected if required.

c Buffering must be disabled when performing Flash memory commands and during writes.

Table 48: Static memory controller configurations

Notes:

- Buffering enables the transaction order to be rearranged to improve memory performance. If the transaction order is important, the buffers must be disabled.
- Extended wait and page mode cannot be enabled at the same time.

Write protection

Each static memory chip select can be configured for write-protection. SRAM usually is unprotected and ROM devices must be write-protected (to avoid potential bus conflict when performing a write access to ROM), but the P field in the Static Memory Configuration register (see "Static Memory Configuration 0–3 registers" on page 230) can be set to write-protect SRAM as well as ROM devices. If a write access is made to a write-protected memory bank, an error is indicated by the HRESP[1:0] signal. If a write access is made to a memory bank containing ROM devices and the chip select is not write-protected. An error is not returned and the write access proceeds as normal. Note that this might lead to a bus conflict.

Extended wait transfers

The static memory controller supports extremely long transfer times. In normal use, the memory transfers are timed using the Static Memory Read Delay register (StaticWaitRd, see "Static Memory Read Delay 0–3 registers" on page 236) and Static Memory Wait Delay register (StaticWaitWr, see "Static Memory Write Delay 0–3 registers" on page 238). These registers allow transfers with up to 32 wait states. If a very slow static memory device has to be accessed, however, you can enable the static configuration extended wait (EW) bit. When EW is enabled, the Static Extended Wait register ("Static Memory Extended Wait register" on page 224) is used to time both the read and write transfers. The Static Extended Wait register allows transfers to have up to 16368 wait states.

Notes:

- Using extremely long transfer times might mean that SDRAM devices are not refreshed correctly.
- Very slow transfers can degrade system performance, as the external memory interface is tied up for long periods of time. This has detrimental effects on time critical services, such as interrupt latency and low latency devices; for example, video controllers.

Memory mapped peripherals

Some systems use external peripherals that can be accessed using the static memory interface. Because of the way many of these peripherals function, the read and write transfers to them must not be buffered. The buffer must therefore be disabled.

Static memory initialization

Static memory must be initialized as required after poweron reset (reset_n) by programming the relevant registers in the memory controller as well as the configuration registers in the external static memory device.

Access sequencing and memory width

The data width of each external memory bank must be configured by programming the appropriate bank configuration register (Static Memory Configuration 0–3). When the external memory bus is narrower that the transfer initiated from the current main bus master, the internal bus transfer takes several external bus transfers to complete.

For example, if bank 0 is configured as 8-bit wide memory and a 32-bit read is initiated, the AHB bus stalls while the memory controller reads four consecutive bytes from the memory. During these accesses, the static memory controller block demultiplexes the four bytes into one 32-bit word on the AHB bus.

Wait state generation

Each bank of the memory controller must be configured for external transfer wait states in read and write accesses. Configure the banks by programming the appropriate bank control registers:

- "Static Memory Configuration 0–3 registers" on page 230 (StaticConfig[n])
- "Static Memory Write Enable Delay 0–3 registers" on page 234 (StaticWaitWen[n])
- "Static Memory Output Enable Delay 0–3 registers" on page 235 (StaticWaitOen[n])
- "Static Memory Read Delay 0–3 registers" on page 236 (StaticWaitRd[n])
- "Static Memory Write Delay 0–3 registers" on page 238 (StaticWaitWr[n])

- "Static Memory Page Mode Read Delay 0–3 registers" on page 237 (StaticWaitPage[n])
- "Static Memory Turn Round Delay 0–3 registers" on page 239 (StaticWaitTurn[n])
- "Static Memory Extended Wait register" on page 224 (StaticExtendedWait)

The number of cycles in which an AMBA transfer completes is controlled by two additional factors:

- Access width
- External memory width

Each bank of the memory controller has a programmable enable for the extended wait (EW). The WAITRD wait state field in the Static Memory Read Delay register can be programmed to select from 1–32 wait states for read memory accesses to SRAM and ROM, or the initial read access to page mode devices. The WAITWR wait state field in the Static Memory Write Delay register can be programmed to select from 1–32 wait states for access to SRAM. The Static Memory Page Mode Read Delay register can be programmed to select from 1–32 wait states for page mode accesses.

Static memory read control

There are three types of static memory read controls:

- Output enable programmable delay
- ROM, SRAM, and flash
- Asynchronous page mode read

Output enable programmable delay

The delay between the assertion of the chip select and the output enable is programmable from 0 to 15 cycles using the wait output enable bits (WAITOEN) in the Static Memory Output Enable Delay registers (see "Static Memory Output Enable Delay 0–3 registers" on page 235). The delay is used to reduce power consumption for memories that cannot provide valid output data immediately after the chip select has been asserted. The output enable is always deasserted at the same time as the chip select. at the end of the transfer.

ROM, SRAM, and Flash

The memory controller uses the same read timing control for ROM, SRAM, and flash devices. Each read starts with the assertion of the appropriate memory bank chip select signals (STCSOUT_n) and memory address (ADDROUT[27:0]). The read access time is determined by the number of wait states programmed for the WAITRD field in the Static Memory Read Delay register. The WAITTURN field in the Static Memory Turn round Delay register determines the number of bus turnaround wait states added between external read and write transfers.

Figure 41 shows an external memory read transfer with the minimum zero wait states (WAITRD=0). Cycles T0 through T4 are internal AHB bus cycles. These cycles are required to arbitrate for control of the AHB bus. Maximum performance is achieved when accessing the external device with load multiple (LDM) or store multiple (STM) CPU instructions. Table 49 provides the timing parameters. Table 50 describes the transactions in Figure 41.

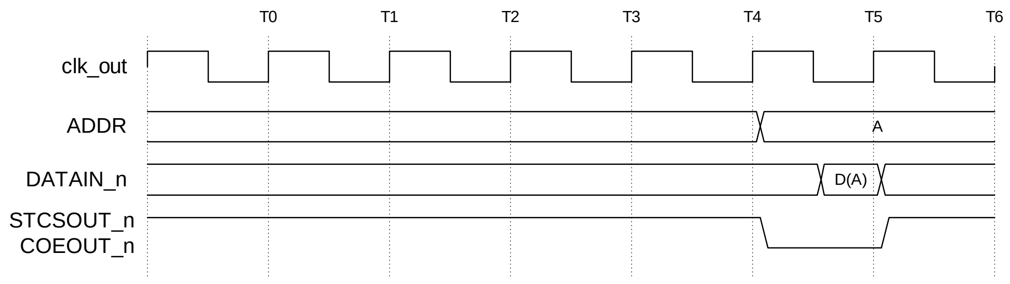


Figure 41: External memory 0 wait state read timing diagram

Timing parameter	Value
WAITRD	0
WAITOEN	0
WAITPAGE	N/A
WAITWR	N/A
WAITWEN	N/A
WAITTURN	N/A

Table 49: Static memory timing parameters

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.
T4-T5	Static memory address, chip select, and control signals submitted to static memory.
T5-T6	Read data returned from the static memory. Data is provided to AHB.

Table 50: External memory 0 wait state read

Figure 42 shows an external memory read transfer with two wait states (WAITRD=2). Seven AHB cycles are required for the transfer, five for the standard read access and an additional two because of the programmed wait states added (WAITRD). Table 51 provides the timing parameters. Table 52 describes the transactions in Figure 42.

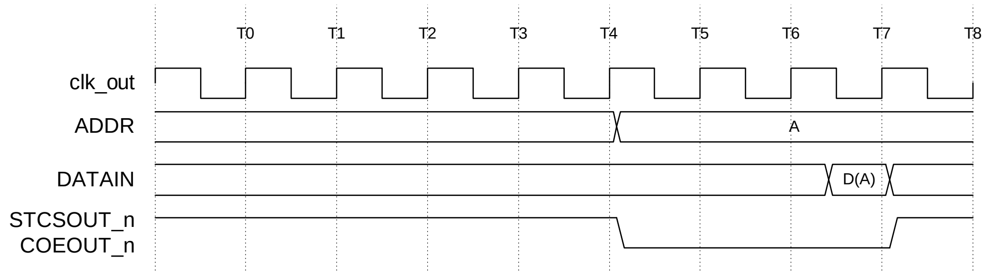


Figure 42: External memory 2 wait state read timing diagram

Timing parameter	Value
WAITRD	2
WAITOEN	0
WAITPAGE	N/A
WAITWR	N/A
WAITEN	N/A
WAITTURN	N/A

Table 51: Static memory timing parameters

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.
T4-T5	Static memory address, chip select, and control signals submitted to static memory.
T5-T6	Read wait state 1.
T6-T7	Read wait state 2.
T7-T8	Read data returned from the static memory. Data is provided to the AHB.

Table 52: External memory 2 wait state read

Figure 43 shows an external memory read transfer with two output enable delay states (WAITOEN=2). Seven AHB cycles are required for the transfer, five for the standard read and an additional two because of the output delay states added. Table 53 provides the timing parameters. Table 54 describes the transactions for Figure 43.

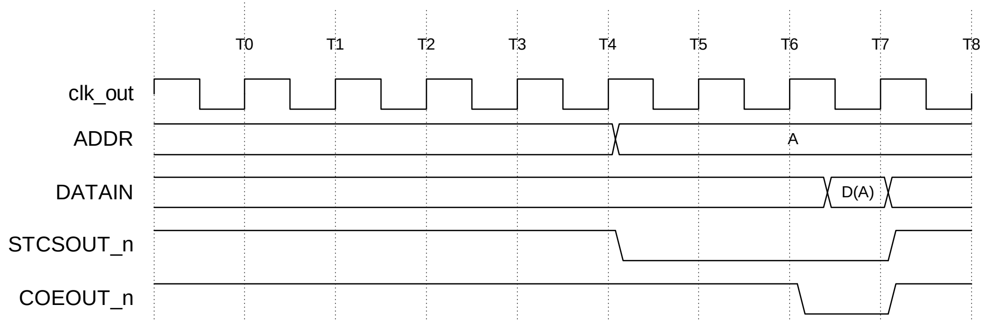


Figure 43: External memory 2 output enable delay state read timing diagram

Timing parameter	Value
WAITRD	2
WAITOEN	2
WAITPAGE	N/A
WAITWR	N/A
WAITWEN	N/A
WAITTURN	N/A

Table 53: Static memory timing parameters

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.
T4-T5	Static memory address, chip select, and control signals submitted to static memory. Static memory output enable inactive.
T5-T6	Static memory output enable inactive.
T6-T7	Static memory output enable active.
T7-T8	Read data returned from static memory. Data is provided to the AHB.

Table 54: External memory 2 output enable delay state

Figure 44 shows external memory read transfers with zero wait states (WAITRD=0). These transfers can be non-sequential transfers or sequential transfers of a specified burst length. Bursts of unspecified length are interpreted as INCR4 transfers. All transfers are treated as separate reads, so have the minimum of five AHB cycles added.

Table 55 provides the timing parameters. Table 56 describes the transactions for Figure 44.

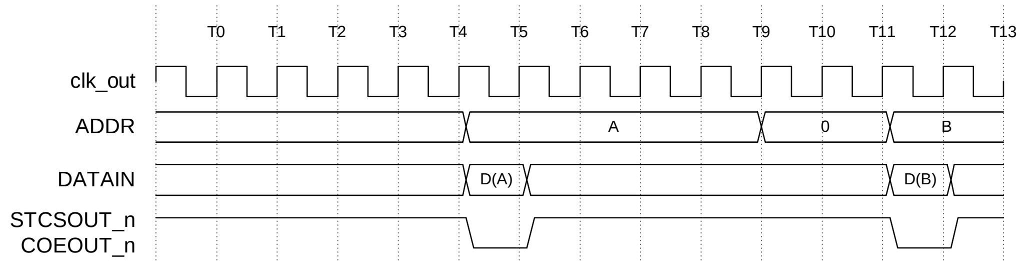


Figure 44: External memory 2 0 wait state read timing diagram

Timing parameter	Value
WAITRD	0
WAITOEN	0
WAITPAGE	N/A
WAITWR	N/A
WAITWEN	N/A
WAITTURN	N/A

Table 55: Static memory timing parameters

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.
T4-T5	Static memory address, chip select, and control signals submitted to static memory.
T5-T6	Read data returned to static memory. Data is provided to the AHB.
T6-T7	AHB address provided to memory controller. AHB transaction processing.

Table 56: External memory 2 0wait state reads

Cycle	Description
T7	AHB address provided to memory controller.
T7-T8	AHB transaction processing.
T8-T11	Arbitration of AHB memory ports.
T11-T12	Static memory address, chip select, and control signals submitted to static memory.
T12-T13	Read data returned from static memory. Data is provided to the AHB.

Table 56: External memory 2 0wait state reads

Figure 45 shows a burst of zero wait state reads with the length specified. Because the length of the burst is known, the chip select can be held asserted during the whole burst and generate the external transfers before the current AHB transfer has completed. The first read requires five arbitration cycles; the three subsequent sequential reads have zero AHB arbitration cycles added because the external transfers are automatically generated. Table 57 provides the timing parameters. Table 58 describes the transactions for Figure 45.

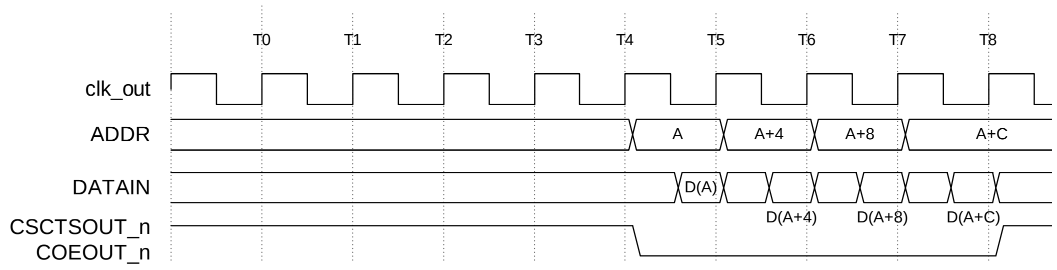


Figure 45: External memory 0 wait fixed length burst read timing diagram

Timing parameter	Value
WAITRD	0
WAITOEN	0
WAITPAGE	N/A
WAITWR	N/A

Table 57: SRAM timing parameters

Timing parameter	Value
WAITWEN	N/A
WAITTURN	N/A

Table 57: SRAM timing parameters

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.
T4-T5	Static memory read 0 address, chip select, and control signals submitted to static memory.
T5-T6	Static memory read 1 address, chip select, and control signals submitted to static memory. Read data 0 returned from static memory. Read data 0 is provided to the AHB.
T6-T7	Static memory read 2 address, chip select, and control signals submitted to static memory. read data 1 returned from the static memory. Read data 1 is provided to the AHB.
T7-T8	Static emory read 3 address, chip select, and control signals submitted to static memory. Read data 2 returned from the static memory. Read data 2 is provided to the AHB.
T8-T9	Read data 3 returned from the static memory. Read data 3 is provided to the AHB.

Table 58: External memory zero wait fixed length burst read

Figure 46 shows a burst of two wait state reads with the length specified. The WAITRD value is used for all transfers in the burst. Table 59 provides the timing parameters. Table 60 describes the transactions for Figure 46.

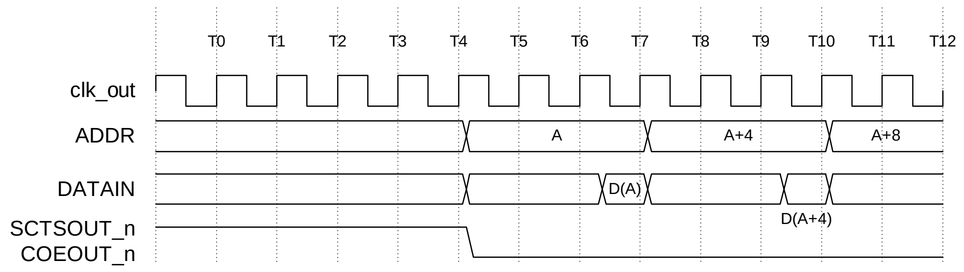


Figure 46: External memory 2 wait states fixed length burst read timing diagram

Timing parameter	Value
WAITRD	2
WAITOEN	0
WAITPAGE	N/A
WAITWR	N/A
WAITWEN	N/A
WAITTURN	N/A

Table 59: SRAM timing diagrams

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of memory ports.
T4-T5	Static memory address, chip select, and control signals submitted to static memory.
T5-T6	Read wait state 1.
T6-T7	Read wait state 2.

Table 60: External memory 2 wait states fixed length burst read

Cycle	Description
T7-T8	Read data 0 returned from the static memory. Read data 0 is provided to the AHB. Static memory transfer 1, address, chip select, and control signals submitted to static memory.
T8-T9	Read wait state 1.
T9-T10	Read wait state 2.
T10-T11	Read data 1 returned from the static memory. Read data 1 is provided to the AHB. Static memory transfer 2, address, chip select, and control signals submitted to static memory.
T11-T12	Read wait state 1.

Table 60: External memory 2 wait states fixed length burst read

Asynchronous page mode read

The memory controller supports asynchronous page mode read of up to four memory transfers by updating address bits A[1] and A[0]. This feature increases the bandwidth by using a reduced access time for the read accesses that are in page mode. The first read access takes static wait read and WAITRD cycles. Subsequent read accesses that are in page mode take static wait page and WAITPAGE cycles. The chip select and output enable lines are held during the burst, and only the lower two address bits change between subsequent accesses. At the end of the burst, the chip select and output enable lines are deasserted together.

Figure 47 shows an external memory page mode read transfer with two initial wait states and one sequential wait state. The first read requires five AHB arbitration cycles (plus three wait states); the following (up to 3) sequential transfers have only one AHB wait state. This gives increased performance over the equivalent nonpage mode ROM timing (see Figure 46, "External memory 2 wait states fixed length burst read timing diagram," on page 132). Table 61 provides the timing parameters. Table 62 describes the transactions for Figure 47.

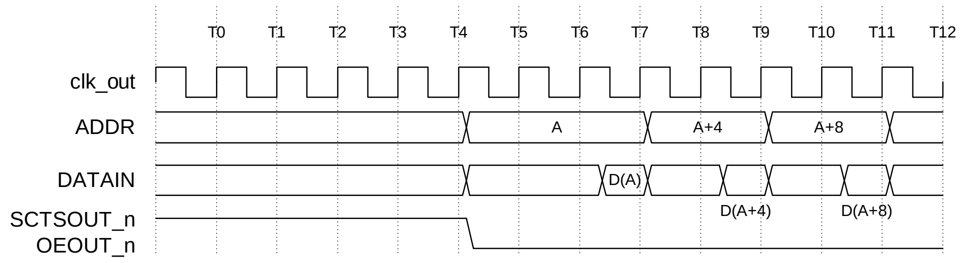


Figure 47: External memory page mode read transfer timing diagram

Timing parameter	Value
WAITRD	2
WAITOEN	0
WAITPAGE	1
WAITWR	N/A
WAITWEN	N/A
WAITTURN	N/A

Table 61: Static memory timing parameters

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.
T4-T5	Static memory transfer 0, address, chip select, and control signals submitted to static memory.
T5-T6	Read wait state 1.
T6-T7	Read wait state 2.
T7-T8	Read data 0 returned from static memory. Read data is provided to the AHB. Static memory transfer 1, address, chip select, and control signals submitted to static memory.

Table 62: External memory page mode read

Cycle	Description
T8-T9	Read page mode wait state 1.
T9-T10	Read data 1 returned from the static memory. Read data 1 is provided to the AHB. Static memory transfer 2, address, chip select, and control signals submitted to static memory.
T10-T11	Read page mode wait state 1.
T11-T12	Read data 2 returned from the static memory. Read data 2 is provided to the AHB. Static memory transfer 3, address, chip select, and control signals submitted to static memory.

Table 62: External memory page mode read

Figure 48 shows a 32-bit read from an 8-bit page mode ROM device, causing four burst reads to be performed. A total of eight AHB wait states are added during this transfer, five AHB arbitration cycles and then one for each of the subsequent reads. WAITRD and WAITPAGE are 0. Table 63 provides the timing parameters. Table 64 describes the transactions for Figure 48.

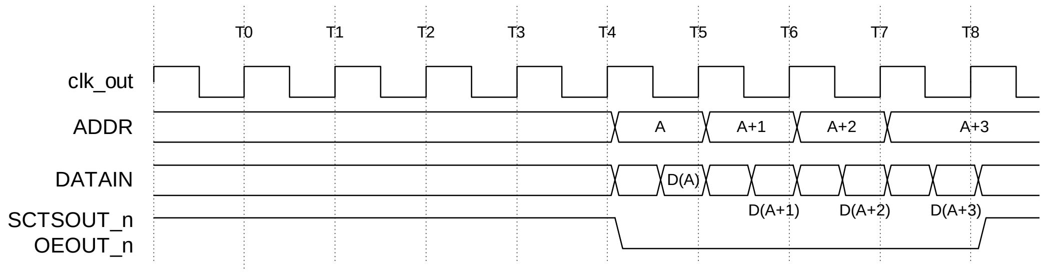


Figure 48: External memory 32-bit burst read from 8-bit memory timing diagram

Timing parameters	Value
WAITRD	0
WAITOEN	0
WAITPAGE	0

Table 63: Static memory timing parameters

Timing parameters	Value
WAITWR	N/A
WAITWEN	N/A
WAITTURN	N/A

Table 63: Static memory timing parameters

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.
T4-T5	Static memory transfer m0, address, chip select, and control signals submitted to static memory.
T5-T6	Static memory transfer 1, address, chip select, and control signals submitted to static memory. Read data byte 0 returned from static memory.
T6-T7	Static memory transfer 2, address, chip select, and control signals submitted to static memory. Read data byte 1 returned from the static memory.
T7-T8	Static memory transfer 3, address chip select, and control signals submitted to static memory. Read data byte 2 returned from the static memory.
T8-T9	Read data byte 3 returned from the static memory. Read data 32-bit word is provided to the AHB.

Table 64: External memory 32-bit burst read from 8-bit memory

Static memory write control

Write enable programming delay

The delay between the assertion of the chip select and the write enable is programmable from 1 to 16 cycles using the WAITWEN bits of the Static Memory Write Enable Delay (StaticWaitWen[3:0]) registers. The delay reduces the power consumption for memories. The write enable is asserted on the rising edge of HCLK after the assertion of the chip select for zero wait states. The write enable is always deasserted a cycle before the chip select, at the end of the transfer. BLSOUT_n (byte lane signal) has the same timing as WEOUT_n (write enable signal) for writes to 8-bit devices that use the byte lane selects instead of the write enables.

SRAM

Write timing for SRAM starts with assertion of the appropriate memory bank chip selects (STCSOUT[n]_n) and address signals (ADDROUT[27:0]_n). The write access time is determined by the number of wait states programmed for the WAITWR field in the Static Memory Write Delay register (see "Static Memory Write Delay 0–3 registers" on page 238). The WAITTURN field in the bank control register (see "Static Memory Turn Round Delay 0–3 registers" on page 239) determines the number of bus turnaround wait states added between external read and write transfers.

Figure 49 shows a single external memory write transfer with minimum zero wait states (WAITWR=0). One wait state is added. Table 65 provides the timing parameters. Table 66 describes the transactions for Figure 49.

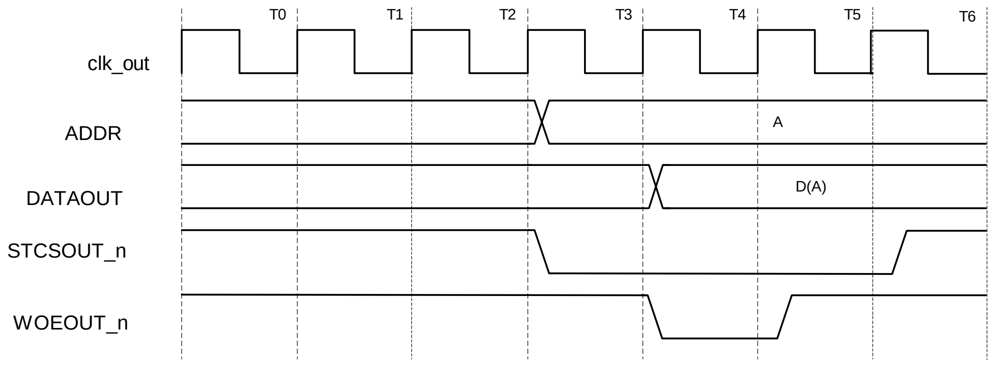


Figure 49: External memory 0 wait state write timing diagram

Timing parameters	Value
WAITRD	N/A
WAITOEN	N/A
WAITPAGE	N/A
WAITWR	0

Table 65: Static memory timing parameters

Timing parameters	Value
WAITWEN	0
WAITTURN	N/A

Table 65: Static memory timing parameters

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.
T4-T5	Static memory transfer 0, address, chip select, and control signals submitted to static memory. Write data is read from the AHB memory port. Write enable inactive.
T5-T6	Write enable taken active. Write data submitted to static memory. Static memory writes the data.
T6-T7	Static memory writes the data. Write enable taken inactive.
T7-T8	Static memory control signals taken inactive.

Table 66: External memory 0 wait state write

Figure 50 shows a single external memory write transfer with two wait states (WAITWR=2). One AHB wait state is added. Table 67 provides the timing parameters. Table 68 describes the transactions for Figure 50.

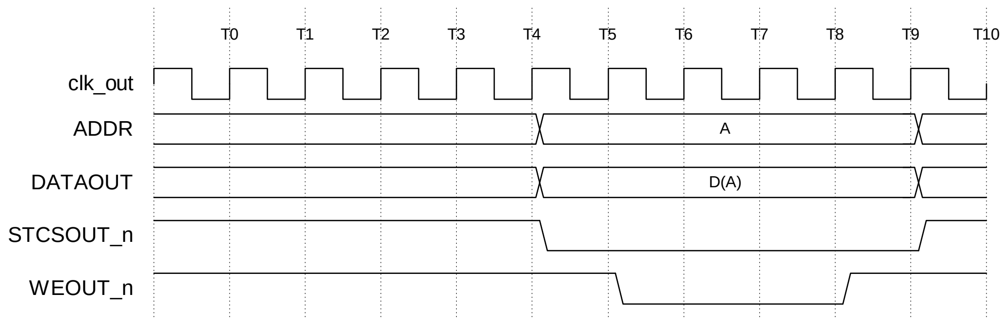


Figure 50: External memory 2 wait state write timing diagram

Timing parameter	**Value**
WAITRD	N/A
WAITOEN	N/A
WAITPAGE	N/A
WAITWR	2
WAITWEN	0
WAITTURN	N/A

Table 67: Static memory timing parameters

Cycle	**Description**
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.
T4-T5	Static memory transfer 0, address, chip select, and control signals submitted to static memory. Write data is read from the AHB memory port. Write enable inactive.
T5-T6	Write enable taken active. Write data submitted to static memory.

Table 68: External memory 2 wait state write

Cycle	Description
T6-T7	Wait state 1.
T7-T8	Wait state 2.
T8-T9	Static memory writes the data. Write enable taken inactive.
T9-T10	Static memory control signals taken inactive.

Table 68: External memory 2 wait state write

Figure 51 shows a single external memory write transfer with two write enable delay states (WAITWEN=2). One wait state is added. Table 69 provides the timing parameters.

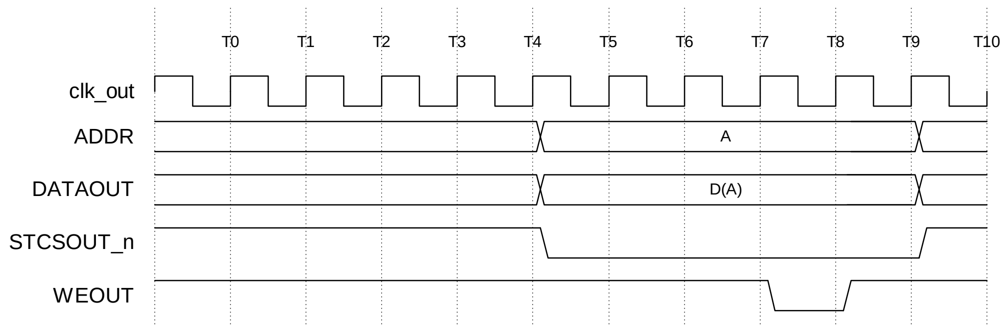


Figure 51: External memory 2 write enable delay write timing diagram

Timing parameters	Value
WAITRD	N/A
WAITOEN	N/A
WAITPAGE	N/A
WAITWR	2
WAITWEN	2
WAITTURN	N/A

Table 69: Static memory timing parameters

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.
T4-T5	Static memory transfer 0, address, chip select, and control signals submitted to static memory. Write data is read from the AHB memory port. Write enable active.
T5-T6	Write data submitted to static memory. Write enable wait state 1.
T6-T7	Write enable wait state 2.
T7-T8	Write enable taken active.
T8-T9	Static memory writes the data. Write enable taken inactive.
T9-T10	Static memory control signals taken inactive.

Table 70: External memory 2 write enable delay write

Figure 52 shows two external memory write transfers with zero wait states (WAITWR=0). Four AHB wait states are added to the second write, because this write can be started only when the first write has completed. This is the timing of any sequence of write transfers, nonsequential to nonsequential or nonsequential to sequential, with any value of HBURST. The maximum speed of write transfers is controlled by the external timing of the write enable relative to the chip select, so all external writes must take two cycles to complete: the cycle in which write enable is asserted and the cycle in which write enable is deasserted. Table 71 provides the timing parameters. Table 72 describes the transactions for Figure 52.

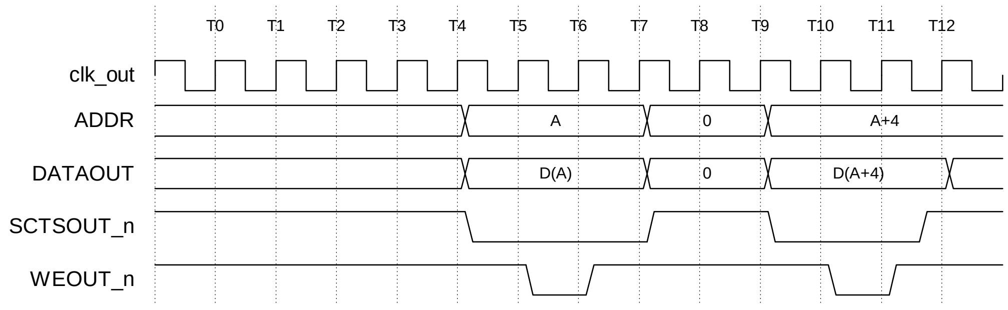


Figure 52: External memory 2 0 wait writes timing diagram

Timing parameter	Value
WAITRD	N/A
WAITOEN	N/A
WAITPAGE	N/A
WAITWR	0
WAITWEN	0
WAITTURN	0

Table 71: Static memory timing parameters

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.
T4-T5	Static memory transfer 0, address, chip select, and control signals submitted to static memory. Write data 0 is read from the AHB memory port. Write enable inactive.
T5-T6	Write enable taken active. Write data submitted to static memory.

Table 72: External memory 2 0 wait writes

Cycle	Description
T6-T7	Static memory writes data 0. Write enable taken inactive. Write data 1 is read from AHB memory port.
T7-T8	Static memory control signals taken inactive.
T8-T9	Memory controller processing.
T9-T10	Static memory transfer 1, address, chip select, and control signals submitted to static memory. Write enable inactive. Write data submitted to static memory.
T10-T11	Write enable taken active.
T11-T12	Static memory writes data 1. Write enable taken inactive.

Table 72: External memory 2 0 wait writes

Flash memory
Write timing for flash memory is the same as for SRAM devices.

Bus turnaround

The memory controller can be configured for each memory bank to use external bus turnaround cycles between read and write memory accesses. The WAITTURN field can be programmed for 1 to 16 turnaround wait states, to avoid bus contention on the external memory databus. Bus turnaround cycles are generated between external bus transfers as follows:

- Read to read (different memory banks
- Read to write (same memory bank)
- Read to write (different memory banks)

Figure 53 shows a zero wait read followed by a zero wait write with default turnaround between the transfers of two cycles because of the timing of the AHB transfers. Table 73 provides the timing parameters. Table 74 describes the transactions for Figure 53.

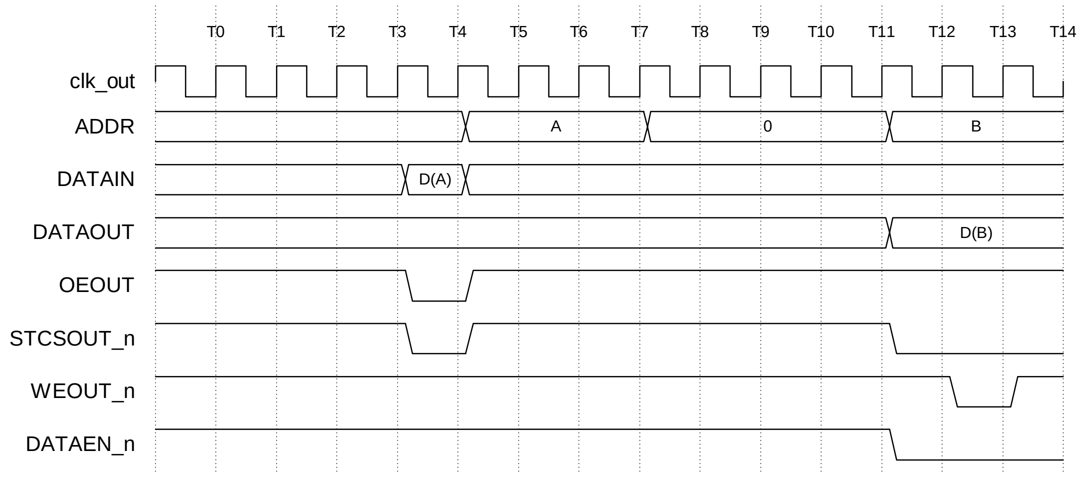


Figure 53: Read followed by write (both 0 wait) with no turnaround

Timing parameter	Value
WAITRD	0
WAITOEN	0
WAITPAGE	N/A
WAITWR	0
WAITWEN	0
WAITTURN	0

Table 73: Static memory timing parameters

Cycle	Description
T0	AHB address provided to the memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.
T4-T5	AHB write address provided to memory controller.

Table 74: Read followed by write (both 0 wait) with no turnaround

Cycle	Description
T5-T6	Memory controller processing.
T6-T7	Memory controller processing.
T7-T8	Static memory transfer address, chip select, and control signals submitted to static memory. Write data is read from AHB memory port. Write enable inactive.
T8-T9	Write enable taken active. Write data submitted to static memory.
T9-T10	Static memory control signals taken inactive.
T10-T11	Memory controller processing.
T11-T12	Static memory transfer 1, address, chip select, and control signals submitted to static memory. Write enable inactive. Write data submitted to static memory.
T12-T13	Write enable taken active.
T13-T14	Static memory writes data 1. Write enable taken inactive.

Table 74: Read followed by write (both 0 wait) with no turnaround

Figure 54 shows a zero wait write followed by a zero wait read with default turnaround between the transfers of one cycle. Three wait states are added to the write transfer; five wait states are added to the read transfer. The five AHB arbitration cycles for the read transfer include two wait states to allow the previous write access to complete and the three standard wait states for the read transfer.

Table 75 provides the timing parameters. Table 76 describes the transactions for Figure 54.

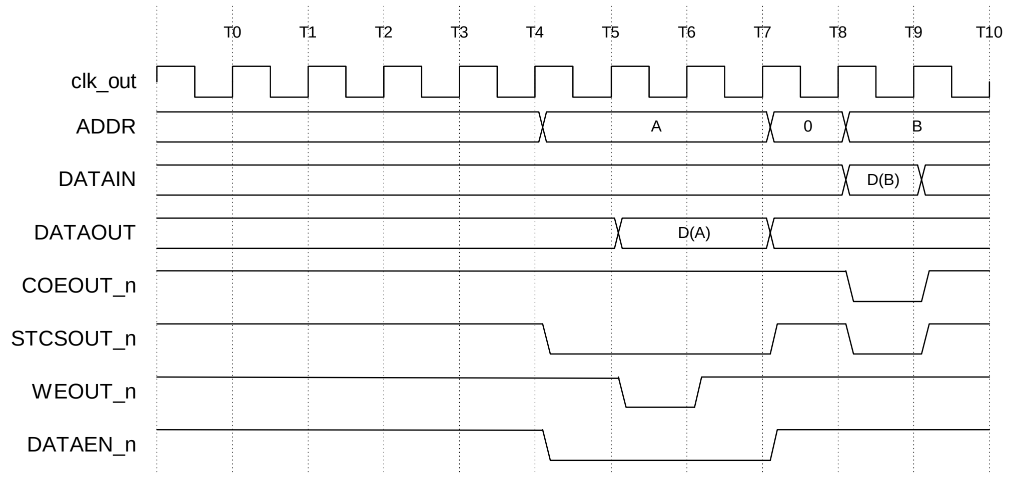


Figure 54: Write followed by a read (both 0 wait) with no turnaround

Timing parameters	Value
WAITRD	0
WAITOEN	0
WAITPAGE	N/A
WAITWR	0
WAITWEN	0
WAITTURN	0

Table 75: Static memory timing parameters

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of memory ports.

Table 76: Write followed by read (both 0 wait) with no turnaround

Cycle	Description
T4-T5	Static memory address, chip select, and control signals submitted to static memory. Write data is read from AHB memory port. Write enable inactive. AHB read address provided to memory controller.
T5-T6	Write enable taken active. Write data submitted to static memory.
T6-T7	Static memory writes the data. Write enable taken inactive.
T7-T8	Static memory control signals taken inactive.
T8-T9	Static memory address, chip select, and control signals submitted to static memory.
T9-T10	Read data returned from the static memory. Data is provided to the AHB.

Table 76: Write followed by read (both 0 wait) with no turnaround

Figure 55 shows a zero wait read followed by a zero wait write with two turnaround cycles added. The standard minimum of three AHB arbitration cycles are added to the read transfer and two wait states are added to the write transfer (as for any read-write transfer sequence).

Table 77 provides the timing parameters. Table 78 describes the transactions for Figure 55.

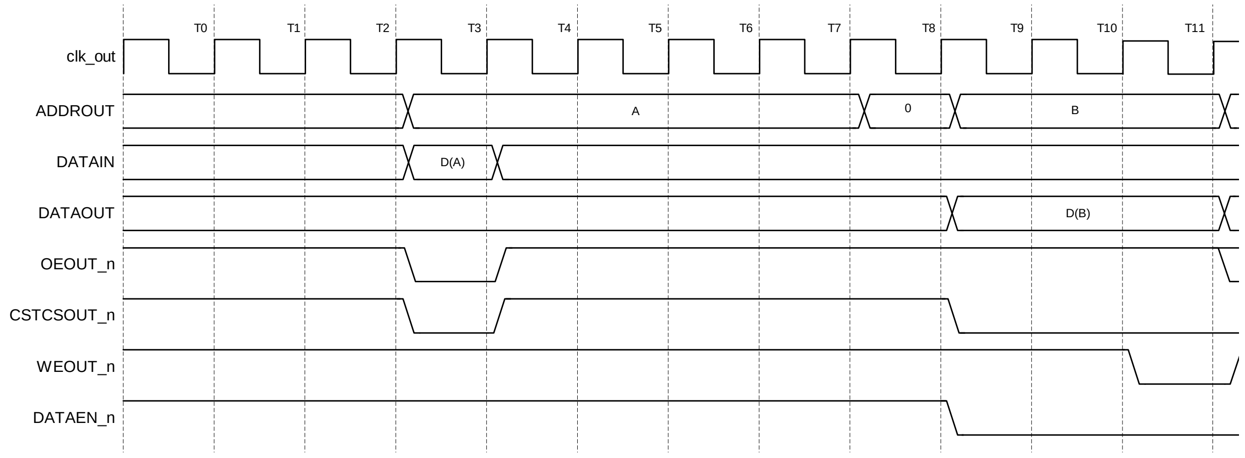


Figure 55: Read followed by a write (all 0 wait state) with two turnaround cycles

Timing parameters	Value
WAITRD	0
WAITOEN	0
WAITPAGE	N/A
WAITWR	0
WAITWEN	0
WAITTURN	2

Table 77: Static memory timing parameters

Cycle	Description
T0	AHB address provided to memory controller.
T0-T1	AHB transaction processing.
T1-T4	Arbitration of AHB memory ports.

Table 78: Read followed by a write (all 0 wait state) with two turnaround cycles

Cycle	Description
T4-T5	Static memory address, chip select, and control signals submitted to static memory.
T5-T6	Read data returned from static memory. Data is provided to the AHB. AHB write address provided to memory controller.
T6-T7	Turn around cycle 1.
T7-T8	Turn around cycle 2.
T8-T9	Static memory transfer address, chip select, and control signals submitted to static memory. Write enable inactive.
T9-T10	Memory controller processing.
T10-T11	Write enable taken active. Write data submitted to static memory.
T11-T12	Static memory writes the data. Write enable taken inactive.

Table 78: Read followed by a write (all 0 wait state) with two turnaround cycles

Byte lane control

The memory controller generates the byte lane control signals BLSOUT[3:0]_n according to these attributes:

- Little or big endian operation
- AMBA transfer width, indicated by HSIZE[2:0]
- External memory bank databus width, defined within each control register
- The decoded HADDR[1:0] value for write accesses only

Word transfers are the largest size transfers supported by the memory controller. Any access tried with a size greater that a word causes an error response. Each memory chip select can be 8, 16, or 32 bits wide. The memory type used determines how the WEOUT_n and BLSOUT_n signals are connected to provide byte, halfword, and word access.

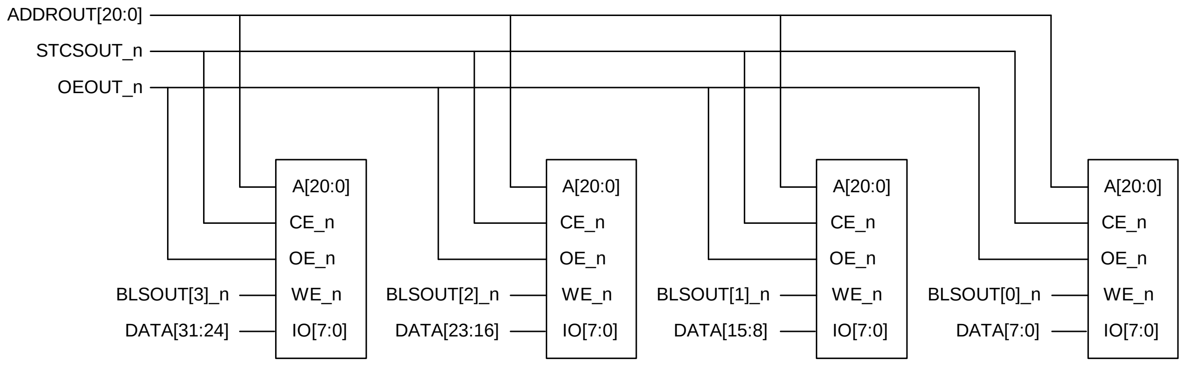
For read accesses, you must control the BLSOUT_n signals by driving them all high or all low. Do this by programming the byte lane state (PB) bit in the Static Configuration [3:0] register. "Memory banks constructed from 8-bit or non-byte-partitioned memory devices" on page 150 and "Memory banks constructed from 8-bit or non-byte-partitioned memory devices" on page 150 explain why different connections, with respect to WEOUT_n and BLSOUT[3:0]_n, for different memory configurations.

Address connectivity

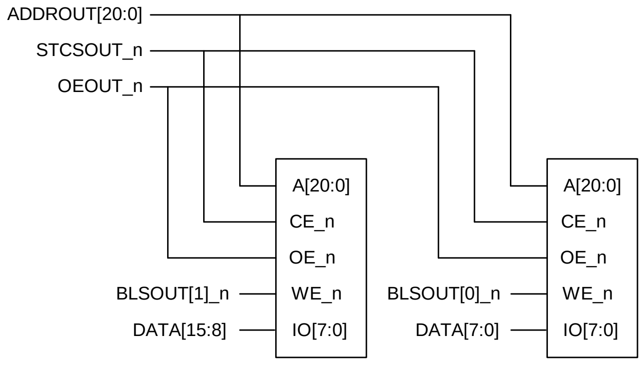
The static memory address output signal ADDROUT[27:0] must be right-justified.

Memory banks constructed from 8-bit or non-byte-partitioned memory devices

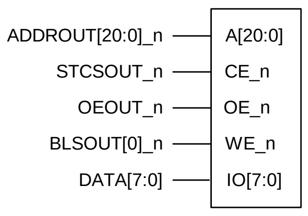
For memory banks constructed from 8-bit or non-byte-partitioned memory devices, it is important that the byte lane state (PB) bit is cleared to 0 within the respective memory bank control register. This forces all BLSOUT[3:0]_n lines high during a read access, as the byte lane selects are connected to the device write enables.



32-bit bank consisting of four 8-bit devices



16-bit bank consisting of two 8-bit devices



8-bit bank consisting of one 8-bit device

Figure 56: Memory banks constructed from 8-bit memory

Figure 56 shows 8-bit memory configuring memory banks that are 8-, 16-, and 32-bits wide. In each of these configurations, the BLSOUT[3:0]_n signals are connected to write enable (WE_n) inputs of each 8-bit memory. The WEOUT signal from the memory controller is not used.

- For write transfers, the appropriate BLSOUT[3:0]_n byte lane signals are asserted low, and direct the data to the addressed bytes.
- For read transfers, all BLSOUT[3:0]_n signals are deasserted high, enabling the external bus to be defined for at least the width of the accessed memory.

Memory banks constructed from 16-or 32-bit memory devices

For memory banks constructed from 16- or 32-bit memory devices, it is important that the byte lane select (PB) bit is set to 1 within the respective memory bank control register. This asserts all BLSOUT[3:0]_n lines low during a read access as, during a read, all device bytes must be selected to avoid undriven byte lanes on the read data value. With 16- and 32-bit wide memory devices, byte select signals exist and must be appropriately controlled; see Figure 57, "Memory banks constructed from 16-bit memory," on page 151 and Figure 58, "Memory banks constructed from 32-bit memory," on page 152.

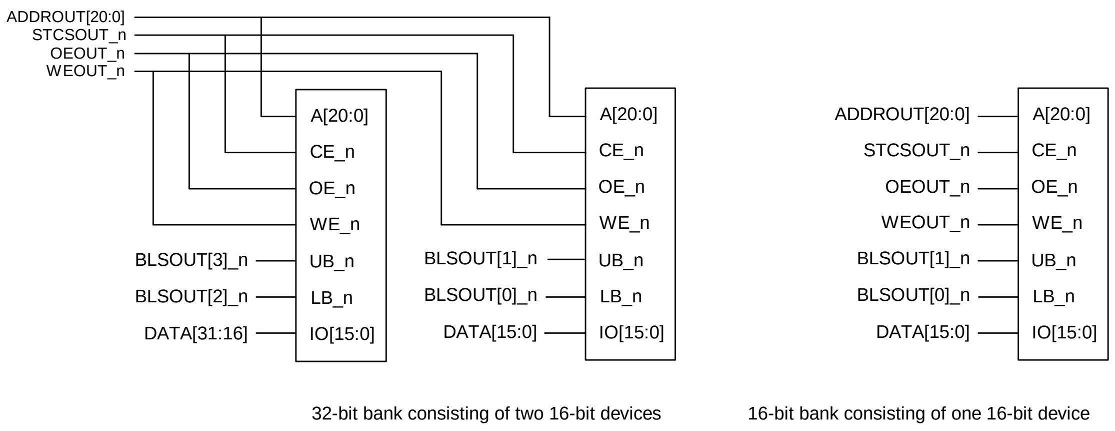


Figure 57: Memory banks constructed from 16-bit memory

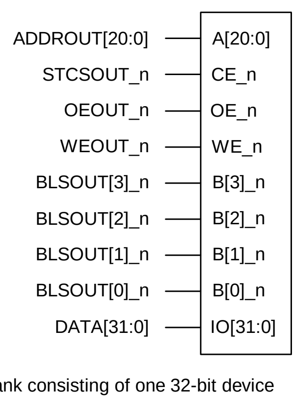


Figure 58: Memory banks constructed from 32-bit memory

Figure 59 shows connections for a typical memory system with different data width memory devices.

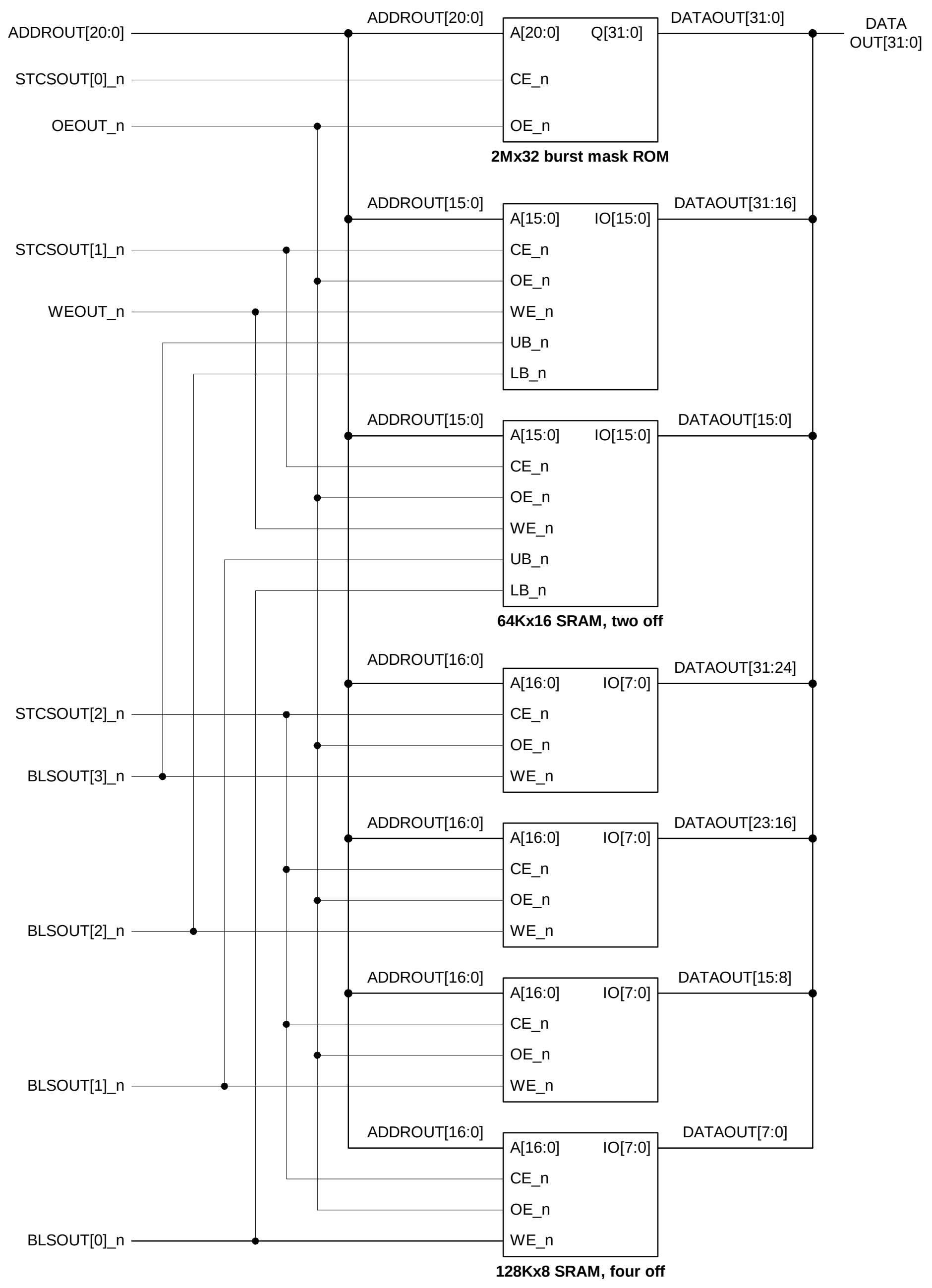


Figure 59: Typical memory connection diagram (1)

Byte lane control and databus steering

For little and big endian configurations, address right-justified

The tables in this section (Table 79 through Table 125) show the relationship of signals HSIZE[2:0], HADDR[1:0], ADDROUT[1:0], and BLSOUT[3:0] and mapping of data between the AHB system databus and the external memory databus. This mapping applies to both the static and dynamic memory controllers.

Internal transfer width	**Access: Read, little endian, 8-bit external bus**				**External data mapping on to system databus**			
					HRDATA to DATA			
	HSIZE [2:0]	**HADDR [1:0]**	**ADDROUT [1:0]**	**BLSOUT [0]**	**[31:24]**	**23:16]**	**[15:8]**	**[7:0]**
Word (4 transfers)	010	--	11	0	[7:0]	-	-	-
			10	0	-	[7:0]	-	-
			01	0	-	-	[7:0]	-
			00	0	-	-	-	[7:0]
Halfword (2 transfers)	001	1-	11	0	[7:0]	-	-	-
			10	0	-	[7:0]	-	-
Halfword (2 transfers)	001	0-	01	0	-	-	[7:0]	-
			00	0	-	-	-	[7:0]
Byte	000	11	11	0	[7:0]	-	-	-
Byte	000	10	10	0	-	[7:0]	-	-
Byte	000	01	01	0	-	-	[7:0]	-
Byte	000	00	00	0	-	-	-	[7:0]

Table 79: Little endian read, 8-bit external bus

Internal transfer width	**Access: Read, little endian, 16-bit external bus**				**External data mapping on to system databus**			
					HRDATA to DATA			
	HSIZE [2:0]	**HADDR [1:0]**	**ADDROUT [0]**	**BLSOUT [1:0]**	**[31:24]**	**23:16]**	**[15:8]**	**[7:0]**
Word (2 transfers	010	--	1	00	[15:8]	[7:0]	-	-
			0	00	-	-	[15:8]	[7:0]
Halfword	001	1-	1	00	[15:8]	[7:0]	-	-
Halfword	001	0-	0	00	-	-	[15:8]	[7:0]
Byte	000	11	1	01	[15:8]	-	-	-
Byte	000	10	1	10	-	[7:0]	-	-
Byte	000	01	0	01	-	-	[15:8]	-
Byte	000	00	0	10	-	-	-	[7:0]

Table 80: Little endian read, 16-bit external bus

Internal transfer width	**Access: Read, little endian, 32-bit external bus**			**External data mapping on to system databus**			
				HRDATA to DATA			
	HSIZE [2:0]	**HADDR [1:0]**	**BLSOUT [3:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	010	--	0000	[31:24]	[23:16]	[15:8]	[7:0]
Halfword	001	1-	0011	[31:24]	[23:16]	-	-
Halfword (2 transfers)	001	0-	1100	-	-	[15:8]	[7:0]
Byte	000	11	0111	[31:24]	-	-	-

Table 81: Little endian read, 32-bit external bus

Internal transfer width	Access: Read, little endian, 32-bit external bus			External data mapping on to system databus HRDATA to DATA			
	HSIZE [2:0]	HADDR [1:0]	BLSOUT [3:0]	[31:24]	[23:16]	[15:8]	[7:0]
Byte	000	10	1011	-	[23:16]	-	-
Byte	000	01	1101	-	-	[15:8]	-
Byte	000	00	1110	-	-	-	[7:0]

Table 81: Little endian read, 32-bit external bus

Internal transfer width	Access: Write, little endian, 8-bit external bus				System data mapping on to external databus DATA to HRDATA			
	HSIZE [2:0]	HADDR [1:0]	ADDROUT [1:0]	BLSOUT [0]	[31:24]	[23:16]	[15:8]	[7:0]
Word (4 transfers	010	--	11	0	-	-	-	[31:24]
			10	0	-	-	-	[23:16]
			01	0	-	-	-	[15:8]
			00	0	-	-	-	[7:0]
Halfword (2 transfers)	001	1-	11	0	-	-	-	[31:24]
			10	0	-	-	-	[23:16]
Halfword (2 transfers)	001	0-	01	0	-	-	-	[15:8]
			00	0	-	-	-	[7:0]
Byte	000	11	11	0	-	-	-	[31:24]
Byte	000	10	10	0	-	-	-	[23:16]
Byte	000	01	01	0	-	-	-	[15:8]
Byte	000	00	00	0	-	-	-	[7:0]

Table 82: Little endian write, 8-bit external bus

Internal transfer width	Access: Write, little endian, 16-bit external bus				System data mapping on to external databus DATA to HRDATA			
	HSIZE [2:0]	HADDR [1:0]	ADDROUT [0]	BLSOUT [1:0]	[31:24]	[23:16]	[15:8]	[7:0]
Word (2 transfers	010	--	1 0	00 00	-- -	- -	[31:24] [15:8]	[23:16] [7:0]
Halfword	001	1-	1	00	-	-	[31:24]	[23:16]
Halfword	001	0-	0	00	-	-	[15:8]	[7:0]
Byte	000	11	1	01	-	-	[31:24]	-
Byte	000	10	1	10	-	-	-	[23:16]
Byte	000	01	0	01	-	-	[15:8]	-
Byte	000	00	0	10	-	-	-	[7:0]

Table 83: Little endian write, 16-bit external bus

Internal transfer width	Access: Write, little endian, 32-bit external bus			System data mapping on to external databus DATA to HRDATA			
	HSIZE [2:0]	HADDR [1:0]	BLSOUT [3:0]	[31:24]	[23:16]	[15:8]	[7:0]
Word	010	--	0000	[31:24]	[23:16]	[15:8]	[7:0]
Halfword	001	1-	0011	[31:24]	[23:16]	-	-
Halfword	001	0-	1100	-	-	[15:8]	[7:0]
Byte	000	11	0111	[31:24]	-	-	-
Byte	000	10	1011	-	[23:16]	-	-

Table 84: Little endian write, 32-bit external bus

Internal transfer width	Access: Write, little endian, 32-bit external bus			System data mapping on to external databus			
				DATA to HRDATA			
	HSIZE [2:0]	**HADDR [1:0]**	**BLSOUT [3:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Byte	000	01	1101	-	-	[15:8]	-
Byte	000	00	1110	-	-	-	[7:0]

Table 84: Little endian write, 32-bit external bus

Internal transfer width	Access: Read, big endian, 8-bit external bus				External data mapping on to system databus			
					HRDATA to DATA			
	HSIZE [2:0]	**HADDR [1:0]**	**ADDROUT [1:0]**	**BLSOUT [0]**	**[31:24]**	**23:16]**	**[15:8]**	**[7:0]**
Word (4 transfers	010	--	11	0	-	-	-	[7:0]
			10	0	-	-	[7:0]	-
			01	0	-	[7:0]	-	-
			00	0	[7:0]	-		-
Halfword (2 transfers)	001	1-	11	0	-	-	-	[7:0]
			10	0	-	-	[7:0]	-
Halfword (2 transfers)	001	0-	01	0	-	[7:0]	-	-
			00	0	[7:0]	-	-	-
Byte	000	11	11	0	-	-	-	[7:0]
Byte	000	10	10	0	-	-	[7:0]	-
Byte	000	01	01	0	-	[7:0]	-	-
Byte	000	00	00	0	[7:0]	-	-	-

Table 85: Big endian read, 8-bit external bus

Internal transfer width	Access: Read, big endian, 16-bit external bus				External data mapping on to system databus			
					HRDATA to DATA			
	HSIZE [2:0]	**HADDR [1:0]**	**ADDROUT [1:0]**	**BLSOUT [1:0]**	**[31:24]**	**23:16]**	**[15:8]**	**[7:0]**
Word (2 transfers	010	--	1-	00	-	-	[15:8]	[7:0]
			0-	00	[15:8]	[7:0]	-	-
Halfword	001	1-	1-	00	-	-	[15:8]	[7:0]
Halfword	001	0-	0-	00	[15:8]	[7:0]	-	-
Byte	000	11	1-	10	-	-	-	[7:0]
Byte	000	10	1-	01	-	-	[15:8]	-
Byte	000	01	0-	10	-	[7:0]	-	-
Byte	000	00	0-	01	[15:8]	-	-	-

Table 86: Big endian read, 16-bit external bus

Internal transfer width	Access: Read, big endian, 32-bit external bus				External data mapping on to system databus			
					HRDATA to DATA			
	HSIZE [2:0]	**HADDR [2:1]**	**ADDROUT [1:0]**	**BLSOUT [3:0]**	**[31:24]**	**23:16]**	**[15:8]**	**[7:0]**
Word	010	--	--	0000	[31:24]	[23:16]	[15:8]	[7:0]
Halfword	001	1-	--	1100	-	-	[15:8]	[7:0]
Halfword (2 transfers)	001	0-	--	0011	[31:24]	[23:16]	-	-
Byte	000	11	--	1110	-	-	-	[7:0]
Byte	000	10	--	1101	-	-	[15:8]	-

Table 87: Big endian read, 32-bit external bus

Internal transfer width	Access: Read, big endian, 32-bit external bus				External data mapping on to system databus HRDATA to DATA			
	HSIZE [2:0]	HADDR [2:1]	ADDROUT [1:0]	BLSOUT [3:0]	[31:24]	23:16]	[15:8]	[7:0]
Byte	000	01	--	1011	-	[23:16]	-	-
Byte	000	00	--	0111	[31:24]	-	-	-

Table 87: Big endian read, 32-bit external bus

Internal transfer width	Access: Write, big endian, 8-bit external bus				System data mapping on to external databus DATA to HRDATA			
	HSIZE [2:0]	HADDR [1:0]	ADDROUT [1:0]	BLSOUT [0]	[31:24]	[23:16]	[15:8]	[7:0]
Word (4 transfers	010	--	11	0	-	-	-	[7:0]
			10	0	-	-	-	[15:8]
			01	0	-	-	-	[23:16]
			00	0	-	-	-	[31:24]
Halfword (2 transfers)	001	1-	11	0	-	-	-	[7:0]
			10	0	-	-	-	[15:8]
Halfword (2 transfers)	001	0-	01	0	-	-	-	[23:16]
			00	0	-	-	-	[31:24]
Byte	000	11	11	0	-	-	-	[7:0]
Byte	000	10	10	0	-	-	-	[15:8]
Byte	000	01	01	0	-	-	-	[23:16]
Byte	000	00	00	0	-	-	-	[31:24]

Table 88: Big endian write, 8-bit external bus

Internal transfer width	Access: Write, big endian, 16-bit external bus				System data mapping on to external databus DATA to HRDATA			
	HSIZE [2:0]	HADDR [1:0]	ADDROUT [1:0]	BLSOUT [1:0]	[31:24]	[23:16]	[15:8]	[7:0]
Word (2 transfers	010	--	1- 0-	00 00	-- -	- -	[15:8][31:24]	[7:0] [23:16]
Halfword	001	1-	1-	00	-	-	[15:8]	[7:0]
Halfword	001	0-	0-	00	-	-	[31:24]	[23:16]
Byte	000	11	1-	10	-	-	-	[7:0]
Byte	000	10	1-	01	-	-	[15:8]	-
Byte	000	01	0-	10	-	-	-	[23:16]
Byte	000	00	0-	01	-	-	[31:24]	-

Table 89: Big endian write, 16-bit external bus

Internal transfer width	Access: Write, big endian, 32-bit external bus			System data mapping on to external databus DATA			
	HSIZE [2:0]	HADDR [1:0]	BLSOUT [3:0]	[31:24]	[23:16]	[15:8]	[7:0]
Word	010	--	0000	[31:24]	[23:16]	[15:8]	[7:0]
Halfword	001	1-	1100	-	-	[15:8]	[7:0]
Halfword	001	0-	0011	[31:24]	[23:16]	-	-
Byte	000	11	1110	-	-	-	[7:0]
Byte	000	10	1101	-	-	[15:8]	-

Table 90: Big endian write, 32-bit external bus

Internal transfer width	Access: Write, big endian, 32-bit external bus			System data mapping on to external databus			
				DATA			
	HSIZE [2:0]	HADDR [1:0]	BLSOUT [3:0]	[31:24]	[23:16]	[15:8]	[7:0]
Byte	000	01	1011	-	[23:16]	-	-
Byte	000	00	0111	[31:24]	-	-	-

Table 90: Big endian write, 32-bit external bus

Dynamic memory controller

Write protection

Each dynamic memory chip select can be configured for write-protection by setting the appropriate bit in the write protect (P) field on the Dynamic Memory Configuration register (see "Dynamic Memory Configuration 0–3 registers" on page 225). If a write access is performed to a write-protected memory bank, an ERROR response is generated on the HRESP[1:0] signal.

Access sequencing and memory width

The data width of each chip select must be configured by programming the appropriate Dynamic Memory Configuration register. When the chip select data bus width is narrower than the transfer initiated from the current AMBA bus master, the internal bus transfer takes several external bus transfers to complete. If chip select 4 is configured as 16-bit wide memory, for example, and a 32-bit read is initiated, the AHB bus stalls while the memory controller reads two consecutive words from memory. During these accesses, the memory controller block demultiplexes the two 16-bit words into one 32-bit word and places the result onto the AHB bus.

Word transfers are the widest transfers supported by the memory controller. Any access tried with a size larger than a word generates an error response.

Address mapping

This section provides tables that show how AHB address bus addresses map to the external dynamic memory address ADDROUT[14:0] for different memory configurations and bus widths. The address mapping is selected by programming the address mapping bits in the Dynamic Memory Configuration registers (see "Dynamic Memory Configuration 0–3 registers" on page 225).

The information provided includes:

- **Memory controller output address (ADDROUT).** Indicates the address lines output from the memory controller.
- **Memory device connections.** Indicate the device signals that must be connected to the memory controller AddrOut lines.
- **AHB addresses to row address.** Indicates the input HADDR address bits used from the AHB transfer for the row access.
- **AHB address to column address.** Indicates the input HADDR address bits used from the AHB transfer for the column access.

Notes:

- For all tables in this section:
  - ** indicates that the bit is controlled by the SDRAM controller. The SDRAM controller always transfers 32-bits of data at a time. For chip selects with a 16-bit wide databus, the SDRAM controller performs two transfers: a column transfer with the lowest bit set to 0 and a column transfer with the lowest bit set to 1.
  - BA, BA0, and BA1 indicate the bank address signals. AP indicates the auto precharge signal (usually, address bit 10).
- Separate tables are provided for two different address mapping schemes: row, bank, column (RBC) or bank, row, column (BRC), and for 32-bit and 16-bit wide buses:
  - **32-bit wide databus address mappings, SDRAM (RBC)** (see "32-bit wide databus address mappings, SDRAM (RBC)" on page 164). These address mappings are used for 32-bit data bus chip select with SDR-SDRAM memory

devices. The row-bank-column address mapping scheme allows memory accesses to be performed efficiently to nearby memory regions.

– **32-bit wide databus address mappings (BRC)** (see "32-bit wide databus address mappings (BRC)" on page 175). These address mappings are used for 32-bit data bus chip select with SDR-SDRAM or low power SDR-SDRAM. The bank-row-column address mapping scheme allows the low-power SDR-SDRAM memory features to be used efficiently.

– **16-bit wide databus address mappings, SDRAM (RBC)** (see "16-bit wide databus address mappings, SDRAM (RBC)" on page 185). These address mappings are used for 16-bit data bus chip select with SDR-SDRAM memory devices. The row-bank-column address mapping scheme allows memory accesses to be performed efficiently to nearby memory regions.

– **16-bit wide databus address mappings (BRC)** (see "16-bit wide databus address mappings (BRC)" on page 193). These address mappings are used for 16-bit data bus chip select with SDR-SDRAM and low-power SDR-SDRAM. The bank-row-column address mapping scheme allows the low-power SDR-SDRAM memory features to be used efficiently.

32-bit wide databus address mappings, SDRAM (RBC)

Table 91 through Table 103 show 32-bit wide databus address mappings for several SDRAM (RBC) devices.

Table 91 shows the outputs from the memory controller and the corresponding inputs to the 16M SDRAM (1Mx16, pin 13 used as bank select).

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
14	-	-	-
13	BA	10	10
12	-	-	-
11	-	-	-
10	10/AP	21	AP
9	9	20	-

Table 91: Address mapping for 16M SDRAM (1Mx16, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
8	8	19	-
7	7	18	9
6	6	17	8
5	5	16	7
4	4	15	6
3	3	14	5
2	2	13	4
1	1	12	3
0	0	11	2

Table 91: Address mapping for 16M SDRAM (1Mx16, RBC)

Table 92 shows the outputs from the memory controller and the corresponding inputs to the 16M SDRAM (2Mx8, pin 14 used as bank select).

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
14	BA	11	11
13	-	-	-
12	-	-	-
11	-	-	-
10	10/AP	22	AP
9	9	21	-
8	8	20	10
7	7	19	9
6	6	18	8
5	5	17	7
4	4	16	6

Table 92: Address mapping for 16M SDRAM (2Mx8, RBC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
3	3	15	5
2	2	14	4
1	1	13	3
0	0	12	2

Table 92: Address mapping for 16M SDRAM (2Mx8, RBC)

Table 93 shows the outputs from the memory controller and the corresponding inputs to the 64M SDRAM (2Mx32, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	11	11
13	BA0	10	10
12	-	-	-
11	-	-	-
10	10/AP	22	AP
9	9	21	-
8	8	20	-
7	7	19	9
6	6	18	8
5	5	17	7
4	4	16	6
3	3	15	5
2	2	14	4
1	1	13	3
0	0	12	2

Table 93: Address mapping for 64M SDRAM (2Mx32, RBC)

Table 94 shows the outputs from the memory controller and the corresponding inputs to the 64M SDRAM (4Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	11	11
13	BA0	10	10
12	-	-	-
11	11	23	-
10	10/AP	22	AP
9	9	21	-
8	8	20	-
7	7	199	9
6	6	18	8
5	5	17	7
4	4	16	6
3	3	15	5
2	2	14	4
1	1	13	3
0	0	12	2

Table 94: Address mapping for 64M SDRAM (4Mx16, RBC)

Table 95 shows the outputs from the memory controller and the corresponding inputs to the 64 M SDRAM (8Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	11	11
13	BA0	12	12
12	-	-	-
11	11	24	-
10	10/AP	23	AP
9	9	22	-
8	8	21	10
7	7	20	9
6	6	19	8
5	5	18	7
4	4	17	6
3	3	16	5
2	2	15	4
1	1	14	3
0	0	13	2

Table 95: Address mapping for 64M SDRAM (8Mx8, RBC)

Table 96 shows the outputs from the memory controller and the corresponding inputs to the 128M SDRAM (4Mx32, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	11	11
13	BA0	10	10
12	-	-	-

Table 96: Address mapping for 128M SDRAM (4Mx32, RBC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
11	11	23	-
10	10/AP	22	AP
9	9	21	-
8	8	20	-
7	7	19	9
6	6	18	8
5	5	17	7
4	4	16	6
3	3	15	5
2	2	14	4
1	1	13	3
0	0	12	2

Table 96: Address mapping for 128M SDRAM (4Mx32, RBC)

Table 97 shows the outputs from the memory controller and the corresponding inputs to the 64M SDRAM (8Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	11	11
13	BA0	12	12
12	-	-	-
11	11	24	-
10	10/AP	23	AP
9	9	22	-
8	8	21	10
7	7	20	9

Table 97: Address mapping for 128 SDRAM (8Mx16, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
6	6	19	8
5	5	18	7
4	4	17	6
3	3	16	5
2	2	15	4
1	1	14	3
0	0	13	2

Table 97: Address mapping for 128 SDRAM (8Mx16, RBC)

Table 98 shows the outputs from the memory controller and the corresponding inputs to the 128M SDRAM (16Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
14	BA1	13	13
13	BA0	12	12
12	12	-	-
11	11	25	-
10	10/AP	24	AP
9	9	23	11
8	8	22	10
7	7	21	9
6	6	20	8
5	5	19	7
4	4	18	6
3	3	17	5
2	2	16	4

Table 98: Address mapping for 128 SDRAM (16Mx8, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
1	1	15	3
0	0	14	2

Table 98: Address mapping for 128 SDRAM (16Mx8, RBC)

Table 99 shows the outputs from the memory controller and the corresponding inputs to the 256M SDRAM (8Mx32, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
14	BA1	11	11
13	BA0	10	10
12	12	24	-
11	11	23	-
10	10/AP	22	AP
9	9	21	-
8	8	20	-
7	7	19	9
6	6	18	8
5	5	17	7
4	4	16	6
3	3	15	5
2	2	14	4
1	1	13	3
0	0	12	2

Table 99: Address mapping for 256 SDRAM (8Mx32, RBC)

Table 100 shows the outputs from the memory controller and the corresponding inputs to the 256M SDRAM (16Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	11	11
13	BA0	12	12
12	12	25	-
11	11	24	-
10	10/AP	23	AP
9	9	22	-
8	8	21	10
7	7	20	9
6	6	19	8
5	5	18	7
4	4	17	6
3	3	16	5
2	2	15	4
1	1	14	3
0	0	13	2

Table 100: Address mapping for 256M SDRAM (16Mx16, RBC)

Table 101 shows the outputs from the memory controller and the corresponding inputs to the 256M SDRAM (32Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	13	13
13	BA0	12	12
12	12	26	-

Table 101: Address mapping for 256M SDRAM (32Mx8, RBC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
11	11	25	-
10	10/AP	24	AP
9	9	23	11
8	8	22	10
7	7	21	9
6	6	20	8
5	5	19	7
4	4	18	6
3	3	17	5
2	2	16	4
1	1	15	3
0	0	14	2

Table 101: Address mapping for 256M SDRAM (32Mx8, RBC)

Table 102 shows the outputs from the memory controller and the corresponding inputs to the 512M SDRAM (32Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	13	13
13	BA0	12	12
12	12	26	-
11	11	25	-
10	10/AP	24	AP
9	9	23	11
8	8	22	10
7	7	21	9

Table 102: Address mapping for 512M SDRAM (32Mx16, RBC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
6	6	20	8
5	5	19	7
4	4	18	6
3	3	17	5
2	2	16	4
1	1	15	3
0	0	14	2

Table 102: Address mapping for 512M SDRAM (32Mx16, RBC)

Table 103 shows the outputs from the memory controller and the corresponding inputs to the 512M SDRAM (64Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	13	13
13	BA0	14	14
12	12	27	-
11	11	26	12
10	10/AP	25	AP
9	9	24	11
8	8	23	10
7	7	22	9
6	6	21	8
5	5	20	7
4	4	19	6
3	3	18	5
2	2	17	4

Table 103: Address mapping for 512M SDRAM (64Mx8, RBC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
1	1	16	3
0	0	15	2

Table 103: Address mapping for 512M SDRAM (64Mx8, RBC)

32-bit wide databus address mappings (BRC)

Table 104 through Table 116 show 32-bit wide databus address mappings for several SDRAM (BRC) devices.

Table 104 shows the outputs from the memory controller and the corresponding inputs to the 16M SDRAM (1x16, pin 14 used as bank select).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA	21	21
13	-	-	-
12	-	-	-
11	-	-	-
10	10/AP	20	AP
9	9	19	-
8	8	18	-
7	7	17	9
6	6	16	8
5	5	15	7
4	4	14	6
3	3	13	5
2	2	12	4
1	1	11	3
0	0	10	2

Table 104: Address mapping for 16M SDRAM (1Mx16, BRC)

Table 105 shows the outputs from the memory controller and the corresponding inputs to the 16M SDRAM (2Mx8, pin 13 used as bank select).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	-	-	-
13	BA	22	22
12	-	-	-
11	-	-	-
10	10/AP	21	AP
9	9	20	-
8	8	19	10
7	7	18	9
6	6	17	8
5	5	16	7
4	4	15	6
3	3	14	5
2	2	13	4
1	1	12	3
0	0	11	2

Table 105: Address mapping for 16M SDRAM (2Mx8, BRC)

Table 106 shows the outputs from the memory controller and the corresponding inputs to the 64M SDRAM (2Mx32, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	21	21
13	BA0	22	22
12	-	-	-

Table 106: Address mapping for 64M SDRAM (2Mx32, BRC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
11	-	-	-
10	10/AP	20	AP
9	9	19	-
8	8	18	-
7	7	17	9
6	6	16	8
5	5	15	7
4	4	14	6
3	3	13	5
2	2	12	4
1	1	11	3
0	0	10	2

Table 106: Address mapping for 64M SDRAM (2Mx32, BRC)

Table 107 shows the outputs from the memory controller and the corresponding inputs to the 64M SDRAM (4Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	23	23
13	BA0	22	22
12	-	-	-
11	11	21	-
10	10/AP	20	AP
9	9	19	-
8	8	18	-
7	7	17	9

Table 107: Address mapping for 64M SDRAM (4Mx16, BRC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
6	6	16	8
5	5	15	7
4	4	14	6
3	3	13	5
2	2	12	4
1	1	11	3
0	0	10	2

Table 107: Address mapping for 64M SDRAM (4Mx16, BRC)

Table 108 shows the outputs from the memory controller and the corresponding inputs to the 64M SDRAM (8Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	23	23
13	BA0	24	24
12	-	-	-
11	11	22	-
10	10/AP	21	AP
9	9	20	-
8	8	19	10
7	7	18	9
6	6	17	8
5	5	16	7
4	4	15	6
3	3	14	5
2	2	13	4

Table 108: Address mapping for 64M SDRAM (8Mx8, BRC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
1	1	12	3
0	0	11	2

Table 108: Address mapping for 64M SDRAM (8Mx8, BRC)

Table 109 shows the outputs from the memory controller and the corresponding inputs to the 128M SDSRAM (4Mx32, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	23	23
13	BA0	22	22
12	-	-	-
11	11	21	-
10	10/AP	20	AP
9	9	19	-
8	8	18	-
7	7	17	9
6	6	16	8
5	5	15	7
4	4	14	6
3	3	13	5
2	2	12	4
1	1	11	3
0	0	10	2

Table 109: Address mapping for 128M SDRAM (4Mx32, BRC)

Table 110 shows the outputs from the memory controller and the corresponding inputs to the 128M SDRAM (8Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	23	23
13	BA0	24	24
12	-	-	-
11	11	22	-
10	10/AP	21	AP
9	9	20	-
8	8	19	10
7	7	18	9
6	6	17	8
5	5	16	7
4	4	15	6
3	3	14	5
2	2	13	4
1	1	12	3
0	0	11	2

Table 110: Address mapping for 128M SDRAM (8Mx16, BRC)

Table 111 shows the outputs from the memory controller and the corresponding inputs to the 128M SDRAM (16Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	25	25
13	BA0	24	24
12	12	-	-

Table 111: Address mapping for 128M SDRAM (16Mx8, BRC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
11	11	23	-
10	10/AP	22	AP
9	9	21	11
8	8	20	10
7	7	19	9
6	6	18	8
5	5	17	7
4	4	16	6
3	3	15	5
2	2	14	4
1	1	13	3
0	0	12	2

Table 111: Address mapping for 128M SDRAM (16Mx8, BRC)

Table 112 shows the outputs from the memory controller and the corresponding inputs to the 256M SDRAM (8Mx32, pins 13 and 14 used as bank selects.

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	23	23
13	BA0	24	24
12	12	22	-
11	11	21	-
10	10/AP	20	AP
9	9	19	-
8	8	18	-
7	7	17	9

Table 112: Address mapping for 256M SDRAM (8Mx32, BRC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
6	6	16	8
5	5	15	7
4	4	14	6
3	3	13	5
2	2	12	4
1	1	11	3
0	0	10	2

Table 112: Address mapping for 256M SDRAM (8Mx32, BRC)

Table 113 shows the outputs from the memory controller and the corresponding inputs to the 256M SDRAM (16Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
14	BA1	25	25
13	BA0	24	24
12	12	23	-
11	11	22	-
10	10/AP	21	AP
9	9	20	-
8	8	19	10
7	7	18	9
6	6	17	8
5	5	16	7
4	4	15	6
3	3	14	5
2	2	13	4

Table 113: Address mapping for 256M SDRAM (16Mx16, BRC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
1	1	12	3
0	0	11	2

Table 113: Address mapping for 256M SDRAM (16Mx16, BRC)

Table 114 shows the outputs from the memory controller and the corresponding inputs to the 256M SDRAM (32Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	25	25
13	BA0	26	26
12	12	24	-
11	11	23	-
10	10/AP	22	AP
9	9	21	11
8	8	20	10
7	7	19	9
6	6	18	8
5	5	17	7
4	4	16	6
3	3	15	5
2	2	14	4
1	1	13	3
0	0	12	2

Table 114: Address mapping for 256M SDRAM (32Mx8, BRC)

Table 115 shows the outputs from the memory controller and the corresponding inputs to the 512M SDRAM (32Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	25	25
13	BA0	26	26
12	12	24	-
11	11	23	-
10	10/AP	22	AP
9	9	21	11
8	8	20	10
7	7	19	9
6	6	18	8
5	5	17	7
4	4	16	6
3	3	15	5
2	2	14	4
1	1	13	3
0	0	12	2

Table 115: Address mapping for 512M SDRAM (32Mx16, BRC)

Table 116 shows the outputs from the memory controller and the corresponding inputs to the 512M SDRAM (64Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	27	27
13	BA0	26	26
12	12	25	-

Table 116: Address mapping for 512M SDRAM (64x8, BRC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
11	11	24	12
10	10/AP	23	AP
9	9	22	11
8	8	21	10
7	7	20	9
6	6	19	8
5	5	18	7
4	4	17	6
3	3	16	5
2	2	15	4
1	1	14	3
0	0	13	2

Table 116: Address mapping for 512M SDRAM (64x8, BRC)

16-bit wide databus address mappings, SDRAM (RBC)

Table 117 through Table 126 show 16-bit wide databus address mappings for SDRAM (RBC) devices.

Table 117 shows the outputs from the memory controller and the corresponding inputs to the 16M SDRAM (1Mx16, pin 14 used as bank select).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA	9	9
13	-	-	-
12	-	-	-
11	-	-	-
10	10/AP	20	AP

Table 117: Address mapping for 16M SDRAM (1Mx16, RBC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
9	9	19	-
8	8	18	-
7	7	17	8
6	6	16	7
5	5	15	6
4	4	14	5
3	3	13	4
2	2	12	3
0	0	10	**

Table 117: Address mapping for 16M SDRAM (1Mx16, RBC)

Table 118 shows the outputs from the memory controller and the corresponding inputs to the 16M SDRAM (2Mx8, pin 13 used as bank select).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA	-	-
13	-	10	10
12	-	-	-
11	-	-	-
10	10/AP	11	AP
9	9	21	-
8	8	20	9
7	7	19	8
6	6	18	7
5	5	17	6

Table 118: Address mapping for 16M SDRAM (2Mx8, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
4	4	16	5
3	3	15	4
2	2	14	3
1	1	13	2
0	0	12	**

Table 118: Address mapping for 16M SDRAM (2Mx8, RBC)

Table 119 shows the outputs from the memory controller and the corresponding inputs to the 64M SDRAM (4Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
14	BA1	9	9
13	BA0	10	10
12	-	-	-
11	11	22	-
10	10/AP	21	AP
9	9	20	-
8	8	19	-
7	7	18	8
6	6	17	7
5	5	16	6
4	4	15	5
3	3	14	4
2	2	13	3
1	1	12	2
0	0	11	**

Table 119: Address mapping for 64M SDRAM (4Mx16, RBC)

Table 120 shows the outputs from the memory controller and the corresponding inputs to the 64M SDRAM (8Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	11	11
13	BA0	10	10
12	-	-	-
11	11	23	-
10	10/AP	22	AP
9	9	21	-
8	8	20	9
7	7	19	8
6	6	18	7
5	5	17	6
4	4	16	5
3	3	15	4
2	2	14	3
1	1	13	2
0	0	12	**

Table 120: Address mapping for 64M SDRAM (8Mx8, RBC)

Table 121 shows the outputs from the memory controller and the corresponding inputs tot he 128M SDRAM (8Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	11	11
13	BA0	10	10
12	-	-	-

Table 121: Address mapping for 128M SDRAM (8Mx16, RBC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
11	11	23	-
10	10/AP	22	AP
9	9	21	-
8	8	20	9
7	7	19	8
6	6	18	7
5	5	17	6
4	4	16	5
3	3	15	4
2	2	14	3
1	1	13	2
0	0	12	**

Table 121: Address mapping for 128M SDRAM (8Mx16, RBC)

Table 122 shows the outputs from the memory controller and the corresponding inputs to the 128M SDRAM (16Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	11	11
13	BA0	12	12
12	-	-	-
11	11	24	-
10	10/AP	23	AP
9	9	22	10
8	8	21	9
7	7	20	8

Table 122: Address mapping for 128M SDRAM (16Mx8, RBC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
6	6	19	7
5	5	18	6
4	4	17	5
3	3	16	4
2	2	15	3
1	1	14	2
0	0	13	**

Table 122: Address mapping for 128M SDRAM (16Mx8, RBC)

Table 123 shows the outputs from the memory controller and the corresponding inputs to the 256M SDRAM (16Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	11	11
13	BA0	10	10
12	12	24	-
11	11	23	-
10	10/AP	22	AP
9	9	21	-
8	8	20	9
7	7	19	8
6	6	18	7
5	5	17	6
4	4	16	5
3	3	15	4
2	2	14	3

Table 123: Address mapping for 256M SDRAM (16Mx16, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
1	1	13	2
0	0	12	**

Table 123: Address mapping for 256M SDRAM (16Mx16, RBC)

Table 124 shows the outputs from the memory controller and the corresponding inputs to the 256M SDRAM (32Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
14	BA1	11	11
13	BA0	12	12
12	12	25	-
11	11	24	-
10	10/AP	23	AP
9	9	22	10
8	8	21	9
7	7	20	8
6	6	19	7
5	5	18	6
4	4	17	5
3	3	16	4
2	2	15	3
1	1	14	2
0	0	13	**

Table 124: Address mapping for 256M SDRAM (32Mx8, RBC)

Table 125 shows the outputs from the memory controller and the corresponding inputs to the 512M SDRAM (32Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	11	11
13	BA0	12	12
12	12	25	-
11	11	24	-
10	10/AP	23	AP
9	9	22	10
8	8	21	9
7	7	20	8
6	6	19	7
5	5	18	6
4	4	17	5
3	3	16	4
2	2	15	3
1	1	14	2
0	0	13	**

Table 125: Address mapping for 512M SDRAM (32Mx16, RBC)

Table 126 shows the outputs from the memory controller and the corresponding inputs to the 512M SDRAM (64Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	13	13
13	BA0	12	12
12	12	26	-
11	11	25	11
10	10/AP	24	AP
9	9	23	10
8	8	22	9
7	7	21	8
6	6	20	7
5	5	19	6
4	4	18	5
3	3	17	4
2	2	16	3
1	1	15	2
0	0	14	**

Table 126: Address mapping for 512M SDRAM (64Mx8, RBC)

16-bit wide databus address mappings (BRC)

Table 127 through Table 136 show 16-bit wide databus address mappings for SDRAM (BRC) devices.

Table 127 shows the outputs from the memory controller and the corresponding inputs to the 16M SDRAM (1Mx16, pin 13 used as bank select).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	-	-	-
13	BA	20	20
12	-	-	-
11	-	-	-
10	10/AP	19	AP
9	9	18	-
8	8	17	-
7	7	16	8
6	6	15	7
5	5	14	6
4	4	13	5
3	3	12	4
2	2	11	3
1	1	10	2
0	0	9	**

Table 127: Address mapping for 16M SDRAM (1Mx16, BRC)

Table 128 shows the outputs from the memory controller and the corresponding inputs to the 16M SDRAM (2Mx8, pin 14 used as a bank select).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA	21	21
13	-	-	-
12	-	-	-
11	-	-	-
10	10/AP	20	AP

Table 128: Address mapping for 16M SDRAM (2Mx8, BRC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
9	9	19	-
8	8	18	9
7	7	17	8
6	6	16	7
5	5	15	6
4	4	14	5
3	3	13	4
2	2	12	3
1	1	11	2
0	0	10	**

Table 128: Address mapping for 16M SDRAM (2Mx8, BRC)

Table 129 shows the outputs from the memory controller and the corresponding inputs to the 64M SDRAM (4Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	21	21
13	BA0	22	22
12	-	-	-
11	11	20	-
10	10/AP	19	AP
9	9	18	-
8	8	17	-
7	7	16	8
6	6	15	7
5	5	14	6

Table 129: Address mapping for 64M SDRAM (4Mx16, BRC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
4	4	13	5
3	3	12	4
2	2	11	3
1	1	10	2
0	0	9	**

Table 129: Address mapping for 64M SDRAM (4Mx16, BRC)

Table 130 shows the outputs from the memory controller and the corresponding inputs to the 64M SDRAM (8Mx*, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	23	23
13	BA0	22	22
12	-	-	-
11	11	21	-
10	10/AP	20	AP
9	9	19	-
8	8	18	9
7	7	17	8
6	6	16	7
5	5	15	6
4	4	14	5
3	3	13	4
2	2	12	3
1	1	11	2
0	0	10	**

Table 130: Address mapping for 64M SDRAM (8Mx8, BRC)

Table 131 shows the outputs from the memory controller and the corresponding inputs to the 128M SDRAM (8Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT**)**	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	23	23
13	BA0	22	22
12	-	-	-
11	11	21	-
10	10/AP	20	AP
9	9	19	-
8	8	18	9
7	7	17	8
6	6	16	7
5	5	15	6
4	4	14	5
3	3	13	4
2	2	12	3
1	1	11	2
0	0	10	**

Table 131: Address mapping for 128M SDRAM (8Mx16, BRC)

Table 132 shows the outputs from the memory controller and the corresponding inputs to the 128 SDRAM (16Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT**)**	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	23	23
13	BA0	24	24
12	-	-	-

Table 132: Address mapping for 128M SDRAM (16Mx8, BRC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
11	11	22	-
10	10/AP	21	AP
9	9	20	10
8	8	19	9
7	7	18	8
6	6	17	7
5	5	16	6
4	4	15	5
3	3	14	4
2	2	13	3
1	1	12	2
0	0	11	**

Table 132: Address mapping for 128M SDRAM (16Mx8, BRC)

Table 133 shows the outputs for the memory controller and the corresponding inputs to the 256M SDRAM (16Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	23	23
13	BA0	24	24
12	12	22	-
11	11	21	-
10	10/AP	20	AP
9	9	19	-
8	8	18	9
7	7	17	8

Table 133: Address mapping for 256M SDRAM (16Mx16, BRC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
6	6	16	7
5	5	15	6
4	4	14	5
3	3	13	4
2	2	12	3
1	1	11	2
0	0	10	**

Table 133: Address mapping for 256M SDRAM (16Mx16, BRC)

Table 134 shows the outputs for the memory controller and the corresponding inputs to the 256M SDRAM (32Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
14	BA1	25	25
13	BA0	24	24
12	12	23	-
11	11	22	-
10	10/AP	21	AP
9	9	20	10
8	8	19	9
7	7	18	8
6	6	17	7
5	5	16	6
4	4	15	5
3	3	14	4
2	2	13	3

Table 134: Address mapping for 256M SDRAM (32Mx8, BRC)

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
1	1	12	2
0	0	11	**

Table 134: Address mapping for 256M SDRAM (32Mx8, BRC)

Table 135 shows the outputs from the memory controller and the corresponding inputs to the 512M SDRAM (32Mx16, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	25	25
13	BA0	24	24
12	12	23	-
11	11	22	-
10	10/AP	21	AP
9	9	20	10
8	8	19	9
7	7	18	8
6	6	17	7
5	5	16	6
4	4	15	5
3	3	14	4
2	2	13	3
1	1	12	2
0	0	11	**

Table 135: Address mapping for 512M SDRAM (32Mx16, BRC)

Table 136 shows the outputs from the memory controller and the corresponding inputs to the 512M SDRAM (64Mx8, pins 13 and 14 used as bank selects).

Output address (ADDROUT)	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
14	BA1	26	26
13	BA0	25	25
12	12	24	-
11	11	23	11
10	10/AP	22	AP
9	9	21	10
8	8	20	9
7	7	19	8
6	6	18	7
5	5	17	6
4	4	16	5
3	3	15	4
2	2	14	3
1	1	13	2
0	0	12	**

Table 136: Address mapping for 512M SDRAM (64Mx8, BRC)

Registers

The external memory is accessed using the AHB memory interface ports. Addresses are not fixed, but are determined by the AHB decoder and can be different for any particular system implementation. Transfers to the external memory controller memories are selected by the HSELMPMC[3:0]CS[7:0] signals (where [3:0] indicates the AHB port number and [7:0] indicates the chip select to be accessed.)

Register map

Table 137 lists the registers in the Memory Controller register map.

All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Address	Register	Description
A070 0000	Control register	Control register
A070 0004	Status register	Status register
A070 0008	Config register	Configuration register
A070 0020	DynamicControl	Dynamic Memory Control register
A070 0024	DynamicRefresh	Dynamic Memory Refresh Timer
A070 0028	DynamicReadConfig	Dynamic Memory Read Configuration register
A070 0030	DynamictRP	Dynamic Memory Precharge Command Period (t_{RP})
A070 0034	DynamictRAS	Dynamic Memory Active to Precharge Command Period (t_{RAS})
A070 0038	DynamictSREX	Dynamic Memory Self-Refresh Exit Time (t_{SREX})
A070 003C	DynamictAPR	Dynamic Memory Last Data Out to Active Time (t_{APR})
A070 0040	DynamictDAL	Dynamic Memory Data-in to Active Command Time (t_{DAL} or T_{APW})
A070 0044	DynamictWR	Dynamic Memory Write Recovery Time (t_{WR}, t_{DPL}, t_{RWL}, t_{RDL})

Table 137: Memory Controller register map

Address	Register	Description
A070 0048	DynamictRC	Dynamic Memory Active to Active Command Period (t_{RC})
A070 004C	DynamictRFC	Dynamic Memory Auto Refresh Period, and Auto Refresh to Active Command Period (t_{RFC})
A070 0050	DynamictXSR	Dynamic Memory Exit Self-Refresh to Active Command (t_{XSR})
A070 0054	DynamictRRD	Dynamic Memory Active Bank A to Active B Time (t_{RRD})
A070 0058	DynamictMRD	Dynamic Memory Load Mode register to Active Command Time (t_{MRD})
A070 0080	StaticExtendedWait	Static Memory Extended Wait
A070 0100	DynamicConfig0	Dynamic Memory Configuration Register 0
A070 0104	DynamicRasCas0	Dynamic Memory RAS and CAS Delay 0
A070 0120	DynamicConfig1	Dynamic Memory Configuration Register 1
A070 0124	DynamicRasCas1	Dynamic Memory RAS and CAS Delay 1
A070 0140	DynamicConfig2	Dynamic Memory Configuration Register 2
A070 0144	DynamicRasCas2	Dynamic Memory RAS and CAS Delay 2
A070 0160	DynamicConfig3	Dynamic Memory Configuration Register 3
A070 0164	DynamicRasCas3	Dynamic Memory RAS and CAS Delay 3
A070 0200	StaticConfig0	Static Memory Configuration Register 0
A070 0204	StaticWaitWen0	Static Memory Write Enable Delay 0
A070 0208	StaticWaitOen0	Static Memory Output Enable Delay 0
A070 020C	StaticWaitRd0	Static Memory Read Delay 0
A070 0210	StaticWaitPage0	Static Memory Page Mode Read Delay 0
A070 0214	StaticWaitWr0	Static Memory Write Delay 0
A070 0218	StaticWaitTurn0	Static Memory Turn Round Delay 0
A070 0220	StaticConfig1	Static Memory Configuration Register 1
A070 0224	StaticWaitWen1	Static Memory Write Enable Delay 1
A070 0228	StaticWaitOen1	Static Memory Output Enable Delay 1

Table 137: Memory Controller register map

Address	Register	Description
A070 022C	StaticWaitRd1	Static Memory Read Delay 1
A070 0230	StaticWaitPage1	Static Memory Page Mode Read Delay 1
A070 0234	StaticWaitWr1	Static Memory Write Delay 1
A070 0238	StaticWaitTurn1	Static Memory Turn Round Delay 1
A070 0240	StaticConfig2	Static Memory Configuration Register 2
A070 0244	StaticWaitWen2	Static Memory Write Enable Delay 2
A070 0248	StaticWaitOen2	Static Memory Output Enable Delay 2
A070 024C	StaticWaitRd2	Static Memory Read Delay 2
A070 0250	StaticWaitPage2	Static Memory Page Mode Read Delay 2
A070 0254	StaticWaitWr2	Static Memory Write Delay 2
A070 0258	StaticWaitTurn2	Static Memory Turn Round Delay 2
A070 0260	StaticConfig3	Static Memory Configuration Register 3
A070 0264	StaticWaitWen3	Static Memory Write Enable Delay 3
A070 0268	StaticWaitOen3	Static Memory Output Enable Delay 3
A070 026C	StaticWaitRd3	Static memory Read Delay 3
A070 0270	StaticWaitPage3	Static Memory Page Mode Read Delay 3
A070 0274	StaticWaitWr3	Static Memory Write Delay 3
A070 0278	StaticWaitTurn3	Static Memory Turn Round Delay 3

Table 137: Memory Controller register map

Reset values

Reset values will be noted as appropriate in the Description column of each register table, rather than as a separate column.

Control register

Address: A070 0000

The Control register controls the memory controller operation. The control bits can be changed during normal operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													LPM	ADDM	MCEN

Register bit assignment

Bits	Access	Mnemonic	Description
D31:03	N/A	Reserved	N/A (do not modify)
D02	R/W	LPM	**Low-power mode** 0 Normal mode (reset value on reset_n and HRESETn) 1 Low-power mode Indicates normal or low-power mode. Entering low-power mode reduces memory controller power consumption. Dynamic memory is refreshed as necessary. The memory controller returns to normal functional mode by clearing the low-power mode bit, by AHB, or by power-on reset. If you modify this bit, be sure the memory controller is in idle state. If you modify the L bit, be aware of these conditions: ■ The external memory cannot be accessed in low-power or disabled state. If a memory access is performed in either of these states, an error response is generated. ■ The memory controller AHB programming port can be accessed normally. ■ The memory controller registers can be programmed in low-power and/or disabled state.

Table 138: Control register

Bits	Access	Mnemonic	Description
D01	R/W	ADDM	**Address mirror** 0 Normal memory map 1 Reset memory map. Static memory chip select 1 is mirrored onto chip select 0 and chip select 4 (reset value on reset_n) Indicates normal or reset memory map. On power-on reset, chip select 1 is mirrored to both chip select 0 and chip select 1/chip select 4 memory areas. Clearing the M bit allows chip select 0 and chip select 4 memory to be accessed.
D00	R/W	MCEN	**Memory controller enable** 0 Disabled 1 Enabled (reset value on reset_n and HRESETn) Disabling the memory controller reduces power consumption. When the memory controller is disabled, the memory is not refreshed. The memory controller is enabled by setting the enable bit, by AHB, or by power-on reset. If you modify this bit, be sure the memory controller is in idle state. If you modify the E bit, be aware of these conditions: ■ The external memory cannot be accessed in low-power or disabled state. If a memory access is performed in either of these states, an error response is generated. ■ The memory controller AHB programming port can be accessed normally. ■ The memory controller registers can be programmed in low-power and/or disabled state.

Table 138: Control register

Status register

Address: A070 0004

The Status register provides memory controller status information.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													SA	WBS	BUSY

Register bit assignment

Bits	Access	Mnemonic	Description
D31:03	N/A	Reserved	N/A (do not modify)
D02	R	SA	**Self-refresh acknowledge (SREFACK)** 0 Normal mode 1 Self refresh mode (reset value on reset_n) Indicates the memory controller operating mode.
D01	R	WBS	**Write buffer status** 0 Write buffers empty (reset value on reset_n) 1 Write buffers contain data Enables the memory controller to enter low-power mode or disabled mode clearly.
D00	R	BUSY	**Busy** 0 Memory controller is idle (reset value on HRESETn) 1 Memory controller is busy performing memory transactions, commands, or auto-refresh cycles, or is in self-refresh mode (reset value on reset_n and HRESETn) Ensures that the memory controller enters the low-power or disabled state cleanly by determining whether the memory controller is busy.

Table 139: Status register

Configuration register

Address: A070 0008

The Configuration register configures memory controller operation. It is recommended that this register be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CLK	Reserved							END

Register bit assignment

Bits	Access	Mnemonic	Description
D31:09	N/A	Reserved	N/A (do not modify)
D08	R/W	CLK	**Clock ratio (HCLK:clk-out[3:0]) ratio** 0 1:1 (reset value on reset_n) 1 1:2
D07:01	N/A	Reserved	N/A (do not modify)
D00	R/W	END	Endian mode 0 Little endian mode 1 Big endian mode The value of the endian bit on power-on reset (reset_n) is determined by the gpio[44] signal. This value can be overridden by software. This field is not affected by the AHB reset (HRESETn). **Note:** The value of the gpio[44] signal is reflected in this field. When programmed, this register reflects the last value written into the register. You must flush all data in the memory controller before switching between little endian and big endian modes.

Table 140: Configuration register

Dynamic Memory Control register

Address: A070 0020

The Dynamic Memory Control register controls dynamic memory operation. The control bits can be changed during normal operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd	nRP	Not used	Reserved				SDRAMInit		Rsvd	Not used	Reserved		SR	Not used	CE

Register bit assignment

Bits	Access	Mnemonic	Description
D31:15	N/A	Reserved	N/A (do not modify)
D14	R/W	nRP	**Sync/Flash reset/power down signal (dy_pwr_n)** 0 dy_pwr_n signal low (reset value on reset_n) 1 Set dy_pwr_n signal high
D13	R/W	Not used	**Low-power SDRAM deep-sleep mode** 0 Normal operation (reset value on reset_n) 1 Enter deep power down mode
D12:09	N/A	Reserved	N/A (do not modify)
D08:07	R/W	SDRAMInit	**SDRAM initialization** 00 Issue SDRAM NORMAL operation command (reset value on reset_n) 01 Issue SDRAM MODE command 10 Issue SDRAM PALL (precharge all) command 11 Issue SDRAM NOP (no operation) command
D06	N/A	Reserved	N/A (do not modify)
D05	R/W	Not used	Must write 0.
D04:03	N/A	Reserved	N/A (do not modify)

Table 141: Dynamic Memory Control register

Bits	Access	Mnemonic	Description
D02	R/W	SR	**Self-refresh request (SREFREQ)** 0 Normal mode 1 Enter self-refresh mode (reset value on reset_n) By writing 1 to this bit, self-refresh can be entered under software control. Writing 0 to this bit returns the memory controller to normal mode. The self-refresh acknowledge bit in the Status register (see page 207) must be polled to discover the current operating mode of the memory controller. **Note:** The memory controller exits from power-on reset with the self-refresh bit on high. To enter normal functional mode, set the self-refresh bit low. Writing to this register with the bit set to high places the register into self-refresh mode. This functionality allows data to be stored over SDRAM self-refresh of the ASIC is powered down.
D01	R/W	Not used	Must write 1.
D00	R/W	CE	**Dynamic memory clock enable** 0 Clock enable if idle devices are deasserted to save power (reset value on reset_n) 1 All clock enables are driven high continuously. **Note:** Clock enable must be high during SDRAM initialization.

Table 141: Dynamic Memory Control register

Dynamic Memory Refresh Timer register

Address: A070 0024

The Dynamic Memory Refresh Timer register configures dynamic memory operation. It is recommended that this register be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode.These bits can, however, be changed during normal operation if necessary.

Note: The Dynamic Memory Refresh Timer register is used for all four dynamic memory chip selects. The worst case value for all chip selects must be programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					REFRESH										

Register bit assignment

Bits	Access	Mnemonic	Description
D31:11	N/A	Reserved	N/A (do not modify)
D10:0	R/W	REFRESH	**Refresh timer** 0x0 Refresh disabled (reset value on reset_n) 0x1–0x77F n(x16) 16n HCLK ticks between SDRAM refresh cycles

Table 142: Dynamic Memory Refresh Timer register

Examples

Generic formula: DynamicRefresh = $(((t_{REF}$ / #rows) * speed grade) / 32)

For 4k rows:

Refresh period = 64µs

Speed grade = 200 MHz

Calculation = $((64e^{-3} / 4096) * 200e^{+6}) / 32$ = 97 = 0x61

For 8k rows:

Refresh period = 64µs

Speed grade = 150 MHz

Calculation = $((64e^{-3} / 8192) * 150e^{+6}) / 32$ = 36 = 0x24

Notes:

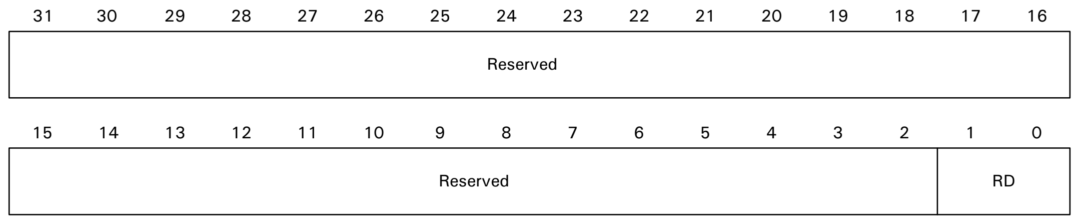
- The refresh cycles are evenly distributed. There might be slight variations, however, when the auto-refresh command is issued, depending on the status of the memory controller.
- Unlike other SDRAM memory timing parameters, the refresh period is programmed in the HCLK domain.

Dynamic Memory Read Configuration register

Address: A070 0028

The Dynamic Memory Read Configuration register allows you to configure the dynamic memory read strategy. Modify this register only during system initialization.

Note: The Dynamic Memory Read Configuration register is used for all four dynamic memory chip selects. The worst case value for all chip selects must be programmed.



Register bit assignment

Bits	Access	Mnemonic	Description
D31:02	N/A	Reserved	N/A (do not modify)
D01:00	RW	RD	**Read data strategy** 00 Reserved. 01 Command delayed strategy, using CLKDELAY (command delayed, clock out not delayed). 10 Command delayed strategy plus one clock cycle, using CLKDELAY (command delayed, clock out not delayed). 11 Command delayed strategy plus two clock cycles, using CLKDELAY (command delayed, clock out not delayed).

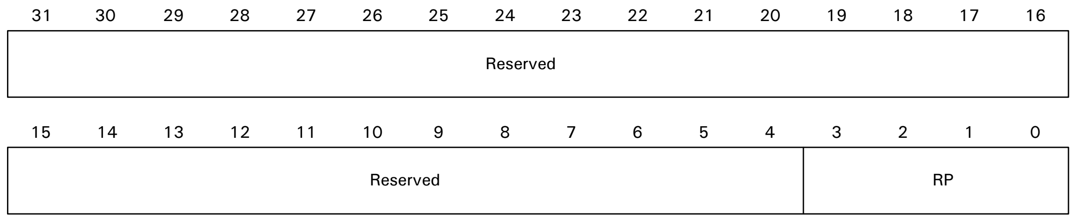
Table 143: Dynamic Memory Read Configuration register

Dynamic Memory Precharge Command Period register

Address: A070 0030

The Dynamic Memory Precharge Command Period register allows you to program the precharge command period, t_{RP}. Modify this register only during system initialization. This value normally is found in SDRAM datasheets as t_{RP}.

Note: The Dynamic Memory Precharge Command Period register is used for all four dynamic memory chip selects. The worst case value for all chip selects must be programmed.



Register bit assignment

Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	RP	**Precharge command period (t_{RP})** 0x0–0xE n+1 clock cycles, where the delay is in CLK cycles. **0xF** 16 clock cycles (reset value on reset_n)

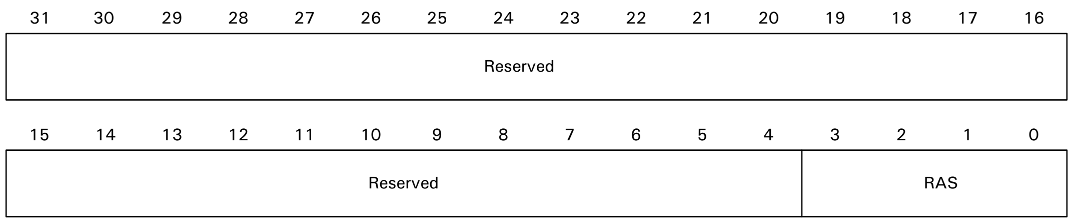
Table 144: Dynamic Memory Precharge Command Period register

Dynamic Memory Active to Precharge Command Period register

Address: A070 0034

The Dynamic Memory Active to Precharge Command Period register allows you to program the active to precharge command period, t_{RAS}. It is recommended that this register be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode. This value normally is found in SDRAM datasheets as t_{RAS}.

Note: The Dynamic Memory Active to Precharge Command Period register is used for all four dynamic memory chip selects. The worst case value for all chip selects must be programmed.



Register bit assignment

Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	RAS	**Active to precharge command period (t_{RAS})** 0x0–0xE n+1 clock cycles, where the delay is in CLK cycles. **0xF** 16 clock cycles (reset value on reset_n)

Table 145: Dynamic Memory Active to Precharge Command Period register

Dynamic Memory Self-refresh Exit Time register

Address: A070 0038

The Dynamic Memory Self-refresh Exit Time register allows you to program the self-refresh exit time, t_{SREX}. It is recommended that this register be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode. This value normally is found in SDRAM data sheets as t_{SREX}.

Note: The Dynamic Memory Self-refresh Exit Time register is used for all four dynamic memory chip selects. The worst case value for all chip selects must be programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												SREX			

Register bit assignment

Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	SREX	**Self-refresh exit time (t_{SREX})** 0x0–0xE n+1 clock cycles, where the delay is in CLK cycles. **0xF** 16 clock cycles (reset value on reset_n)

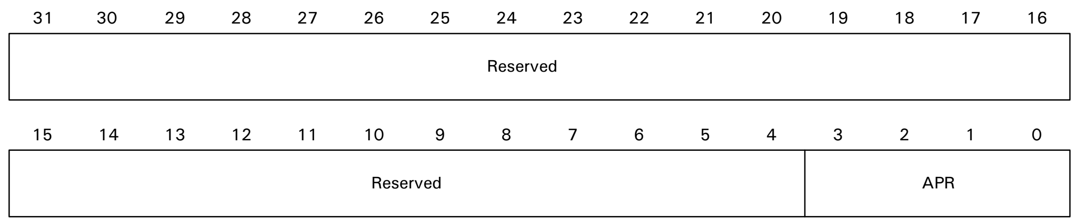
Table 146: Dynamic Memory Self-refresh Exit Time register

Dynamic Memory Last Data Out to Active Time register

Address: A070 003C

The Dynamic Memory Last Data Out to Active Time register allows you to program the last-data-out to active command time, t_{APR}. It is recommended that this register be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode. This value normally is found in SDRAM datasheets as t_{APR}.

Note: The Dynamic Memory Last Data Out to Active Time register is used for all four dynamic memory chip selects. The worst case value for all chip selects must be programmed.



Register bit assignment

Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	APR	**Last-data-out to active command time (t_{APR})** 0x0–0xE n+1 clock cycles, where the delay is in CLK cycles. **0xF** 16 clock cycles (reset value on reset_n)

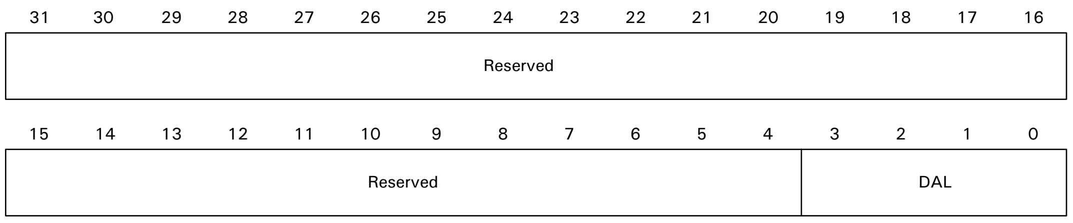
Table 147: Dynamic Memory Last Data Out to Active Time register

Dynamic Memory Data-in to Active Command Time register

Address: A070 0040

The Dynamic Memory Data-in to Active Command Time register allows you to program the data-in to active command time, t_{DAL}. It is recommended that this register be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode. This value normally is found in SDRAM data sheets as t_{DAL} or t_{APW}.

Note: The Dynamic Memory Data-in Active Command Time register is used for all four dynamic memory chip selects. The worst case value for all chip selects must be programmed.



Register bit assignment

Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	DAL	**Data-in to active command (t_{DAL} or t_{APW})** 0x0–0xE n+1 clock cycles, where the delay is in CLK cycles. **0xF** 15 clock cycles (reset value on reset_n)

Table 148: Dynamic Memory Data-in Active Command Time register

Dynamic Memory Write Recovery Time register

Address: A070 0044

The Dynamic Memory Write Recovery Time register allows you to program the write recovery time, t_{WR}. It is recommended that this register be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode. This value normally is found in SDRAM datasheets as t_{WR}, t_{DPL}, t_{RWL}, or t_{RDL}.

Note: The Dynamic Memory Write Recovery Time register is used for all four dynamic memory chip selects. The worst case value for all chip selects must be programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												WR			

Register bit assignment

Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	WR	Write recovery time (t_{WR}, t_{DPL}, t_{RWL}, or t_{RDL}) 0x0–0xE n+1 clock cycles, where the delay is in CLK cycles. **0xF** 16 clock cycles (reset value on reset_n)

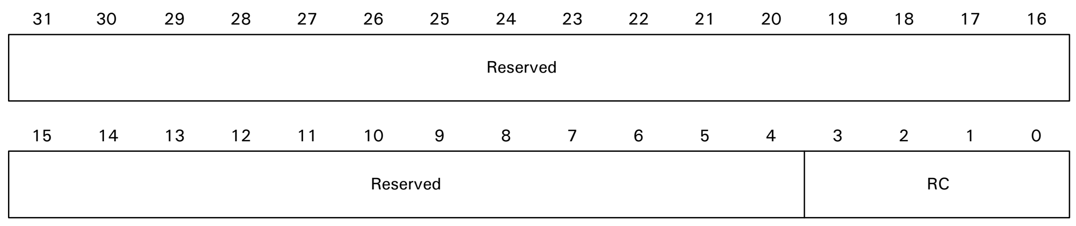
Table 149: Dynamic Memory Write Recovery TIme register

Dynamic Memory Active to Active Command Period register

Address: A070 0048

The Dynamic Memory Active to Active Command Period register allows you to program the active to active command period, t_{RC}. It is recommended that this register be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode. This value normally is found in SDRAM datasheets as t_{RC}.

Note: The Dynamic Memory Active to Active Command period register is used for all four dynamic memory chip selects. The worst case value for all chip selects must be programmed.



Register bit assignment

Bits	Access	Mnemonic	Description
D31:05	N/A	Reserved	N/A (do not modify)
D04:00	R/W	RC	**Active to active command period (t_{RC})** 0x0–0x1E n+1 clock cycles, where the delay is in CLK cycles. **0x1F** 32 clock cycles (reset value on reset_n)

Table 150: Dynamic Memory Active to Active Command Period register

Dynamic Memory Auto Refresh Period register

Address: A070 004C

The Dynamic Memory Auto Refresh Period register allows you to program the auto-refresh period and the auto-refresh to active command period, t_{RFC}. It is recommended that this register be modified during initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode. This value normally is found in SDRAM datasheets as t_{RFC} or t_{RC}.

Note: The Dynamic Memory Auto Refresh Period register is used for all four dynamic memory chip selects. The worst case value for all chip selects must be programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											RFC				

Register bit assignment

Bits	Access	Mnemonic	Description
D31:05	N/A	Reserved	N/A (do not modify)
D04:00	R/W	RFC	**Auto-refresh period and auto-refresh to active command period** 0x0–0x1E n+1 clock cycles, where the delay is in CLK cycles. **0x1F** 32 clock cycles (reset value on reset_n)

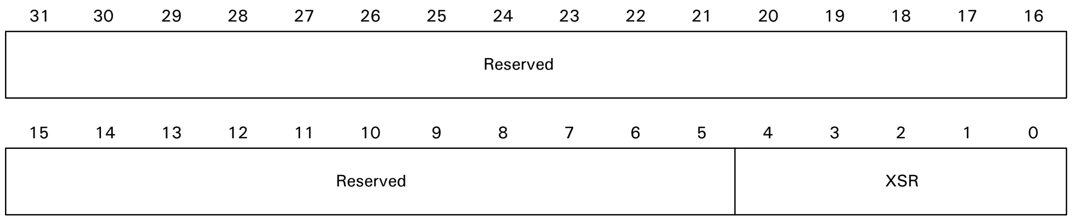
Table 151: Dynamic Memory Auto Refresh Period register

Dynamic Memory Exit Self-refresh register

Address: A070 0050

The Dynamic memory Exit Self-refresh register allows you to program the exit self-refresh to active command time, t_{XSR}. It is recommended that this register be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode. This value normally is found in SDRAM datasheets as t_{XSR}.

Note: The Dynamic Memory Exit Self-refresh register is used for all four dynamic memory chip selects. The worst case value for all the chip selects must be programmed.



Register bit assignment

Bits	Access	Mnemonic	Description
D31:05	N/A	Reserved	N/A (do not modify)
D04:00	R/W	XSR	**Exit self-refresh to active time command** 0x0–0x1E n+1 clock cycles, where the delay is in CLK cycles. **0x1F** 32 clock cycles (reset value on reset_n)

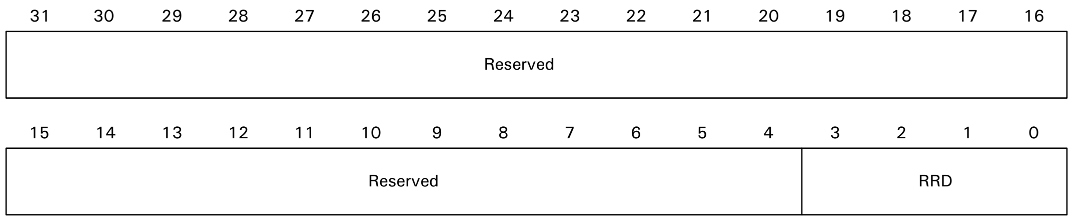
Table 152: Dynamic Memory Exit Self-refresh register

Dynamic Memory Active Bank A to Active Bank B Time register

Address: A070 0054

The Dynamic Memory Active Bank A to Active Bank B Time register allows you to program the active bank A to active bank B latency, t_{RRD}. It is recommended that this register be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode. This value normally is found in SDRAM datasheets as t_{RRD}.

Note: The Dynamic Memory Active Bank A to Active Bank B Time register is used for all four dynamic memory chip selects. The worst case value for all chip selects must be programmed.



Register bit assignment

Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	RRD	**Active bank A to Active bank B latency** 0x0–0xE n+1 clock cycles, where the delay is in CLK cycles 0xF 16 clock cycles (reset on reset_n)

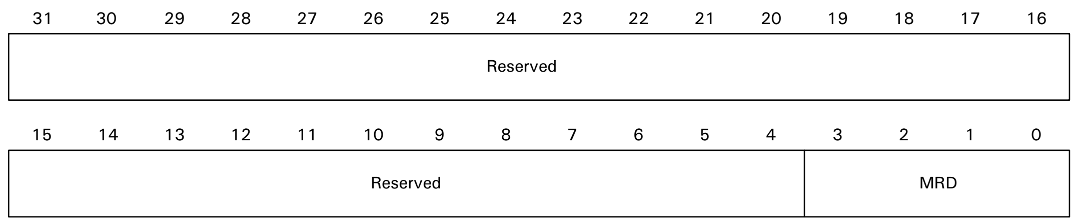
Table 153: Dynamic Memory Active Bank A to Active Bank B Time register

Dynamic Memory Load Mode register to Active Command Time register

Address: A070 0058

The Dynamic Memory Load Mode register to Active Command Time register allows you to program the Load Mode register to active command time, t_{MRD}. It is recommended that this register be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode. This value normally is found in SDRAM datasheets as t_{MRD} or t_{RSA}.

Note: The Dynamic Memory Load Mode register to Active Command Time register is used for all four chip selects. The worst case value for all chip selects must be programmed.



Register bit assignment

Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	MRD	**Load Mode register to active command time** 0x0–0xE n+1 clock cycles, where the delay is in CLK cycles 0xF 16 clock cycles (reset on reset_n)

Table 154: Dynamic Memory Load Mode register to Active Command Time register

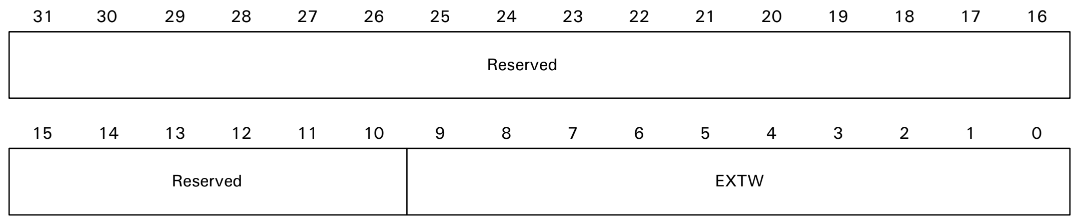
Static Memory Extended Wait register

Address: A070 0080

The Static Memory Extended Wait register times long static memory read and write transfers (which are longer than can be supported by the Static Memory Read Delay registers (see page 236) or the Static Memory Write Delay registers (see page 238)) when the EW (extended wait) bit in the related Static Memory Configuration register (see page 230) is enabled.

There is only one Static Memory Extended Wait register, which is used by the relevant static memory chip select if the appropriate EW bit is set in the Static Memory Configuration register.

It is recommended that this register be modified during system initialization, or when there are no current or outstanding transactions. If necessary, however, these control bits can be changed during normal operation.



Register bit assignment

Bits	Access	Mnemonic	Description
D31:10	N/A	Reserved	N/A (do not modify)
D09:00	R/W	EXTW	**External wait timeout** 0x0 16 clock cycles, where the delay is in HCLK cycles 0x1–0x3FF (n=1) x16 clock cycles

Table 155: Static Memory Extended Wait register

Example

Static memory read/write time = 16 us

CLK frequency = 50 MHz

This value must be programmed into the Static Memory Extended Wait register:

(16 x 10^{-6} x 50 x 10^{6} / 16) - 1 = 49

Dynamic Memory Configuration 0–3 registers

Address: A070 0100 / 0120 / 0140 / 0160

The Dynamic Memory Configuration 0–3 registers allow you to program the configuration information for the relevant dynamic memory chip select. These registers are usually modified only during system initialization.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											Protect	BDMC	Reserved		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd	AM	Rsvd	AM1						Reserved		MD		Reserved		

Register bit assignment

Bits	Access	Mnemonic	Description
D31:21	N/A	Reserved	N/A (do not modify)
D20	R/W	Protect	**Write protect** 0 Writes not protected (reset value on reset_n) 1 Write protected
D19	R/W	BDMC	**Buffer enable** 0 Buffer disabled for accesses to this chip select (reset value on reset_n) 1 Buffer enabled for accesses to this chip select. The buffers must be disabled during SDRAM initialization. The buffers must be enabled during normal operation.
D18:15	N/A	Reserved	N/A (do not modify)

Table 156: Dynamic Memory Configuration 0–3 registers

Bits	Access	Mnemonic	Description
D14	R/W	AM	**Address mapping** 0 Reset value on reset_n See Table 157, “Address mapping,” on page 226 for more information.
D13	N/A	Reserved	N/A (do not modify)
D12:07	R/W	AM1	**Address mapping** 00000000 Reset value on reset_n The SDRAM column and row width and number of banks are computed automatically from the address mapping. See Table 157, “Address mapping,” on page 226 for more information.
D06:05	N/A	Reserved	N/A (do not modify)
D04:03	R/W	MD	**Memory device** 00 SDRAM (reset value on reset_n) 01 Low-power SDRAM 10 Reserved 11 Reserved
D02:00	N/A	Reserved	N/A (do not modify)

Table 156: Dynamic Memory Configuration 0–3 registers

Table 157 shows address mapping for the Dynamic Memory Configuration 0-3 registers. Address mappings that are not shown in the table are reserved.

[14]	[12]	[11:9]	[8:7]	Description
16-bit external bus high-performance address mapping (row, bank column)				
0	0	000	00	16 Mb (2Mx8), 2 banks, row length=11, column length=9
0	0	000	01	16 Mb (1Mx16), 2 banks, row length=11, column length=8
0	0	001	00	64 Mb (8Mx80, 4 banks, row length=12, column length=9
0	0	001	01	64 Mb (4Mx16), 4 banks, row length=12, column length=8
0	0	010	00	128 Mb (16Mx8), 4 banks, row length=12, column length=10
0	0	010	01	128 Mb (8Mx16), 4 banks, row length=12, column length=9

Table 157: Address mapping

[14]	[12]	[11:9]	[8:7]	Description
0	0	011	00	256 Mb (32Mx8), 4 banks, row length=13, column length=10
0	0	011	01	256 Mb (16Mx16), 4 banks, row length=13, column length=9
0	0	100	00	512 Mb (64Mx8), 4 banks, row length=13, column length=11
0	0	100	01	512 Mb (32Mx16), 4 banks, row length=13, column length=10
16-bit external bus low-power SDRAM address mapping (bank, row, column)				
0	1	000	00	16 Mb (2Mx8), 2 banks, row length=11, column length=9
0	1	000	01	16 Mb (1Mx16), 2 banks, row length=11, column length=8
0	1	001	00	64 Mb (8Mx8), 4 banks, row length 12, column length=9
0	1	001	01	64 Mb (4Mx16), 4 banks, row length=12, column length=8
0	1	010	00	128 Mb (16Mx8), 4 banks, row length=12, column length=10
0	1	010	01	128 Mb (8Mx16), 4 banks, row length=12, column length=9
0	1	011	00	256 Mb (32Mx8), 4 banks, row length=13, column length=10
0	1	011	01	256 Mb (16Mx16), 4 banks, row length=13, column length=9
0	1	100	00	512 Mb (64Mx8), 4 banks, row length=13, column length=11
0	1	100	01	512 Mb (32Mx16, 4 banks, row length=13, column length=10
32-bit extended bus high-performance address mapping (row, bank, column)				
1	0	000	00	16 Mb (2Mx8), 2 banks, row length=11, column length=9
1	0	000	01	16 Mb (1Mx16), 2 banks, row length=11, column length=8
1	0	001	00	64 Mb (8Mx8), 4 banks, row length=12, column length=9
1	0	001	01	64 Mb (4Mx16), 4 banks, row length=12, column length=8
1	0	001	10	64 Mb (2Mx32), 4 banks, row length=11, column length=8
1	0	010	00	128 Mb (16Mx8), 4 banks, row length=12, column length=10
1	0	010	01	128 Mb (8Mx16), 4 banks, row length=12, column length=9
1	0	010	10	128 Mb (4Mx32), 4 banks, row length=12, column length=8
1	0	011	00	256 Mb (32Mx8), 4 banks, row length=13, column length=10
1	0	011	01	256 Mb (16Mx16), 4 banks, row length=13, column length=9
1	0	011	10	256 Mb (8Mx32), 4 banks, row length=13, column length=8

Table 157: Address mapping

[14]	[12]	[11:9]	[8:7]	Description
1	0	100	00	512 Mb (64Mx8), 4 banks, row length=13, column length=11
1	0	100	01	512 Mb (32Mx16), 4 banks, row length=13, column length=10
32-bit extended bus low-power SDRAM address mapping (bank, row, column)				
1	1	000	00	16 Mb (2Mx8), 2 banks, row length=11, column length=9
1	1	000	01	16 Mb (1Mx16), 2 banks, row length=11, column length=8
1	1	001	00	64 Mb (8Mx8), 4 banks, row length=12, column length=9
1	1	001	01	64 MB (4Mx16), 4 banks, row length=12, column length=8
1	1	001	10	64 Mb (2Mx32), 4 banks, row length=11, column length=8
1	1	010	00	128 Mb (16Mx8), 4 banks, row length=12, column length=10
1	1	010	01	128 Mb (8Mx16), 4 banks, row length=12, column length=9
1	1	010	10	128 Mb (4Mx32), 4 banks, row length=12, column length=8
1	1	011	00	256 Mb (32Mx8), 4 banks, row length=13, column length=10
1	1	011	01	256 Mb (16Mx16), 4 banks, row length=13, column length=9
1	1	011	10	256 Mb (8Mx32), 4 banks, row length=13, column length=8
1	1	100	00	512 Mb (64Mx8), 4 banks, row length=13, column length=11
1	1	100	01	512 Mb (32Mx16), 4 banks, row length=13, column length=10

Table 157: Address mapping

A chip select can be connected to a single memory device; in this situation, the chip select data bus width is the same as the device width. As an alternative, the chip select can be connected to a number of external devices. In this situation, the chip select data bus width is the sum of the memory device databus widths.

Examples

For a chip select connected to	Select this mapping
32-bit wide memory device	32-bit wide address mapping
16-bit wide memory device	16-bit wide address mapping
4 x 8-bit wide memory devices	32-bit wide address mapping
2 x 8-bit memory devices	16-bit wide address mapping

Dynamic Memory RAS and CAS Delay 0–3 registers

Address: A070 0104 / 0124 / 0144 / 0164

The Dynamic Memory RAS and CAS Delay 0–3 registers allow you to program the RAS and CAS latencies for the relevant dynamic memory. It is recommended that these registers be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode.

Note: The values programmed into these registers must be consistent with the values used to initialize the SDRAM memory device.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CAS		Reserved						RAS	

Register bit assignment

Bits	Access	Mnemonic	Description
D31:10	N/A	Reserved	N/A (do not modify)
D09:08	R/W	CAS	**CAS latency** 00 Reserved 01 One clock cycle, where the RAS to CAS latency (RAS) and CAS latency (CAS) are defined in CLK cycles 10 Two clock cycles 11 Three clock cycles (reset value on reset_n)
D07:02	N/A	Reserved	N/A (do not modify)
D01:00	R/W	RAS	**RAS latency (active to read.write delay)** 00 Reserved 01 One clock cycle, where the RAS to CAS latency (RAS) and CAS latency (CAS) are defined in CLK cycles 10 Two clock cycles 11 Three clock cycles (reset value on reset_n)

Table 158: Dynamic Memory RAS and CAS Delay 0–3 registers

Static Memory Configuration 0–3 registers

Address: A070 0200 / 0220 / 0240 / 0260

The Static Memory Configuration 0–3 registers configure the static memory configuration. It is recommended that these registers be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											PSMC	BSMC	Reserved		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							EW	PB	PC	Reserved		PM	Rsvd	MW	

Register bit assignment

Bits	Access	Mnemonic	Description
D31:21	N/A	Reserved	N/A (do not modify)
D20	R/W	PSMC	**Write protect** 0 Writes not protected (reset value on reset_n) 1 Write protected
D19	R/W	BSMC	**Buffer enable** 0 Write buffer disabled (reset value on reset_n) 1 Write buffer enabled **Note:** This field must always be set to 0 when a peripheral other than SRAM is attached to the static ram chip select.
D18:09	N/A	Reserved	N/A (do not modify)

Table 159: Static Memory Configuration 0–3 registers

Bits	Access	Mnemonic	Description
D08	R/W	EW	**Extended wait** 0 Extended wait disabled (reset value on reset_n) Extended wait enabled Extended wait uses the Static Extended Wait register (see page 224) to time both the read and write transfers, rather than the Static Memory Read Delay 0–3 registers (see page 236) and Static Memory Write Delay 0–3 registers (see page 238). This allows much longer transactions. Extended wait also can be used with the ta_strb signal to allow a slow peripheral to terminate the access. In this case, the Static Memory Extended Wait register (see page 224) can be programmed with the maximum timeout limit. A high value on ta_strb is then used to terminate the access before the maximum timeout occurs. **Note:** Extended wait and page mode cannot be selected simultaneously.

Table 159: Static Memory Configuration 0–3 registers

Bits	Access	Mnemonic	Description
D07	R/W	PB	**Byte lane state** 0 For reads, all bits in byte_lane_sel_n[3:0] are high. For writes, the respective active bits in byte_lane_sel_n[3:0] are low (reset value for chip select 0, 2, and 3 on reset_n). 1 For reads, the respective active bits in byte_lane_sel_n[3:0] are low. For writes, the respective active bits in byte_lane_sel_n[3:0] are low. **Note:** Setting this bit to 0 disables the write enable signal. WE_n will always be set to 1 (that is, you must use byte lane select signals). The value of the chip select 1 byte lane state field on power-on reset (reset_n) is determined by the boot_strap[0] signal. This value can be overridden by software. This field is not affected by AHB reset (HRESETn). The byte lane state bit (PB) enables different types of memory to be connected. For byte-wide static memories, the byte_lane_sel_n[3:0] signal from the memory controller is usually connected to WE_n (write enable). In this case, for reads, all byte_lane_sel_n[3:0] bits must be high, which means that the byte lane state bit must be low. 16-bit wide static memory devices usually have the byte_lane_sel_n[3:0] signals connected to the nUB and nLB (upper byte and lower byte) signals in the static memory. In this case, a write to a particular byte must assert the appropriate nUB or nLB signal low. For reads, all nUB and nLB signals must be asserted low so the bus is driven. In this case, the byte lane state must be high. **Note:** For chip select 1, the value of the boot-strap[0] signal is reflected in this field. When programmed, this register reflects the last value written into it.
D06	R/W	PC	**Chip select polarity** 0 Active low chip select 1 Active high chip select The value of the chip select polarity on power-on reset (reset_n) for chip select 1 is determined by the gpio[49] signal. This value can be overridden by software. This field is not affected by AHB reset (HRESETn).
D05:04	N/A	Reserved	N/A (do not modify)

Table 159: Static Memory Configuration 0–3 registers

Bits	Access	Mnemonic	Description
D03	R/W	PM	**Page mode** 0 Disabled (reset on reset_n) 1 Async page mode enabled (page length four) In page mode, the memory controller can burst up to four external accesses. Devices with asynchronous page mode burst four or higher are supported. Asynchronous page mode burst two devices are not supported and must be accessed normally.
D02	N/A	Reserved	N/A (do not modify)
D01:00	R/W	MW	**Memory width** 00 8 bit (reset value for chip select 0, 2, and 3 on reset_n) 01 16 bit 10 32 bit 11 Reserved The value of the chip select 1 memory width field on power-on reset (reset_n) is determined by the boot_strap[4:3] signal. This value can be overridden by software. This field is not affected by AHB reset (HRESETn). **Note:** For chip select 1, the value of the boot_strap[4:3] signal is reflected in this field. When programmed, this register reflects the last value written into it.

Table 159: Static Memory Configuration 0–3 registers

Note: Synchronous burst mode memory devices are not supported.

Static Memory Write Enable Delay 0–3 registers

Address: A070 0204 / 0224 / 0244 / 0264

The Static Memory Write Enable Delay 0–3 registers allow you to program the delay from the chip select to the write enable assertion. The Static Memory Write Enable Delay register is used in conjunction with the Static Memory Write Delay registers, to control the width of the write enable signals. It is recommended that these registers be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												WWEN			

Register bit assignment

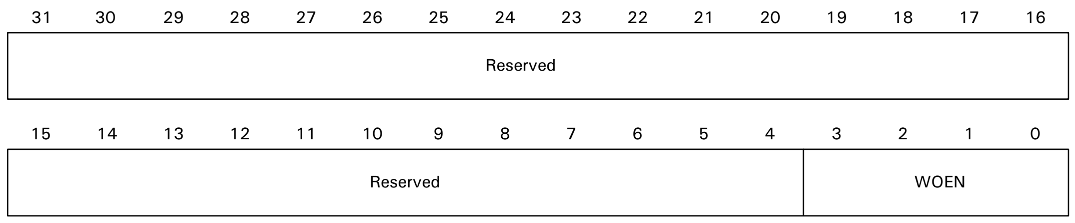
Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	WWEN	**Wait write enable (WAITWEN)** 0000 One HCLK cycle delay between assertion of chip select and write enable (reset value on reset_n). 0001–1111 (n+1) HCLK cycle delay, where the delay is (WAITWEN+1) x t_{HCLK} Delay from chip select assertion to write enable.

Table 160: Static Memory Write Enable Delay 0–3 registers

Static Memory Output Enable Delay 0–3 registers

Address: A070 0208 / 0228 / 0248 / 0268

The Static Memory Output Enable Delay 0–3 registers allow you to program the delay from the chip select or address change, whichever is later, to the output enable assertion. The Static Memory Output Enable Delay register is used in conjunction with the Static Memory Read Delay registers, to control the width of the output enable signals. It is recommended that these registers be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode.



Register bit assignment

Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	WOEN	**Wait output enable (WAITOEN)** 0000 No delay (reset value on reset_n). 0001–1111 n cycle delay, where the delay is WAITOEN x t_{HCLK} Delay from chip select assertion to output enable.

Table 161: Static Memory Output Enable Delay 0–3 registers

Static Memory Read Delay 0–3 registers

Address: A070 020C / 022C / 024C / 026C

The Static Memory Read Delay 0–3 registers allow you to program the delay from the chip select to the read access. It is recommended that these registers be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode. These registers are not used if the extended wait bit is set in the related Static Memory Configuration register (see page 230).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											WTRD				

Register bit assignment

Bits	Access	Mnemonic	Description
D31:05	N/A	Reserved	N/A (do not modify)
D04:00	R/W	WTRD	Nonpage mode read wait states or asynchronous page mode read first access wait state (WAITRD) 00000–11110 (n+1) HCLK cycle for read accesses. For nonsequential reads, the wait state time is (WAITRD+1) x t_{HCLK} 11111 32 HCLK cycles for read accesses (reset value on reset_n) Use this equation to compute this field: WTRD = ([T_b + T_a + 10.0] / T_c) - 1 T_b = Total board propagation delay, including any buffers T_a = Peripheral access time T_c = AHB clock period. This is equal to twice the CPU clock period. Any decimal portion must be rounded up. All values are in nanoseconds.

Table 162: Static Memory Read Delay 0–3 registers

Static Memory Page Mode Read Delay 0–3 registers

Address: A070 0210 / 0230 / 0250 / 0270

The Static Memory Page Mode Read Delay 0-3 registers allow you to program the delay for asynchronous page mode sequential accesses. These registers control the overall period for the read cycle. It is recommended that these registers be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											WTPG				

Register bit assignment

Bits	Access	Mnemonic	Description
D31:05	N/A	Reserved	N/A (do not modify)
D04:00	R/W	WTPG	**Asynchronous page mode read after the first wait state (WAITPAGE)** 00000–11110 (n+1) HCLK cycle for read access time. For asynchronous page mode read for sequential reads, the wait state time for page mode accesses after the first read is (WAITPAGE+1) x t_{HCLK} 11111 32 HCLK cycles read access time (reset value on reset_n) Number of wait states for asynchronous page mode read accesses after the first read.

Table 163: Static Memory Page Mode Read Delay 0–3 registers

Static Memory Write Delay 0–3 registers

Address: A070 0214 / 0234 / 0254 / 0274

The Static Memory Write Delay 0–3 registers allow you to program the delay from the chip select to the write access. These registers control the overall period for the write cycle. It is recommended that these registers be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode.These registers are not used if the extended wait bit is enabled in the related Static Memory Configuration register (see page 230).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											WTWR				

Register bit assignment

Bits	Access	Mnemonic	Description
D31:05	N/A	Reserved	N/A (do not modify)
D04:00	R/W	WTWR	**Write wait states (WAITWR)** 00000–11110 (n+2) HCLK cycle write access time. The wait state time for write accesses after the first read is WAITWR (n+2) x t_{HCLK} 11111 332 HCLK cycle write access time (reset value on reset_n) SRAM wait state time for write accesses after the first read.

Table 164: Static Memory Write Delay 0–3 registers

Static Memory Turn Round Delay 0–3 registers

Address: A070 0218 / 0238 / 0258 / 0278

The Static Memory Turn Round Delay 0–3 registers allow you to program the number of bus turnaround cycles. It is recommended that these registers be modified during system initialization, or when there are no current or outstanding transactions. Wait until the memory controller is idle, then enter low-power or disabled mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												WTTN			

Register bit assignment

Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	WTTN	**Bus turnaround cycles (WAITTURN)** 0000–1110 (n+1) HCLK turnaround cycles, where bus turnaround time is (WAITTURN+1) x t_{HCLK} 1111 16 HCLK turnaround cycles (reset value on reset_n).

Table 165: Static Memory Turn Round Delay 0–3 registers

To prevent bus contention on the external memory databus, the WAITTURN field controls the number of bus turnaround cycles added between static memory read and write accesses.

The WAITTURN field also controls the number of turnaround cycles between static memory and dynamic memory accesses.

System Control Module

C H A P T E R 5

The System Control Module configures and oversees system operations for the NS9750, and defines both the NS9750 AHB arbiter system and system memory address space.

System Control Module features

The System Control Module uses the following to configure and maintain NS9750 system operations:

- AHB arbiter system
- System-level address decoding
- 18 programmable timers
  - Watchdog timer
  - Bus monitor timer for the system bus (a second bus monitor timer, for peripheral devices, is discussed in the BBus Bridge chapter)
  - 16 general purpose timers/counters
- Interrupt controller
- Multiple configuration and status registers
- System Sleep/Wake-up processor

Bus interconnection

The AMBA AHB bus protocol uses a central multiplexor interconnection scheme. All bus masters generate the address and control signals that indicate the transfer that the bus masters want to perform. The arbiter determines which master has its address and control signals routed to all slaves. A central decoder is required to control the read data and response multiplexor, which selects the appropriate signals from the slave that is involved in the transfer.

System bus arbiter

The bus arbitration mechanism ensures that only one bus master has access to the system bus at any time. If you are using a system in which bus bandwidth allocation is critical, you must be sure that your worst-case bus bandwidth allocation goals can be met. See "Arbiter configuration examples" on page 258 for information about configuring the AHB arbiter.

The NS9750 high-speed bus system is split into two subsystems:

- **High-speed peripheral subsystem:** Connects all high-speed peripheral devices to a port on the external memory controller.
- **CPU subsystem:** Connects the CPU directly to a second port on the external memory controller.

Figure 60 shows an overview of the NS9750 high-speed bus architecture.

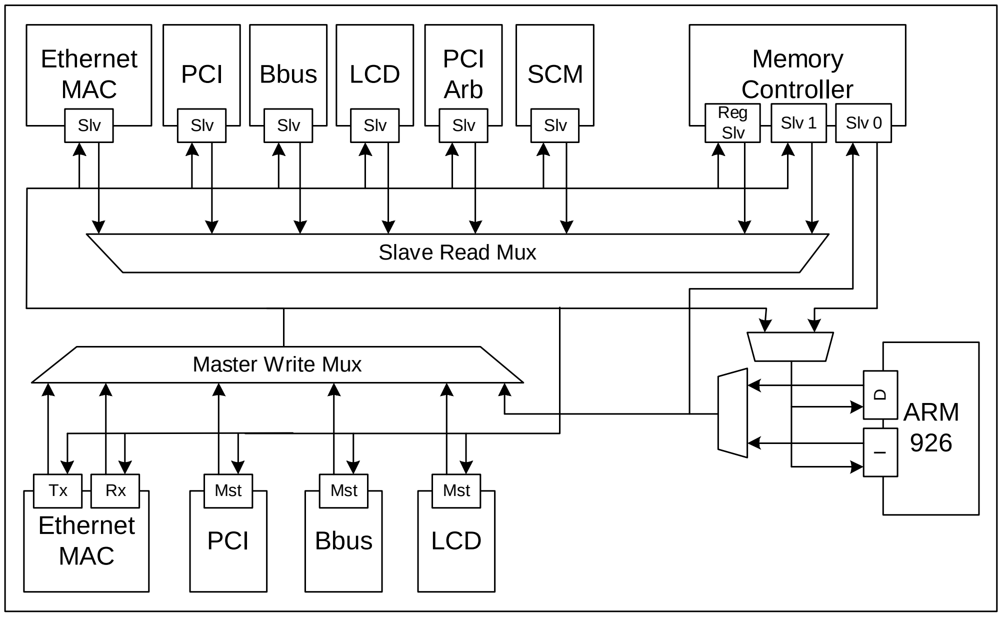


Figure 60: NS9750 bus architecture

The NS9750 high-speed bus contains two arbiters: one for the ARM926 (CPU) and one for the main bus.

- **CPU arbiter.** Splits the bandwidth 50–50 between the data and instruction interfaces. If the CPU access is to external memory, no further arbitration is necessary; the CPU has immediate access to external memory through slave port 0 on the memory controller. If CPU access is to one of the peripherals on the main bus, however, the main arbiter will arbitrate the access.

- **Main arbiter.** Contains a 16-entry Bus Request Configuration (BRC) register. Each BRC entry represents a bus request and grant channel. Each request/grant channel can be assigned to only one bus master at a time. Each bus master can be connected to multiple request/grant channels simultaneously, however, depending on the bus bandwidth requirement of that master.

  Each request/grant channel has a two-bit Bandwidth Reduction Field (BRF) to determine how often each channel can arbitrate for the system bus — 100%, 75%, 50%, or 25%. A BRF value of 25%, for example, causes a channel to be skipped every 3 or 4 cycles. The BRC gates the bus requesting signals going into a 16-entry Bus Request register (BRR). As a default, unassigned channels in the BRC block the corresponding BRR entries from being set by any bus request signals. On powerup, only the CPU is assigned to one of the channels with 100% bandwidth strength as the default setting.

How the bus arbiter works

1. The arbiter evaluates the Bus Request register at every bus clock until one or more bus requests are registered.
2. The arbiter stops evaluating the Bus Request register until a bus grant is issued for the previous evaluation cycle.
3. The arbiter grants the bus to requesting channels, in a round-robin manner, at the rising clock edge of the last address issued for the current transaction (note that each transaction may have multiple transfers), when a SPLIT response is sampled by the arbiter, or when the bus is idling.
4. Each master samples the bus grant signal (hgrant_x) at the end of the current transfer, as indicated by the hready signal. The bus master takes ownership of the bus at this time.
5. The arbiter updates the hmaster [3:0] signals at the same time to indicate the current bus master and to enable the new master's address and control signals to the system bus.

See your AMBA standards documentation for detailed information and illustrations of AMBA AHB transactions.

Ownership

Ownership of the data bus is delayed from ownership of the address/control bus. When hready indicates that a transfer is complete, the master that owns the address/control bus can use the data bus — and continues to own that data bus — until the transaction completes.

Note: If a master is assigned more than one request/grant channel, these channels need to be set and reset simultaneously to guarantee that a non-requesting master will not occupy the system bus.

Locked bus sequence

The arbiter observes the hlock_x signal from each master to allow guaranteed back-to-back cycles, such as read-modified-write cycles. The arbiter ensures that no other bus masters are granted the bus until the locked sequence has completed. To support SPLIT or RETRY transfers in a locked sequence, the arbiter retains the bus master as granted for an additional transfer to ensure that the last transfer in the locked sequence completed successfully.

If the master is performing a locked transfer and the slave issues a split response, the master continues to be granted the bus until the slave finishes the SPLIT response. (This situation degrades AHB performance.)

Relinquishing the bus

When the current bus master relinquishes the bus, ownership is granted to the next requester.

- If there are no new requesters, ownership is granted to a dummy default master. The default master must perform IDLE transfers to keep the arbiter alive.
- *Bus parking* must be maintained if other masters are waiting for SPLIT transfers to complete.
- If the bus is granted to a default master and continues to be in the IDLE state longer than a specified period of time, an AHB bus arbiter timeout is generated (see "Address decoding" on page 261). An AHB bus arbiter timeout can be configured to interrupt the CPU or to reset the chip.

SPLIT transfers

A SPLIT transfer occurs when a slave is not ready to perform the transfer. The slave splits, or masks, its master, taking away the master's bus ownership and allowing other masters to perform transactions until the slave has the appropriate resources to perform its master's transaction.

The bus arbiter supports SPLIT transfers. When a SPLIT response is issued by a slave, the current master is masked for further bus requesting until a corresponding hsplit_x[15:0] signal is issued by the slave indicating that the slave is ready to complete the transfer. The arbiter uses the hsplit_x[15:0] signals to unmask the corresponding master, and treats the master as the highest-priority requester for the immediate next round of arbitration. The master eventually is granted access to the bus to try the transfer again.

Note: The arbiter automatically blocks bus requests with addresses directed at a "SPLITting" slave until that SPLIT transaction is completed.

Arbiter configuration examples

These examples show how to configure the AHB arbiter to guarantee bandwidth to a given master. These are the conditions in this example:

- 5 AHB masters — Ethernet Rx, Ethernet Tx, PCI, BBus, and LCD.
- Memory clock frequency — 100 MHz (this is the AHB clock frequency).
- Average access time per 32-byte memory access — 16 clock cycles.
- The ARM926EJ-S is guaranteed one-half the total memory bandwidth.

In these examples, the bandwidth for each master can be calculated using this formula:

Bandwidth per master:

= [(100MHz/2) / (16 clock cycles per access x 5 masters)] x 32 bytes

= 20 Mbytes/master

The factor 100MHz/2 is given due to the ARM926EJ-S guarantee of one-half the total memory bandwidth. If the ARM926EJ-S consumes less than the guaranteed memory bandwidth, however, the unused bandwidth will be shared by the other masters.

Note: The worst case scenario is that there are 100 Mbytes total to be split by all 5 masters.

Example 1

Since the 20 Mbyte per master guarantee meets the requirements of all masters, the AHB arbiter will be programmed as follows:

BRC0[31:24]	= 8’b1_0_00_0000	channel enabled, 100%, ARM926EJ-S
BRC0[23:16]	= 8’b1_0_00_0001	channel enabled, 100%, Ethernet Rx
BRC0[15:8]	= 8’b1_0_00_0000	channel enabled, 100%, ARM926EJ-S
BRC0[7:0]	= 8’b1_0_00_0010	channel enabled, 100% Ethernet Tx
BRC1[31:24]	= 8’b1_0_00_0000	channel enabled, 100%, ARM926EJ-S
BRC1[23:16]	= 8’b1_0_00_0100	channel enabled, 100%, PCI
BRC1[15:8]	= 8’b1_0_00_0000	channel enabled, 100%, ARM926EJ-S
BRC1[7:0]	= 8’b1_0_00_0101	channel enabled, 100%, BBus
BRC2[31:24]	= 8’b1_0_00_0000	channel enabled, 100%, ARM926EJ-S
BRC2[23:16]	= 8’b1_0_00_0100	channel enabled, 100%, LCD
BRC2[15:8]	= 8’b0_0_00_0000	channel disabled
BRC2[7:0]	= 8’b0_0_00_0000	channel disabled
BRC3[31:24]	= 8’b0_0_00_0000	channel disabled
BRC3[23:16]	= 8’b0_0_00_0000	channel disabled
BRC3[15:8]	= 8’b0_0_00_0000	channel disabled
BRC[7:0]	= 8’b0_0_00_0000	channel disabled

Example 2

In this example, the LCD master needs more than 20 Mbytes and the other masters need less than 20 Mbytes. These are the new requirements:

- Ethernet Rx — 12.5 Mbytes
- Ethernet Tx — 12.5 Mbytes
- PCI — 16 Mbytes
- BBus — 4 Mbytes
- LCD — 25 Mbytes
- Total — 70 Mbytes

This configuration is possible because the total bandwidth is less than the 100 Mbytes available. The LCD master will be configured to have two arbiter slots, resulting in a total of 6 masters.

The available bandwidth per master is calculated using this formula:

Bandwidth per master:

= [(100MHz/2) / (16 clock cycles per access x 6 masters)] x 32 bytes

= 16.667 Mbytes/master

If the LCD is configured for two arbiter channel slots, then, there are 33.334 Mbytes available, which is greater than the 25 Mbytes required. Each of the other masters have 16.667 Mbytes available, which is more than enough to meet their requirements.

Note: When assigning two arbiter channel slots to a master, the slot assignments should be spaced equally.

The AHB arbiter will be programmed as follows:

BRC0[31:24]	= 8'b1_0_00_0000	channel enabled, 100%, ARM926EJ-S
BRC0[23:16]	= 8'b1_0_00_0001	channel enabled, 100%, Ethernet Rx
BRC0[15:8]	= 8'b1_0_00_0000	channel enabled, 100%, ARM926EJ-S
BRC0[7:0]	= 8'b1_0_00_0010	channel enabled, 100%, Ethernet Tx
BRC1[31:24]	= 8'b1_0_00_0000	channel enabled, 100%, ARM926EJ-S
BRC1[23:16]	= 8'b1_0_00_0110	channel enabled, 100%, LCD *first slot*
BRC1[15:8]	= 8'b1_0_00_0000	channel enabled, 100%, ARM926EJ-S
BRC1[7:0]	= 8'b1_0_00_0100	channel enabled, 100%, PCI
BRC2[31:24]	= 8'b1_0_00_0000	channel enabled, 100%, ARM926EJ-S
BRC2[23:16]	= 8'b1_0_00_0101	channel enabled, 100%, BBus
BRC2[15:8]	= 8'b1_0_00_0000	channel enabled, 100%, ARM926EJ-S
BRC2[7:0]	= 8'b1_0_00_0110	channel enabled, 100%, LCD *second slot*
BRC3[31:24]	= 8'b0_0_00_0000	channel disabled
BRC3[23:16]	= 8'b0_0_00_0000	channel disabled
BRC3[15:8]	= 8'b0_0_00_0000	channel disabled
BRC[7:0]	= 8'b0_0_00_0000	channel disabled

Note that the BBus requires 4 Mbytes but has been allocated 16.667 Mbytes. The BBus bandwidth can be reduced using the bandwidth reduction field (in the BRC registers). To reduce the available bandwidth by 25% to 4.167 Mbytes, for example, the 2-bit field can be set to 2'b11. This restricts the BBus master when the system is fully loaded. The new configuration for BBus is:

BRC2[23:16]	= 8'b1_0_11_0101	channel enabled, 25%, BBus

Address decoding

A central address decoder provides a select signal — hsel_x — for each slave on the bus.

Table 166 shows how the system memory address is set up to allow access to the internal and external resources on the system bus. Note that the external memory chip select ranges can be reset after powerup. The table shows the default powerup values; you can change the ranges by writing to the BASE and MASK registers (see "System Memory Chip Select 0 Dynamic Memory Base and Mask registers" on page 303 through "System Memory Chip Select 3 Dynamic Memory Base and Mask registers" on page 306 for more information).

See the BBus bridge chapter for information about BBus peripheral address decoding.

Address range	Size	System functions
0x0000 0000 – 0x0FFF FFFF	256 MB	System memory chip select 4 Dynamic memory (default)
0x1000 0000 – 0x1FFF FFFF	256 MB	System memory chip select 5 Dynamic memory (default)
0x2000 0000 – 0x2FFF FFFF	256 MB	System memory chip select 6 Dynamic memory (default)
0x3000 0000 – 0x3FFF FFFF	256 MB	System memory chip select 7 Dynamic memory (default)
0x4000 0000 – 0x4FFF FFFF	256 MB	System memory chip select 0 Static memory (default)
0x5000 0000 – 0x5FFF FFFF	256 MB	System memory chip select 1 Static memory (default)
0x6000 0000 – 0x6FFF FFFF	256 MB	System memory chip select 2 Static memory (default)
0x7000 0000 – 0x7FFF FFFF	256 MB	System memory chip select 3 Static memory (default)
0x8000 0000 – 0x8FFF FFFF	256 MB	PCI memory
0x9000 0000 – 0x9FFF FFFF	256 MB	BBus memory
0xA000 0000 – 0xA00F FFFF	1 MB	PCI IO

Table 166: System address map

Address range	Size	System functions
0xA010 0000 – 0xA01F FFFF	1 MB	PCI CONFIG_ADDR
0xA020 0000 – 0xA02F FFFF	1 MB	PCI CONFIG_DATA
0xA030 0000 – 0xA03F FFFF	1 MB	PCI arbiter
0xA040 0000 – 0xA04F FFFF	1 MB	BBUS-to-AHB bridge
0xA050 0000 – 0xA05F FFFF	1 MB	Reserved
0xA060 0000 – 0xA06F FFFF	1 MB	Ethernet Communication Module
0xA070 0000 – 0xA07F FFFF	1 MB	Memory controller
0xA080 0000 – 0xA08F FFFF	1 MB	LCD controller
0xA090 0000 – 0xA09F FFFF	1 MB	System Control Module
0xA0A0 0000 – 0xFFFF FFFF	1526	Reserved

Table 166: System address map

The internal registers, unlike system memory, can be accessed only in *privileged access mode*. Privileged access mode is indicated when HPROT[1] is active high.

Table 167 shows the hmaster[3:0] assignments for NS9750.

Master Name	hmaster[3:0] assignment
ARM926 I/D	0000
Ethernet Rx	0001
Ethernet Tx	0010
Reserved	0011
PCI	0100
BBus	0101
LCD	0110

Table 167: Hmaster encoding

Programmable timers

NS9750 provides 18 programmable timers:

- Software watchdog timer
- Bus monitor timer
- 16 general purpose timers

Software watchdog timer

The software watchdog timer, set to specific time intervals, handles gross system misbehaviors. The watchdog timer can be set to timeout in longer ranges of time intervals, typically in seconds.

The software watchdog timer can be enabled or disabled, depending on the operating condition. When enabled, system software must write to the Software Watchdog Timer register before it expires. When the timer does timeout, the system is preconfigured to generate an IRQ, an FIQ, or a RESET to restart the entire system.

General purpose timers/counters

Sixteen general purpose timers/counters (GPTCs), which can be concatenated, provide programmable time intervals to the CPU when used as one or multiple timers. There is one I/O pin associated with each timer.

- When used as a **gated** timer, the GPTC I/O pin is an input qualifier (high/low programmable).
- When used as a **regular** timer (enabled by software) the GPTC I/O pin serves as a terminal count indicator output.

The timers also can be used independently, as up/down counters that monitor the frequency of certain events (events capturing). In these situations, the GPTC I/O pin becomes the clock source of the counter. See "GPIO MUX" on page 34 for information about GPIO pin-to-timer assignments.

Depending on the application, the source clock frequency of the timers/counters can be selected as the CPU clock, the CPU clock with multiple divisor options, or an external pulse event. The source frequency is indicated in the timer clock select field in the appropriate Timer Control register (see "Timer 0–15 Control registers" on page 301).

With a 16-bit counter and a 16-bit prescaler, each GPTC can measure external event length up to minutes in range, and can be individually enabled or disabled. GPTCs can be configured to reload, with the value defined in the appropriate Timer Reload Count register (see page 284), and generates an interrupt upon terminal count. Each GPTC has an interrupt request connected to the IRQ vector interrupt controller (VIC). The priority level and enable/disable of each interrupt can be programmed in the VIC, and the contents of the timer/counter can be read by the CPU.

The GPTCs can be concatenated to form counters for longer time scales.

These control fields should be in the control register of each GPTC:

- Clock frequency selection
- Mode of operation:
  - Internal timer, with or without external terminal count indicator
  - External gated timer with gate active low
  - External gated timer with gate active high
  - External event counter; frequency must be less than one half the CPU clock frequency
- Timer/counter enable
- Count up or down
- Interrupt enable
- Concatenate to upstream timer/counter. That is, use upstream timer/counter’s overflow/underflow output as clock input (16- or 32-bit timer/counter).
- Reload enable

Debug mode

The 16 timers/counters continue to run when the debugger halts the CPU in debug mode. This is not a problem in normal operation.

There is a script available that causes the debugger to continually reset one or more timers while the CPU is halted. Use this debugger script to work around this issue.

```
//    This command file initializes the debugger local variables that are
//    used by the user defined On-Stop and Idle-Mode command descriptors.
//
//    NOTE:      DO NOT CHANGE THIS FILE. This file configures the resources
//               needed to use this feature. To specify an On-Stop or Idle-Mode
//               command for your target, add an EW command to your board init
//               file using the syntax shown below to define the sequence of
//               operations, enable or disable them, and set the Idle-Mode timer
//               interval.
//
//    ..........................................................................
//
//    The On-Stop command descriptor performs a user defined sequence of
//    memory operations whenever program execution stops. This may be
//    used, for example, to disable a watchdog timer or other peripheral.
//    The On-Stop command descriptor is a 16 word (max) buffer in the
//    following format:
//
//        ew MAJIC_ON_STOP_CMD = en, { @op, addr [ , data [ , mask ] ] }...
//
//    The Idle-Mode command descriptor performs a user defined sequence of
//    memory operations periodically while the program is stopped. This
//    may be used, for example, to periodically access a watchdog timer
//    or another peripheral. The Idle-Mode command descriptor is a 16 word
//    (max) buffer in the following format:
//
//        ew MAJIC_IDLE_MODE_CMD = int, { @op, addr [ , data [ , mask ] ] }...
//
//    Parameters:
//
//        en       enable:      0 to disable, 1 to enable
//        int      interval:    0 to disable, non-0 for interval in milliseconds
//        op       opcode:      Any of the $ucd_xxxx aliases below
//        addr     Address of the access (must be properly aligned for op size)
//        data     Data value   for $ucd_wr and $ucd_rmw (omit for $ucd_rd)
//        mask     Data mask    for $ucd_rmw (1's are bits to replace with data)
//
//    ............................................................................
//
```

```
//     Examples:
//
//            ew MAJIC_ON_STOP_CMD = 1, @$ucd_rd8, FFF00003
//
//     Defines an On-Stop command that reads the byte at 0xFFF00003 upon
//     stopping.
//
//            ew MAJIC_ON_STOP_CMD = 1, @$ucd_rmw16, 80000000, C00, F00
//
//     Defines an On-Stop command that reads a 16-bit value from 80000000,
//     masks off bits 11..8, sets those bits to 1100, and writes the result
//     back to 80000000.
//
//            ew MAJIC_IDLE_MODE_CMD = 0n250, @$ucd_wr32, 0x40000000, 0n5000
//
//     Defines the Idle-Mode command that writes the 32-bit value 5000 (decimal)
//     to the register at 0x40000000 every 250 milliseconds.
//
//            ew MAJIC_ON_STOP_CMD= 0
//            ew MAJIC_IDLE_MODE_CMD= 0
//
//     Disables the On-Stop and Idle-Mode command descriptors.
//
//     ..........................................................................
//
ew $ucd_rd8        = 101        //     8-bit read operation
ew $ucd_rd16       = 102        //    16-bit read operation
ew $ucd_rd32       = 104        //    32-bit read operation
ew $ucd_wr8        = 201        //     8-bit write operation
ew $ucd_wr16       = 202        //    16-bit write operation
ew $ucd_wr32       = 204        //    32-bit write operation
ew $ucd_rmw8       = 301        //     8-bit RdModWr operation
ew $ucd_rmw16      = 302        //    16-bit RdModWr operation
ew $ucd_rmw32      = 304        //    32-bit RdModWr operation
//
//     <eof>
```

Interrupt controller

The interrupt system is a simple two-tier priority scheme. Two lines access the CPU core and can interrupt the processor: IRQ (normal interrupt) and FIQ (fast interrupt). FIQ has a higher priority than IRQ.

FIQ interrupts

Most sources of interrupts on NS9750 are from the IRQ line. There is only one FIQ source for timing-critical applications. The FIQ interrupt generally is reserved for timing-critical applications for these reasons:

- The interrupt service routine is executed directly without determining the source of the interrupt.
- Interrupt latency is reduced. The banked registers available for FIQ interrupts are more efficient because a context save is not required.

Note: The interrupt source assigned to the FIQ must be assigned to the highest priority, which is 0.

IRQ interrupts

IRQ interrupts come from several different sources in NS9750, and are managed using the Interrupt Config registers (see "Int (Interrupt) Config (Configuration) registers (0–31)" on page 286). IRQ interrupts can be enabled or disabled on a per-level basis using the Interrupt Enable registers. These registers serve as masks for the different interrupt levels. Each interrupt level has two registers:

- **Interrupt Configuration register.** Use this register to assign the source for each interrupt level, invert the source polarity, select IRQ or FIQ, and enable the level.
- **Interrupt Vector Address register.** Contains the address of the interrupt service routine.

Figure 61 illustrates a 32-vector interrupt controller.

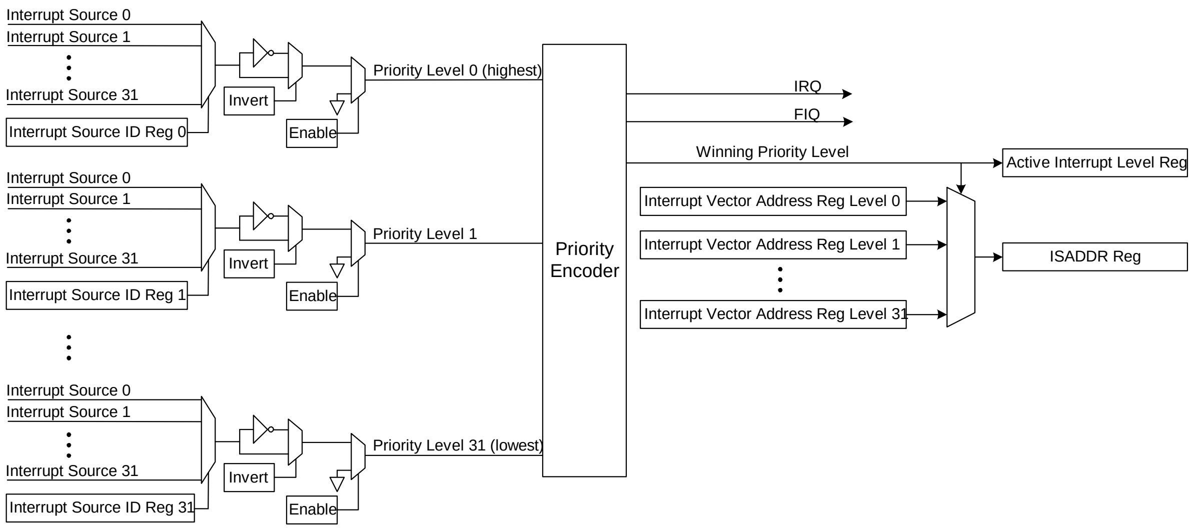

Interrupt Source 0
Interrupt Source 1
Interrupt Source 31
Interrupt Source ID Reg 0
Invert
Enable
Priority Level 0 (highest)
Interrupt Source ID Reg 1
Priority Level 1
Interrupt Source ID Reg 31
Priority Level 31 (lowest)
Priority Encoder
IRQ
FIQ
Winning Priority Level
Active Interrupt Level Reg
Interrupt Vector Address Reg Level 0
Interrupt Vector Address Reg Level 1
Interrupt Vector Address Reg Level 31
ISADDR Reg

Figure 61: Interrupt controller block diagram

The IRQ interrupts are enabled by the respective enabling bits. Once enabled, the interrupt source programmed in the Interrupt Configuration register for each priority level connects the interrupt to one of 32 priority lines going into the *priority encoder block*. The priority encoder block has a fixed order, with line 0 as the highest priority. The interrupt with the highest priority level has its encoded priority level displayed, to select the appropriate vector for the ISRADDR register (see "ISRADDR register" on page 288). The CPU, once interrupted, can read the ISRADDR register to get the address of the Interrupt Service Routine. A read to the ISRADDR register updates the priority encoder block, which masks the current and any lower priority interrupt requests. Writing to this address indicates to the priority hardware that the current interrupt is serviced, allowing lower priority interrupts to become active.

The priority encoder block enables 32 prioritized interrupts to be serviced in nested fashion. A software interrupt can be implemented by writing to a software interrupt register. The software interrupt typically is assigned level 1 or level 2 priority.

Interrupt sources

An Interrupt Status register shows the current active interrupt requests. The Raw Interrupts register shows the status of the unmasked interrupt requests.

The NS9750 interrupt sources are assigned as shown:

Interrupt ID	Interrupt source
0	Watchdog Timer
1	AHB Bus Error
2	BBus Aggregate Interrupt
3	Reserved
4	Ethernet Module Receive Interrupt
5	Ethernet Module Transmit Interrupt
6	Ethernet Phy Interrupt
7	LCD Module interrupt
8	PCI Bridge Module Interrupt
9	PCI Arbiter Module Interrupt
10	PCI External Interrupt 0
11	PCI External Interrupt 1
12	PCI External Interrupt 2
13	PCI External Interrupt 3
14	I2C Interrupt
15	BBus DMA Interrupt
16	Timer Interrupt 0
17	Timer Interrupt 1
18	Timer Interrupt 2
19	Timer Interrupt 3
20	Timer Interrupt 4
21	Timer Interrupt 5
22	Timer Interrupt 6
23	Timer Interrupt 7
24	Timer Interrupt 8 and 9
25	Timer Interrupt 10 and 11

Interrupt ID	Interrupt source
26	Timer Interrupt 12 and 13
27	Timer Interrupt 14 and 15
28	External Interrupt 0
29	External Interrupt 1
30	External Interrupt 2
31	External Interrupt 3

Vectored interrupt controller (VIC) flow

A vectored interrupt controller allows a reasonable interrupt latency for IRQ-line interrupts. When an interrupt occurs, the CPU processor determines whether the interrupt is from a FIQ or IRQ line. If the interrupt comes from the FIQ vector, the interrupt service routine can be executed without knowing the interrupt source.

If the interrupt comes from the IRQ vector, the CPU performs these steps:

1. Reads the service routine address from the VIC's ISRADDR register. The read updates the VIC's priority hardware to prevent current or any lower priority interrupts from interrupting until the higher priority interrupt has occurred.
2. Branches to the interrupt service routine and stacks the workspace so the IRQ can be enabled.
3. Executes the interrupt service routine.
4. Clears the current interrupt from the source.
5. Disables the IRQ and restores the workplace.
6. Writes to the ISRADDR register to clear the current interrupt path in the VIC's priority hardware. Any value can be written.
7. Returns from the interrupt service routine.

System attributes

System software can configure these NS9750 system attributes:

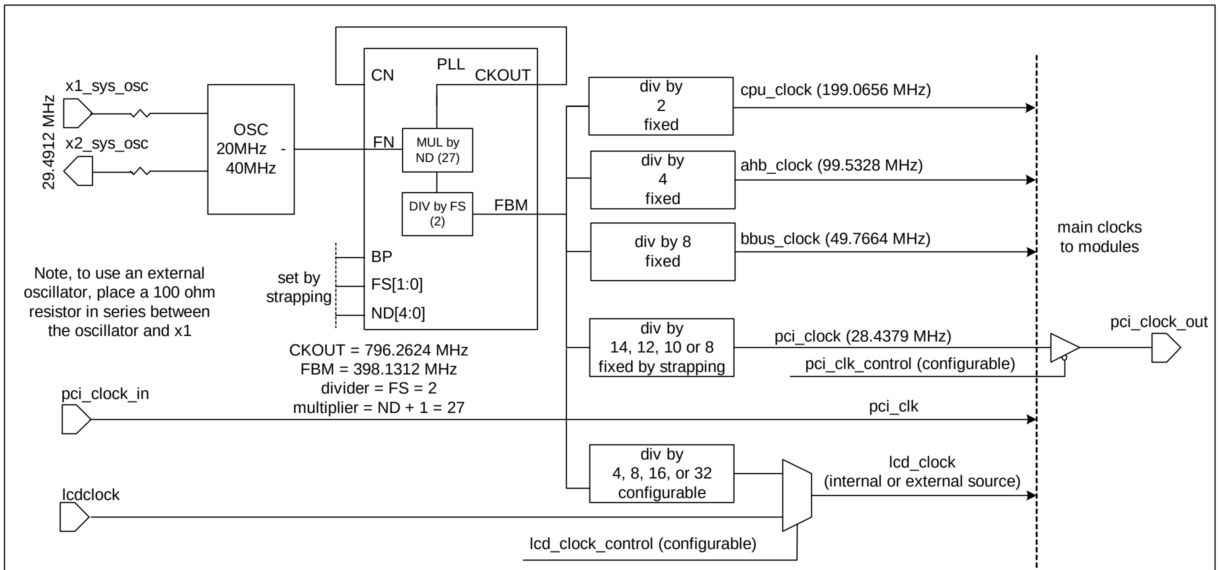
- Little endian/big endian mode
- Watchdog timer enable
- Watchdog timeout generates IRQ/FIQ/RESET
- Watchdog timeout interval
- Enable/disable ERROR response for misaligned data access
- System module clock enables
- Enable access to internal registers in USER mode
- Bus monitor enable
- Bus monitor timeout interval
- Bus arbiter timer enable
- Bus arbiter timeout period
- Bus arbiter timeout response (IRQ/FIQ/RESET)
- Bus bandwidth configuration
- Wake-up processor enable

PLL configuration

PLL operating parameters are initialized on a powerup hardware reset. Software reads the powerup hardware settings by reading the status fields in the PLL Configuration register (see "PLL Configuration register" on page 299). Software can change the PLL configuration after a powerup reset by writing to the appropriate SW field in the PLL Configuration register (see "PLL Configuration register," beginning on page 299). Once the new settings have been written, the PLL SW change bit must be set (see page 300). The PLL settings then are written to the PLL, and the system is reset.

The PLL can be configured at powerup by placing pulldowns on the external memory address pins. NS9750 provides internal pullups to produce a default configuration; see "Bootstrap initialization" on page 272 for information about the powerup configuration.

Figure 62 shows how the PLL clock is used to provide the NS9750 system clocks.



Sample Clock Frequency Settings With 29.4912MHz Crystal (FS= 01, div by 2)

ND+1	f_{VCO}	cpu_clk	hclk	bbus_clk	pci_clk	lcd_clk
27	398.1312	199.0656	99.5328	49.7664	28.4379(14)	99.5328 - 12.4416
26	383.3856	191.6928	95.8464	47.9232	31.9488(12)	95.8464 - 11.9808
25	368.6400	184.3200	92.1600	46.0800	30.7200(12)	92.1600 - 11.5200
24	353.8944	176.9472	88.4736	44.2368	29.4912(12)	88.7872 - 11.0592
23	339.1488	169.5744	84.7872	42.3936	28.2624(12)	84.7872 - 10.5984
22	324.4032	162.2016	81.1008	40.5504	32.4403(10)	81.1008 - 10.1376
21	309.6576	154.8288	77.4144	38.7072	30.9657(10)	77.4144 - 9.6768
20	294.9120	147.4560	73.7280	36.8640	29.4912(10)	73.7280 - 9.2160
19	280.1644	140.0832	70.0416	35.0208	28.0164(10)	70.0416 - 8.7552
18	265.4208	132.7104	66.3552	33.1776	26.5420(10)	66.3552 - 8.2944
17	250.6752	125.3376	62.6688	31.3344	31.3344(8)	62.6688 - 7.8336
16	235.9296	117.9648	58.9824	29.4912	29.4912(8)	58.9824 - 7.3728
15	221.1840	110.5920	55.2960	27.6480	27.6480(8)	55.2960 - 6.9120
14	206.4384	103.2192	51.6096	24.8048	25.8048(8)	51.6096 - 6.4512

Figure 62: NS9750 system clock generation (PLL)

Bootstrap initialization

The PLL and other system configuration settings can be configured at powerup before the CPU boots. External pins are used to configure the necessary control register bits at powerup. External pulldown resistors can be used to configure the PLL and system configuration registers depending on the application. The recommended value is 2.2k ohm to 2.4k ohm.

Table 168 indicates how each bit is used to configure the powerup settings, where 1 indicates the internal pullup resistor and 0 indicates an external pulldown resistor. Table 169 shows PLL ND[4:0] multiplier values.

Pin name	Configuration bits
rtck	**PCI arbiter configuration** 0 External PCI arbiter 1 Internal PCI arbiter
boot_strap[0]	**Chip select 1 byte_lane_enable_n/write_enable_n configuration bootstrap select** 0 write_enable_n for byte-wide devices (default) 1 byte_lane_enable_n (2.4K pulldown added)
boot_strap[4:3]	**Chip select 1 data width bootstrap select** 00 16 bits 01 8 bits 11 32 bits
boot_strap[2]	**Memory interface read mode bootstrap select** **Note:** An external pulldown resistor must be used to select command delayed mode. Clock delayed mode is reserved for future use. 0 Command delayed mode Commands are launched on a 90-degree phase-shifted AHB clock, and the AHB clock is routed to the external dynamic memory. 1 Clock delayed mode Reserved for future use.
boot_strap[1]	**CardBus mode bootstrap select** 0 CardBus mode 1 PCI mode
gpio[49]	**Chip select polarity** 0 Active high 1 Active low
gpio[44]	**Endian mode** 0 Big endian 1 Little endian

Table 168: Configuration pins — Bootstrap initialization

Pin name	Configuration bits
reset_done	**Bootup mode** 0 Boot from SDRAM using serial SPI EEPROM 1 Boot from flash/ROM
gpio[19]	**PLL BP (PLL bypass)** 0 PLL bypassed 1 PLL not bypassed
gpio[17], gpio[12], gpio[10], gpio[8], gpio[4]	**PLL ND[4:0] (PLL multiplier, ND+1)** See Table 169: "PLL ND[4:0] multiplier values."
gpio[2], gpio[0]	**PLL FS[1:0] (PLL frequency select)** **GPIO** **FS** **Divide by** 10 00 1 11 01 2 00 10 4 01 11 8

Table 168: Configuration pins — Bootstrap initialization

Register configuration: gpio 17, 12, 10, 8, 4	Multiplier
11010	**32**
00100	**31**
11000	**30**
11001	**29**
11110	**28**
11111	**27**
11100	**26**
11101	**25**
10010	**24**
10011	**23**
10000	**22**

Table 169: PLL ND[4:0] multiplier values

Register configuration: gpio 17, 12, 10, 8, 4	Multiplier
10001	**21**
10110	**20**
10111	**19**
10100	**18**
10101	**17**
01010	**16**
01011	**15**
01000	**14**
01001	**13**
01110	**12**
01111	**11**
01100	**10**
01101	**9**
00010	**8**
00011	**7**
00000	**6**
00001	**5**
00110	**4**
00111	**3**
00100	**2**
00101	**1**

Table 169: PLL ND[4:0] multiplier values

There are 32 additional GPIO pins that are used to create a general purpose, user-defined ID register (see "Gen ID register" on page 311). These external signals are registered at powerup.

gpio[41]	gpio[40]	gpio[39]	gpio[38]
gpio[37]	gpio[36]	gpio[35]	gpio[34]
gpio[33]	gpio[32]	gpio[31]	gpio[30]
gpio[29]	gpio[28]	gpio[27]	gpio[26]
gpio[25]	gpio[23]	gpio[22]	gpio[21]
gpio[18]	gpio[16]	gpio[15]	gpio[14]
gpio[13]	gpio[11]	gpio[9]	gpio[7]
gpio[6]	gpio[5]	gpio[3]	gpio[1]

Read these signals for general purpose status information.

System configuration registers

Table 170 lists the configuration and status registers for the high-speed AHB bus system. All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Offset	[31:24]	[23:16]	[15:8]	[7:0]
A090 0000	AHB Arbiter Gen Configuration			
A090 0004	BRC0			
A090 0008	BRC1			
A090 000C	BRC2			
A090 0010	BRC3			
A090 0014–A090 0040		Reserved		
A090 0044	Timer 0 Reload Count register			

Table 170: System Control module registers

Offset	[31:24]	[23:16]	[15:8]	[7:0]
A090 0048	Timer 1 Reload Count register			
A090 004C	Timer 2 Reload Count register			
A090 0050	Timer 3 Reload Count register			
A090 0054	Timer 4 Reload Count register			
A090 0058	Timer 5 Reload Count register			
A090 005C	Timer 6 Reload Count register			
A090 0060	Timer 7 Reload Count register			
A090 0064	Timer 8 Reload Count register			
A090 0068	Timer 9 Reload Count register			
A090 006C	Timer 10 Reload Count register			
A090 0070	Timer 11 Reload Count register			
A090 0074	Timer 12 Reload Count register			
A090 0x078	Timer 13 Reload Count register			
A090 007C	Timer 14 Reload Count register			
A090 0080	Timer 15 Reload Count register			
A090 0084	Timer 0 Read register			
A090 0088	Timer 1 Read register			
A090 008C	Timer 2 Read register			
A090 0090	Timer 3 Read register			
A090 0094	Timer 4 Read register			
A090 0098	Timer 5 Read register			
A090 009C	Timer 6 Read register			
A090 00A0	Timer 7 Read register			
A090 00A4	Timer 8 Read register			
A090 00A8	Timer 9 Read register			
A090 00AC	Timer 10 Read register			
A090 00B0	Timer 11 Read register			

Table 170: System Control module registers

Offset	[31:24]	[23:16]	[15:8]	[7:0]
A090 00B4	Timer 12 Read register			
A090 00B8	Timer 13 Read register			
A090 00BC	Timer 14 Read register			
A090 00C0	Timer 15 Read register			
A090 00C4	Interrupt Vector Address Register Level 0			
A090 00C8	Interrupt Vector Address Register Level 1			
A090 00CC	Interrupt Vector Address Register Level 2			
A090 00D0	Interrupt Vector Address Register Level 3			
A090 00D4	Interrupt Vector Address Register Level 4			
A090 00D8	Interrupt Vector Address Register Level 5			
A090 00DC	Interrupt Vector Address Register Level 6			
A090 00E0	Interrupt Vector Address Register Level 7			
A090 00E4	Interrupt Vector Address Register Level 8			
A090 00E8	Interrupt Vector Address Register Level 9			
A090 00EC	Interrupt Vector Address Register Level 10			
A090 00F0	Interrupt Vector Address Register Level 11			
A090 00F4	Interrupt Vector Address Register Level 12			
A090 00F8	Interrupt Vector Address Register Level 13			
A090 00FC	Interrupt Vector Address Register Level 14			
A090 0100	Interrupt Vector Address Register Level 15			
A090 0104	Interrupt Vector Address Register Level 16			
A090 0108	Interrupt Vector Address Register Level 17			
A090 010C	Interrupt Vector Address Register Level 18			
A090 0110	Interrupt Vector Address Register Level 19			
A090 0114	Interrupt Vector Address Register Level 20			
A090 0118	Interrupt Vector Address Register Level 21			
A090 011C	Interrupt Vector Address Register Level 22			

Table 170: System Control module registers

Offset	[31:24]	[23:16]	[15:8]	[7:0]
A090 0120	Interrupt Vector Address Register Level 23			
A090 0124	Interrupt Vector Address Register Level 24			
A090 0128	Interrupt Vector Address Register Level 25			
A090 012C	Interrupt Vector Address Register Level 26			
A090 0130	Interrupt Vector Address Register Level 27			
A090 0134	Interrupt Vector Address Register Level 28			
A090 0138	Interrupt Vector Address Register Level 29			
A090 013C	Interrupt Vector Address Register Level 30			
A090 0140	Interrupt Vector Address Register Level 31			
A090 0144	Int Config 0	Int Config 1	Int Config 2	Int Config 3
A090 0148	Int Config 4	Int Config 5	Int Config 6	Int Config 7
A090 014C	Int Config 8	Int Config 9	Int Config 10	Int Config 11
A090 0150	Int Config 12	Int Config 13	Int Config 14	Int Config 15
A090 0154	Int Config 16	Int Config 17	Int Config 18	Int Config 19
A090 0158	Int Config 20	Int Config 21	Int Config 22	Int Config 23
A090 015C	Int Config 24	Int Config 25	Int Config 26	Int Config 27
A090 0160	Int Config 28	Int Config 29	Int Config 30	Int Config 31
A090 0164	ISRADDR			
A090 0168	Interrupt Status Active			
A090 016C	Interrupt Status Raw			
A090 0170	Timer Interrupt Status register			
A090 0174	Software Watchdog Configuration			
A090 0178	Software Watchdog Timer			
A090 017C	Clock Configuration register			
A090 0180	Reset and Sleep Control register			
A090 0184	Miscellaneous System Configuration register			
A090 0188	PLL Configuration register			

Table 170: System Control module registers

Offset	[31:24]	[23:16]	[15:8]	[7:0]
A090 018C	Active Interrupt Level register			
A090 0190	Timer 0 Control register			
A090 0194	Timer 1 Control register			
A090 0198	Timer 2 Control register			
A090 019C	Timer 3 Control register			
A090 01A0	Timer 4 Control register			
A090 01A4	Timer 5 Control register			
A090 01A8	Timer 6 Control register			
A090 01AC	Timer 7 Control register			
A090 01B0	Timer 8 Control register			
A090 01B4	Timer 9 Control register			
A090 01B8	Timer 10 Control register			
A090 01BC	Timer 11 Control register			
A090 01C0	Timer 12 Control register			
A090 01C4	Timer 13 Control register			
A090 01C8	Timer 14 Control register			
A090 01CC	Timer 15 Control register			
A090 01D0	System Memory Chip Select 0 Dynamic Memory Base			
A090 01D4	System Memory Chip Select 0 Dynamic Memory Mask			
A090 01D8	System Memory Chip Select 1 Dynamic Memory Base			
A090 01DC	System Memory Chip Select 1 Dynamic Memory Mask			
A090 01E0	System Memory Chip Select 2 Dynamic Memory Base			
A090 01E4	System Memory Chip Select 2 Dynamic Memory Mask			
A090 01E8	System Memory Chip Select 3 Dynamic Memory Base			
A090 01EC	System Memory Chip Select 3 Dynamic Memory Mask			
A090 01F0	System Memory Chip Select 0 Static Memory Base			
A090 01F4	System Memory Chip Select 0 Static Memory Mask			

Table 170: System Control module registers

Offset	[31:24]	[23:16]	[15:8]	[7:0]
A090 01F8	System Memory Chip Select 1 Static Memory Base			
A090 01FC	System Memory Chip Select 1 Static Memory Mask			
A090 0200	System Memory Chip Select 2 Static Memory Base			
A090 0204	System Memory Chip Select 2 Static Memory Mask			
A090 0208	System Memory Chip Select 3 Static Memory Base			
A090 020C	System Memory Chip Select 3 Static Memory Mask			
A090 0210	GenID— General purpose, user-defined ID register			
A090 0214	External Interrupt 0 Control register			
A090 0218	External Interrupt 1 Control register			
A090 021C	External Interrupt 2 Control register			
A090 0220	External Interrupt 3 Control register			

Table 170: System Control module registers

AHB Arbiter Gen Configuration register

Address: A090 0000

The AHB Arbiter Gen Configuration register contains miscellaneous control settings for the AHB bus arbiter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EXMA

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:01	N/A	Reserved	N/A	N/A
D00	R/W	EXMA	0x0	**CPU external memory access mode** 0 Enable direct access to external memory through Slv1 1 Disable direct access to external memory, arbitrate with other masters through Slv0

Table 171: AHB Arbiter Gen Configuration register

BRC0, BRC1, BRC2, and BRC3 registers

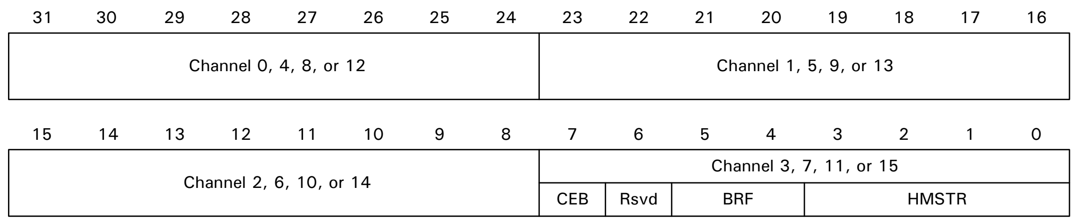
Address: A090 0004 / 0008 / 000C / 0010

The BRC[0:3] registers control the AHB arbiter bandwidth allocation scheme.

Table 172 shows how the channels are assigned in the four registers. Table 173 shows the bit definition, or format, for each channel, using data bits [07:00] as the example.

Register name	[31:24]	[23:16]	[15:08]	[07:00]
BRC0	Channel 0	Channel 1	Channel 2	Channel 3
BRC1	Channel 4	Channel 5	Channel 6	Channel 7
BRC2	Channel 8	Channel 9	Channel 10	Channel 11
BRC3	Channel 12	Channel 13	Channel 14	Channel 15

Table 172: BRC channel assignment



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D07	R/W	CEB	0x0	**Channel enable bit** 0 Disable 1 Enable
D06	N/A	Reserved	N/A	N/A

Table 173: BRC0, BRC1, BRC2, BRC3 register

Bits	Access	Mnemonic	Reset	Description
D05:04	R/W	BRF	0x0	**Bandwidth reduction field** 00 100% 01 75% 10 50% 11 25% Programs the weight for each AHB bus master. Used to limit the round robin scheduler.
D03:00	R/W	HMSTR	0x0	**hmaster** Program a particular AHB bus master number here. Note that a particular master can be programmed to more than one channel.

Table 173: BRC0, BRC1, BRC2, BRC3 register

Timer 0–15 Reload Count registers

Address: A090 0044 (Timer 0) / 0048 / 004C / 0050 / 0054 / 0058 / 005C / 0060 / 0064 / 0068 / 006C / 0070 / 0074 / 0078 / 007C / 0080 (Timer 15)

The Timer Reload registers hold the up/down reload value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Timer reload count (TRCV)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Timer reload count (TRCV)															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	TRCV	0x0	**Timer Reload Count register value** Value loaded into the Timer register after the timer is enabled and after the terminal count has been reached, if the reload enable bit in the corresponding Timer Control register is set.

Table 174: Timer Reload Count register

Timer 0–15 Read register

Address: A090 0084 / 0088 / 008C / 0090 / 0094 / 0098 / 009C / 00A0 / 00A4 / 00A8 / 00AC / 00B0 / 00B4 / 00B8 / 00BC / 00C0

The Timer Read registers read the current state of each Timer register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Timer read (TRR)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Timer read (TRR)															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R	TRR	0x0	**Timer Read register** Reads the current state of each counter in a register.

Table 175: Timer Read register

Interrupt Vector Address Register Level 0–31

Address: A090 00C4 / 00C8 / 00CC / 00D0 / 00D4 / 00D8 / 00DC / 00E0 / 00E4 / 00E8 / 00EC / 00F0 / 00F4 / 00F8 / 00FC / 0100 / 0104 / 0108 / 010C / 0110 / 0114 / 0118 / 011C / 0120 / 0124 / 0128 / 012C / 0130 / 0134 / 0138 / 013C / 0140

The Interrupt Vector Address register configures the interrupt vector address for each interrupt level source. There are 32 levels.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Interrupt vector address register value (IVARV)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Interrupt vector address register value (IVARV)															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	IVARV	0x0	**Interrupt Vector Address register value** Provides the interrupt vector address for the specified interrupt level.

Table 176: Interrupt vector address register

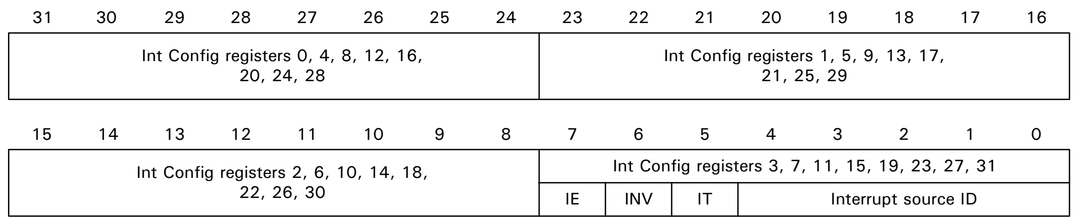
Int (Interrupt) Config (Configuration) registers (0–31)

Address: A090 0144 / 0148 / 014C / 0150 / 0154 / 0158 / 015C / 0160

Each Int Config register is 8 bits in length, and programs each interrupt configuration for each priority level. Table 177 shows how the 32 individual 8-byte registers are mapped in the eight 32-bit registers. Table 178 shows how the bits are assigned in each register, using data bits [07:00] as the example.

Register	[31:24]	[23:16]	[15:08]	[07:00]
A090 0144	Int Config 0	Int Config 1	Int Config 2	Int Config 3
A090 0148	Int Config 4	Int Config 5	Int Config 6	Int Config 7
A090 014C	Int Config 8	Int Config 9	Int Config 10	Int Config 11
A090 0150	Int Config 12	Int Config 13	Int Config14	Int Config 15
A090 0154	Int Config 16	Int Config 17	Int Config 18	Int Config 19
A090 0158	Int Config 20	Int Config 21	Int Config 22	Int Config 23
A090 015C	Int Config 24	Int Config 25	Int Config 26	Int Config 27
A090 0160	Int Config 28	Int Config 29	Int Config 30	Int Config 31

Table 177: Interrupt configuration register address mapping


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
Int Config registers 0, 4, 8, 12, 16, 20, 24, 28
Int Config registers 1, 5, 9, 13, 17, 21, 25, 29
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Int Config registers 2, 6, 10, 14, 18, 22, 26, 30
Int Config registers 3, 7, 11, 15, 19, 23, 27, 31
IE
INV
IT
Interrupt source ID

Register bit assignment

BIts	Access	Mnemonic	Reset	Definition
D07	R/W	IE	0x0	**Interrupt enable** 0 Interrupt is disabled 1 Interrupt is enabled
D06	R	INV	0x0	**Invert** 0 Do not invert the level of the interrupt source. 1 Invert the level of the interrupt source.
D05	R/W	IT	0x0	**Interrupt type** 0 IRQ 1 FIQ
D04:00	R/W	ISD	0x0– 0x1F	**Interrupt source ID** Assign an interrupt ID to each priority level. See "Interrupt sources," beginning on page 268, for the list of interrupt ID numbers.

Table 178: Int Config register

ISRADDR register

Address: A090 0164

The ISRADDR register provides the current ISRADDR value.

The Interrupt Vector Address register for the FIQ interrupt must be assigned a unique value. If this unique address is seen by the IRQ service routine, software must read the ISRADDR register again. The correct IRQ interrupt service routine address is read the second time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Interrupt service routine address (ISRA)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Interrupt service routine address (ISRA)															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	ISRA	0x0	**Interrupt service routine address** ■ A read to this register updates the priority logic block, and masks the current and any lower priority interrupt requests. ■ A write of any value to this register clears the mask, to allow lower priority interrupts to become active.

Table 179: ISRADDR register

Interrupt Status Active

Address: A090 0168

The Interrupt Status Active register shows the current interrupt request.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Interrupt status active (ISA)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Interrupt status active (ISA)															

Register bit assignment

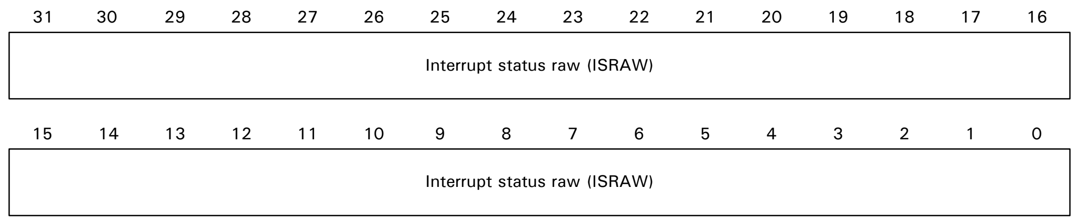
Bits	Access	Mnemonic	Reset	Description
D31:00	R	ISA	0x0	**Interrupt status active** Provides the status of all active, enabled interrupt request levels, where bit 0 is for the interrupt assigned to level 0, bit 1 is for the interrupt assigned to level 1, and so on through bit 31 for the interrupt assigned to level 31.

Table 180: Interrupt Status Active register

Interrupt Status Raw

Address: A090 016C

The Interrupt Status Raw register shows all current interrupt requests.



Register bit assignment

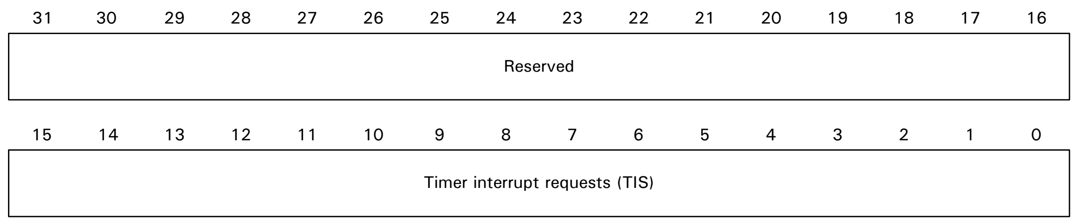
Bits	Access	Mnemonic	Reset	Description
D31:00	R	ISRAW	0x0	**Interrupt status raw** Provides the status of all active, enabled, and disabled interrupt request levels, where bit 0 is for the interrupt assigned to level 0, bit 1 is for the interrupt assigned to level 1, and so on through bit 31 for the interrupt assigned to level 31.

Table 181: Interrupt Status Raw register

Timer Interrupt Status register

Address: A090 0170

The Timer Interrupt Status register shows all current timer interrupt requests.



Register bit assignment

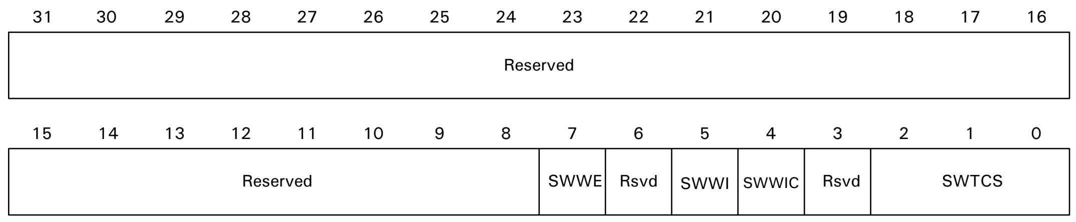
Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A
D15:00	R	TIS	0x0	**Timer interrupt requests, timer 15–timer 0** 0 Inactive 1 Active

Table 182: Timer Interrupt Status register

Software Watchdog Configuration register

Address: A090 0174

The Software Watchdog Configuration register configures the software watchdog timer operation.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A
D07	R/W	SWWE	0x0	**Software watchdog enable** 0 Software watchdog disabled 1 Software watchdog enabled. Once this is set, it cannot be cleared.
D06	N/A	Reserved	N/A	N/A
D05	R/W	SWWI	0x0	**Software watchdog interrupt clear** Write a 1, then a 0 to this bit to clear the software watchdog interrupt.
D04	R/W	SWWIC	0x0	**Software watchdog interrupt response** 0 Generate an interrupt 1 Generate the reset
D03	N/A	Reserved	N/A	**N/A**
D02:00	R/W	SWTCS	0x0	**Software watchdog timer clock select** 000 CPU clock / 2 001 CPU clock / 4 010 CPU clock / 8 011 CPU clock / 16 100 CPU clock / 32 101 CPU clock / 64 110 Reserved 111 Reserved

Table 183: Software Watchdog Configuration register

Software Watchdog Timer register

Address: A090 0178

The Software Watchdog Timer register services the watchdog timer.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Watchdog timer															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Watchdog timer															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	WT	0x0	**Watchdog timer** ■ A *read* to this register gives the current value of the watchdog timer, but will not change the contents. ■ A *write* to the register changes the contents based on the write data value.

Table 184: Software Watchdog Timer register

Clock Configuration register

Address: A090 017C

The Clock Configuration register enables and disables clocks to each module on the AHB bus.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						LPCS			BBC	LCC	MCC	PARBC	PC	Not used	MCC

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:10	N/A	Reserved	N/A	N/A
D09:07	R/W	LPCS	0x0	**LCD panel clock select** 000 AHB clock 001 AHB clock / 2 010 AHB clock / 4 011 AHB clock / 8 1xx LCD clock provided by external clock
D06	R/W	BBC	0x1	**BBus** 0 Clock disabled 1 Clock enabled
D05	R/W	LCC	0x1	**LCD controller** 0 Clock disabled 1 Clock enabled
D04	R/W	MCC	0x1	**Memory controller** 0 Clock disabled 1 Clock enabled
D03	R/W	PARBC	0x1	**PCI arbiter** 0 Clock disabled 1 Clock enabled
D02	R/W	PC	0x1	**PCI** 0 Clock disabled 1 Clock enabled
D01	R/W	Not used	0x0	Must be written to 0.
D00	R/W	MACC	0x1	**Ethernet MAC** 0 Clock disabled 1 Clock enabled

Table 185: Clock Configuration register

Reset and Sleep Control register

Address: A090 0180

The Reset and Sleep Control register resets each module on the AHB bus. To use sleep mode, the CPU must reset and stop the clocks to all modules not used to wake up the CPU. The memory controller must be reset and then re-enabled. The code that resets the memory controller must be loaded into instruction cache first. The last step is to set the CSE bit (D19) in the Reset and Sleep Control register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										BBW	I2CW	CSE	SMWE	EWE	PI3WE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									BBT	LCDC	MEMC	Rsvd	PCIM	Not used	MACM

Register bit assignment

Bits	Access	Mnemonic	Reset	Definition
D31:22	N/A	Reserved	N/A	N/A
D21	R/W	BBW	0x0	**BBus aggregate interrupt wakeup enable** 0 Do not wake up on a BBus aggregate interrupt. 1 Wake up on a BBus aggregate interrupt.
D20	R/W	I2CW	0x0	**I2C interrupt wake up enable** 0 Do not wake up on an I2C interrupt. 1 Wake up on an I2C interrupt.
D19	R/W	CSE	0x0	**CPU sleep enable** System software writes a 1 to this bit to reset and stop the clock to the CPU. Note that software is responsible for stopping the clocks to all other modules before setting this bit. This bit must be cleared after the CPU is woken up, before reentering the sleep state.

Table 186: Reset and Sleep Control register

Bits	Access	Mnemonic	Reset	Definition
D18	R/W	SMWE	0x0	**Serial character match wake-up enable** 0 Do not wake up on receipt of a character match by the serial module. 1 Wake up on receipt of a character match by the serial module.
D17	R/W	EWE	0x0	**Ethernet wake-up enable** 0 Do not wake up on receipt of an Ethernet packet. 1 Wake up on receipt of an Ethernet packet.
D16	R/W	PI3WE	0x0	**PCI interrupt 3 wake-up enable** 0 Do not wake up on PCI interrupt 3 input signal. 1 Wake up on active low PCI interrupt 3 input signal.
D15:07	N/A	Reserved	N/A	N/A
D06	R/W	BBT	0x1	**BBus top** 0 Module reset 1 Module enabled
D05	R/W	LCDC	0x1	**LCD controller** 0 Module reset 1 Module enabled
D04	R/W	MEMC	0x1	**Memory controller** 0 Module reset 1 Module enabled
D03	N/A	Reserved	N/A	N/A
D02	R/W	PCIM	0x1	**PCI module** 0 Module reset 1 Module enabled
D01	R/W	Not used	0x0	Must be written to 0.
D00	R/W	MACM	0x1	**Ethernet MAC** 0 Module reset 1 Module enabled

Table 186: Reset and Sleep Control register

Miscellaneous System Configuration and Status register

Address: A090 0184

The Miscellaneous System Configuration and Status register configures miscellaneous system configuration bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV								Reserved							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		PCIA	Rsvd	BMM	CS1DB	CS1DW		MCCM	PMSS	CS1P	Rsvd	ENDM	MBAR	Rsvd	IRAM0

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:24	R	REV	0x0	**Revision** Indicates the NS9750 hardware identification and revision.
D23:14	N/A	Reserved	N/A	N/A
D13	R	PCIA	0x1	**PCI arbiter configuration** 0 External PCI arbiter 1 Internal PCI arbiter
D12	N/A	Reserved	N/A	N/A
D11	R	BMM	HW strap reset_done	**Bootup memory mode** 0 Boot from SDRAM using SPI serial EEPROM 1 Boot from Flash/ROM on memory chip select 1 Status only; indicates the bootup process.
D10	R	CS1DB	HW strap boot_strap[0]	**Chip select 1 data byte lane configuration HW strap setting** Status bit indicating the hardware strap setting of external memory chip select 1 byte lane/write enable signal configuration. This configuration can be changed by writing to the appropriate control register in the memory controller.

Table 187: Miscellaneous System Configuration and Status register

Bits	Access	Mnemonic	Reset	Description
D09:08	R	CS1DW	HW strap boot_strap[4], boot_strap[3]	**Chip select 1 data width HW strap setting** 00 8 bits 01 16 bits 10 32 bits 11 Reserved Status bits indicating the hardware strap setting of external memory chip select 1 data width. The data width can be changed by writing to the appropriate control register in the memory controller.
D07	R/W	MCCM	HW strap boot_strap[2]	**Memory controller clocking mode HW strap setting** Status bit indicating the hardware strap setting of external memory controller clocking mode. 0 Command delayed mode. Commands are launched on a 90-degree phase-shifted AHB clock, and AHB clock is routed to the external dynamic memory. This option must be used. 1 Clock delayed mode. Reserved for future use.
D06	R	PMSS	HW strap boot_strap[1]	**PCI mode HW strap setting** 0 Card bus mode 1 PCI mode Status bit indicating the hardware strap setting for PCI.
D05	R	CS1P	HW strap gpio[49]	**Chip select 1 polarity HW strap setting** Status bit indicating the hardware strap setting of external memory chip select 1 polarity. The polarity can be changed by writing to the appropriate control registers in the memory controller.
D04	R	Reserved	N/A	N/A
D03	R/W	ENDM	HW strap gpio[44]	**Endian mode** 0 Little endian mode 1 Big endian mode
D02	R/W	MBAR	0x0	**Misaligned bus address response mode** 0 Allow misaligned bus addresses 1 Generate an error response when a misaligned bus address is found; that is, when haddr bits 1 or 0 are not level 0.
D01	N/A	Reserved	N/A	N/A

Table 187: Miscellaneous System Configuration and Status register

Bits	Access	Mnemonic	Reset	Description
D00	R/W	IRAM0	0x0	**Internal register access mode bit 0** 0 Allow access to internal registers using PRIVILEGED mode only 1 Allow access to internal registers using PRIVILEGED or USER mode.

Table 187: Miscellaneous System Configuration and Status register

PLL Configuration register

Address: A090 0188

The PLL Configuration register configures the PLL.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLLBS	PLLFS		PLLIS		PLLND				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLSW	Reserved					PLLBW	FSEL		CPCC		NDSW				

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:26	N/A	Reserved	N/A	N/A
D25	R	PLLBS	HW strap gpio[19]	**PLL bypass status** Status register to determine the powerup strapping settings or the new settings as changed by software.
D24:23	R	PLLFS	HW strap gpio[2], gpio[0]	**PLL FS status [1:0]** Status register to determine the powerup strapping settings or the new settings as changed by software.
D22:21	R	PLLIS	HW strap gpio[24], gpio[22]	**PLL IS status[1:0]** Status register to determine the powerup strapping settings or the new settings as changed by software.

Table 188: PLL Configuration register

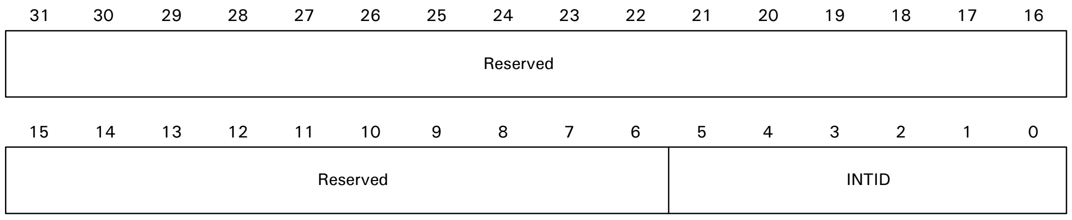
Bits	Access	Mnemonic	Reset	Description
D20:16	R	PLLND	HW strap gpio[17], gpio[12], gpio[10], gpio[8], gpio[4]	**PLL ND status[4:0]** Status register to determine the powerup strapping settings or the new settings as changed by software.
D15	W	PLLSW	0x0	**PLL SW change** Write a 1 to this bit to change the PLL settings as defined in bits D09:00. **Note:** The system is held in reset until the PLL is locked and settled. If the PLL bypass SW bit is set (D09), the PLL setting change and reset is immediate. If the PLL bypass SW bit is not set, the PLL setting change and reset take 4ms to complete.
D14:10	N/A	Reserved	N/A	N/A
D09	R/W	PLLBW	0x0	**PLL bypass SW** 0 PLL operation 1 PLL bypass; use the input reference clock
D08:07	R/W	FSEL	0x0	**PLL frequency select (FS) [1:0]** **PLL** **Output divider value** 00 Divide by 1 01 Divide by 2 10 Divide by 4 11 Divide by 8
D06:05	R/W	CPCC	0x3	**PLL charge pump current control (IS) [1:0]** Recommended settings determined by ND, as follows: **IS** **ND** 00 0–3 01 4–7 10 8–15 11 16–31
D04:00	R/W	NDSW	0x1A	**PLL ND SW [4:0]** PLL multiplier (ND+1).

Table 188: PLL Configuration register

Active Interrupt Level Status register

Address: A090 018C

The Active Interrupt Level Status register shows the current active interrupt level.



Register bit assignment

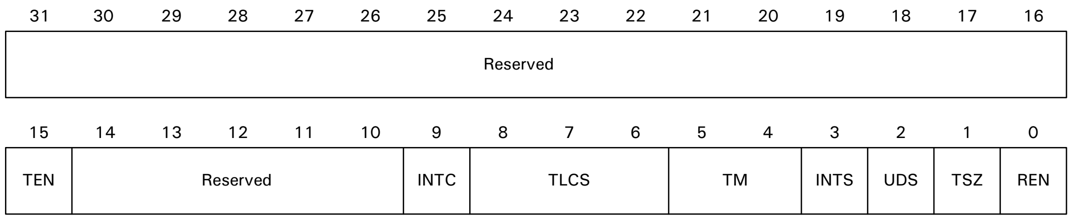
Bits	Access	Mnemonic	Reset	Description
D31:06	N/A	Reserved	N/A	N/A
D05:00	R	INTID	0x0	**Interrupt** The level of the current active interrupt.

Table 189: Active Interrupt Level Status register

Timer 0–15 Control registers

Address: A090 0190 / 0194 / 0198 / 019C / 01A0 / 01A4 / 01A8 / 01AC / 01B0 / 01B4 / 01B8 / 01BC / 01C0 / 01C4 / 01C8 / 01CC

Use the Timer Control registers to select the source clock frequency, as well as other attributes, for each general purpose timer/counter.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
31:16	N/A	Reserved	N/A	N/A
D15	R/W	TEN	0x0	**Timer enable** 0 Timer is disabled 1 Timer is enabled
D14:10	N/A	Reserved	N/A	N/A
D09	R/W	INTC	0x0	**Interrupt clear** Clears the timer interrupt. System software must write a 1, then a 0 to this location to clear the interrupt. If the timer is programmed to halt on terminal count (that is, REN is clear), the software must disable the timer by setting TEN to 0 before clearing the interrupt by writing a 1 and then a 0 to INTC.
D08:06	R/W	TLCS	0x0	**Timer clock select** 000 CPU clock (must be used if this is the high word of two concatenated timers) 001 CPU clock / 2 010 CPU clock / 4 011 CPU clock / 8 100 CPU clock / 16 101 CPU clock / 32 110 CPU clock / 64 111 External pulse event **Notes:** ■ Counting external pulse events, the frequency must be less than one-half the CPU clock frequency. ■ For TLCS settings 000 – 110, the terminal count can be output using GPIO. The terminal count pulse width will be one CPU clock cycle, regardless of the TLCS setting.

Table 190: Timer Control register

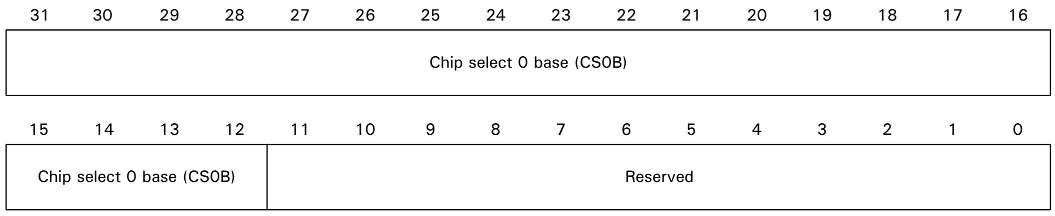
Bits	Access	Mnemonic	Reset	Description
D05:04	R/W	TM	0x0	**Timer mode** 00 Internal timer or external event 01 External low-level, gated timer 10 External high-level, gated timer 11 Concatenate the lower timer. Not applicable on timer 0. **Note:** When either external gated timer option is selected, the timer clock select bits (08:06) determine the frequency.
D03	R/W	INTS	0x0	**Interrupt select** 0 Interrupt disable 1 Generate IRQ
D02	R/W	UDS	0x0	**Up/down select** 0 Up counter 1 Down counter **Note:** When configured as an up counter, the terminal count is 0xFFFF_FFFF. When configured as a down counter, the terminal count is 0x0000_0000.
D01	R/W	TSZ	0x0	**32- or 16-bit timer** 0 16-bit timer 1 32-bit timer
D00	R/W	REN	0x0	**Reload enable** 0 Halt at terminal count. The timer must be disabled, then enabled to reload the timer when the terminal count is reached. The interrupt select (INTS) bit must be cleared during the interrupt service routine when this mode is selected. 1 Reload and resume count at terminal count.

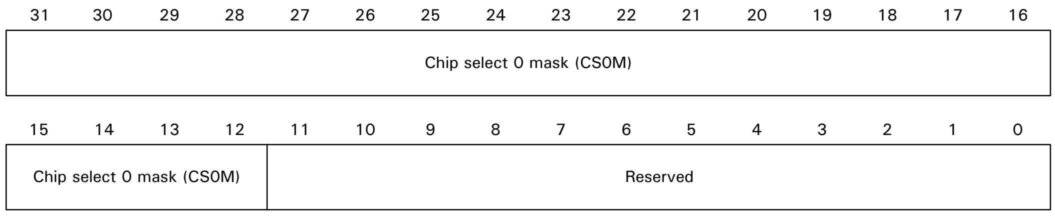
Table 190: Timer Control register

System Memory Chip Select 0 Dynamic Memory Base and Mask registers

Address: A090 01D0 / 01D4

These control registers set the base and mask for system memory chip select 1, with a minimum size of 4K. The powerup default settings produce a memory range of 0x0000 0000 — 0x0FFF FFFF.





Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS0B	0x00000	**Chip select 0 base** Base address for chip select 0 (dynamic).
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS0M	0xF0000	**Chip select 0 mask** Mask or size for chip select 0 (dynamic).
D11:00	N/A	Reserved	N/A	N/A

Table 191: System Memory Chip Select 0 Dynamic Memory Base & Mask registers

System Memory Chip Select 1 Dynamic Memory Base and Mask registers

Address: A090 01D8 / 01DC

These control registers set the base and mask for system memory chip select 1, with a minimum size of 4K. The powerup default settings produce a memory range of 0x1000 0000 — 0x1FFF FFFF.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Chip select 1 base (CS1B)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Chip select 1 base (CS1B)				Reserved											

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Chip select 1 mask (CS1M)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Chip select 1 mask (CS1M)				Reserved											

Register bit assignment

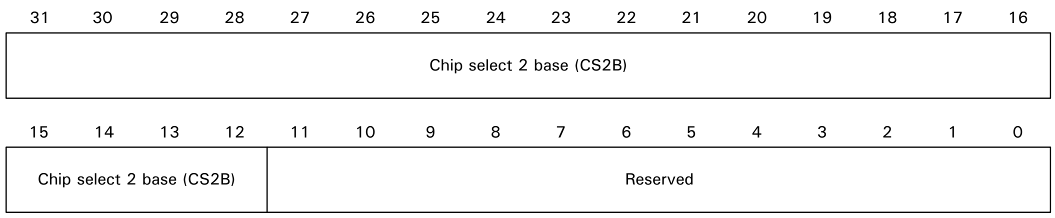
Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS1B	0x10000	**Chip select 1 base** Base address for chip select 1 (dynamic).
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS1M	0xF0000	**Chip select 1 mask** Mask or size for chip select 1 (dynamic).
D11:00	N/A	Reserved	N/A	N/A

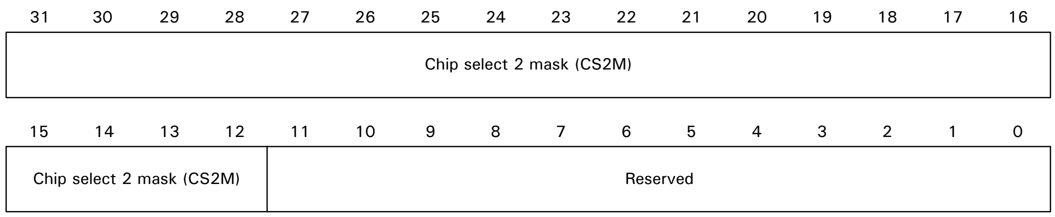
Table 192: System Memory Chip Select 1 Dynamic Memory Base & Mask registers

System Memory Chip Select 2 Dynamic Memory Base and Mask registers

Address: A090 01E0 / 01E4

These control registers set the base and mask for system memory chip select 6, with a minimum size of 4K. The powerup default settings produce a memory range of 0x2000 0000 — 0x2FFF FFFF.





Register bit assignment

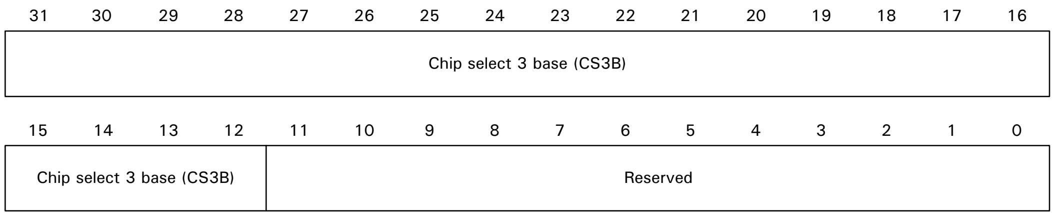
Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS2B	0x20000	**Chip select 2 base** Base address for chip select 2 (dynamic).
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS2M	0xF0000	**Chip select 2 mask** Mask or size for chip select 2 (dynamic).
D11:00	N/A	Reserved	N/A	N/A

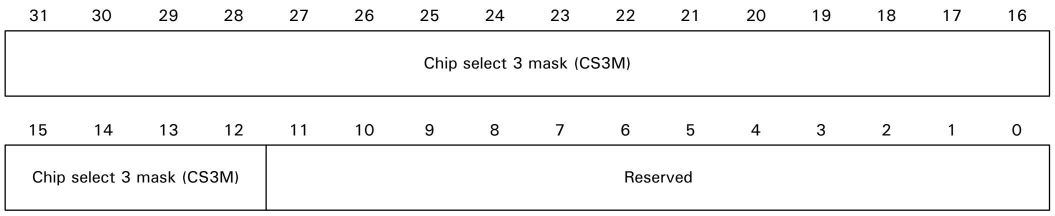
Table 193: System Memory Chip Select 2 Dynamic Memory Base & Mask registers

System Memory Chip Select 3 Dynamic Memory Base and Mask registers

Address: A090 01E8 / 01EC

These control registers set the base and mask for system memory chip select 7, with a minimum size of 4K. The powerup default settings produce a memory range of 0x3000 0000 — 0x3FFF FFFF.





Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS3B	0x30000	**Chip select 3 base** Base address for chip select 3 (dynamic).
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS3M	0xF0000	**Chip select 3 mask** Mask or size for chip select 3 (dynamic).
D11:00	N/A	Reserved	N/A	N/A

Table 194: System Memory Chip Select 3 Dynamic Memory Base & Mask registers

System Memory Chip Select 0 Static Memory Base and Mask registers

Address: A090 01F0 / 01F4

These control registers set the base and mask for system memory chip select 0, with a minimum size of 4K. The powerup default settings produce a memory range of 0x4000 0000 — 0x4FFF FFFF.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Chip select 0 base (CS0B)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Chip select 0 base (CS0B)				Reserved											

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Chip select 0 mask (CS0M)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Chip select 0 mask (CS0M)				Reserved											

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS0B	0x40000	**Chip select 0 base** Base address for chip select 0 (static).
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS0M	0xF0000	**Chip select 0 mask** Mask or size for chip select 0 (static).
D11:00	N/A	Reserved	N/A	N/A

Table 195: System Memory Chip Select 0 Static Memory Base & Mask registers

System Memory Chip Select 1 Static Memory Base and Mask registers

Address: A09001F8 / 01FC

These control registers set the base and mask for system memory chip select 1, with a minimum size of 4K. The powerup default settings produce a memory range of 0x5000 0000 — 0x5FFF FFFF.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Chip select 1 base (CS1B)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Chip select 1 base (CS1B)				Reserved											

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Chip select 1 mask (CS1M)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Chip select 1 mask (CS1M)				Reserved											

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS1B	0x50000	**Chip select 1 base** Base address for chip select 1 (static).
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS1M	0xF0000	**Chip select 1 mask** Mask or size for the chip select 1 (static).
D11:00	N/A	Reserved	N/A	N/A

Table 196: System Memory Chip Select 1 Memory Base and Mask registers

System Memory Chip Select 2 Static Memory Base and Mask registers

Address: A090 0200 / 0204

These control registers set the base and mask for system memory chip select 2, with a minimum size of 4K. The powerup default settings produce a memory range of 0x6000 0000 — 0x6FFF FFFF.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Chip select 2 base (CS2B)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Chip select 2 base (CS2B)				Reserved											

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Chip select 2 mask (CS2M)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Chip select 2 mask (CS2M)				Reserved											

Register bit assignment

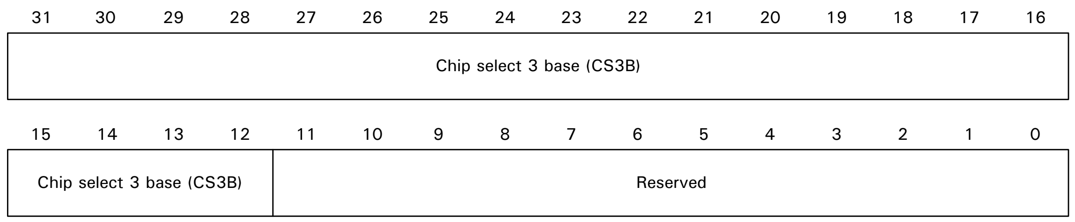
Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS2B	0x60000	**Chip select 2 base** Base address for chip select 2 (static).
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS2M	0xF0000	**Chip select 2 mask** Mask or size for chip select 2 (static).
D11:00	N/A	Reserved	N/A	N/A

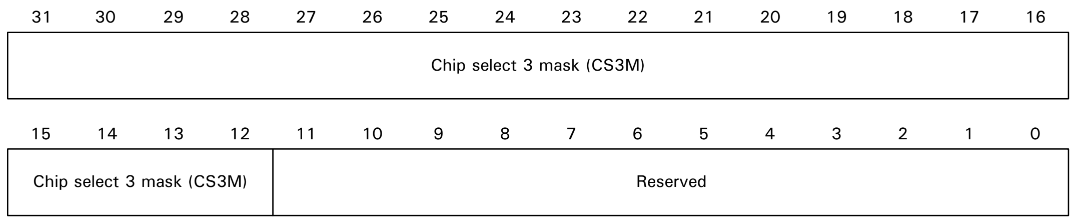
Table 197: System Memory Chip Select 2 Static Memory Base & Mask registers

System Memory Chip Select 3 Static Memory Base and Mask registers

Address: A090 0208 / 020C

These control registers set the base and mask for system memory chip select 3, with a minimum size of 4K. The powerup default settings produce a memory range of 0x7000 0000 — 0x7FFF FFFF.





Register bit assignment

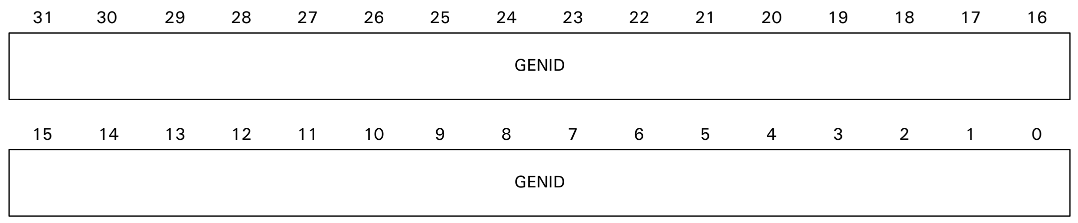
Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS3B	0x70000	**Chip select 3 base** Base address for chip select 3 (static).
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS3M	0xF0000	**Chip select 3 mask** Mask or size for chip select 3 (static).
D11:00	N/A	Reserved	N/A	N/A

Table 198: System Memory Chip Select 3 Static Memory Base & Mask registers

Gen ID register

Address: A090 0210

This register is read-only, and indicates the state of GPIO pins at powerup.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R	GENID	Reflects the status of the GPIO inputs at reset. The GPIO signals are listed in "Bootstrap initialization," beginning on page 272.	**GenID** General Purpose ID register

Table 199: General Purpose ID register

External Interrupt 0–3 Control register

Address: A090 0214 / 0218 / 021C / 0220

The External Interrupt Control registers control the behavior of external interrupts 0–3. The external interrupts are behind GPIO (see "GPIO MUX," beginning on page 34).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												STS	CLR	PLTY	LVEDG

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	N/A	Reserved	N/A	N/A
D03	R	STS	N/A	**Status** Status of the external signal before edge detect or level conversion.
D02	R/W	CLR	0x0	**Clear** Write a 1, then a 0 to this bit to clear the interrupt generated by the edge detect circuit.
D01	R/W	PLTY	0x0	**Polarity** 0 If level-sensitive, the input source is active high. If edge-sensitive, generate an interrupt on the rising edge of the external interrupt. 1 If level-sensitive, the input source is active low. The level is inverted before sending to the interrupt controller. If edge-sensitive, generate an interrupt on the falling edge of the external interrupt.
D00	R/W	LVEDG	0x0	**Level edge** 0 Level-sensitive interrupt 1 Edge-sensitive interrupt

Table 200: External Interrupt 0–3 Control register

Ethernet Communication Module

C H A P T E R 6

The Ethernet Communication module consists of an Ethernet Media Access Controller (MAC) and Ethernet front-end module. The Ethernet MAC interfaces to an external PHY through one of two industry-standard interfaces: MII and RMII. The Ethernet front-end module provides all of the control functions to the MAC.

Overview

The Ethernet MAC module provides the following:

- Station address logic (SAL)
- Statistics module
- Interface to MII (Media Independent Interface) PHY
- Interface to RMII (Reduced Media Independent Interface) PHY

The Ethernet front-end module does the following:

- Provides control functions to the MAC
- Buffers and filters the frames received from the MAC
- Pumps transmit data into the MAC
- Moves frames between the MAC and the system memory
- Reports transmit and receive status to the host

"Legend"

RX_RD = Receive read

RX_WR = Receive write

TX_RD = Transmit read

TX_WR = Transmit write

Figure 63 shows the Ethernet Communications module.

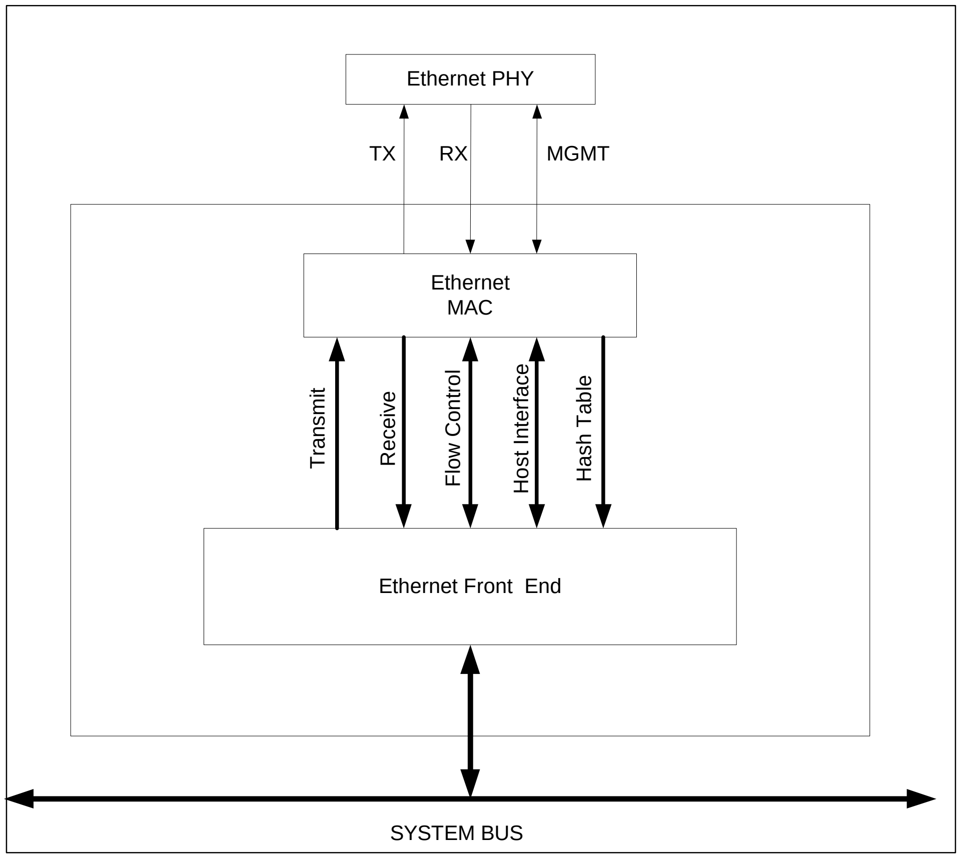


Figure 63: Ethernet Communication module block diagram

Ethernet MAC

The Ethernet MAC includes a full function 10/100 Mbps Media Access Controller (MAC), station address filtering logic (SAL), statistic collection module (STAT), and two software-selectable PHY interfaces — MII and RMII. Figure 64 shows the Ethernet MAC module block diagram, with its associated hierarchy. Table 202 describes the module's features.

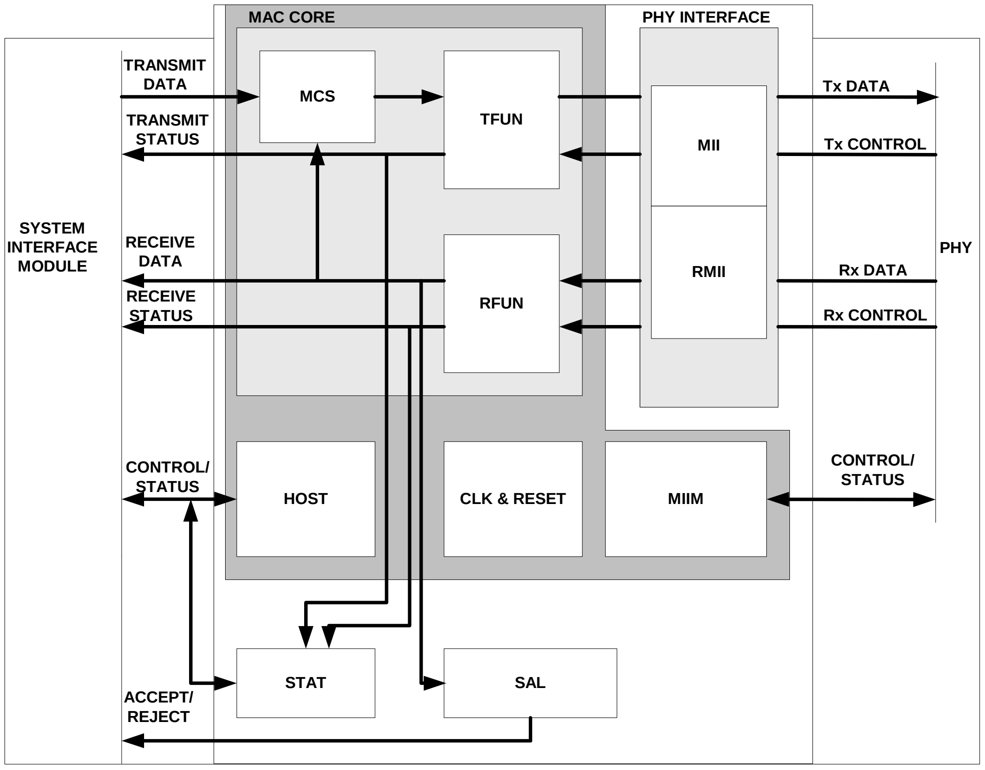


Figure 64: Ethernet MAC block diagram

Feature	Description
MAC Core	**10/100 megabit Media Access Controller** Performs the CSMA/CD function. ■ MCS: MAC control sublayer ■ TFUN: Transmit function ■ RFUN: Receive function
HOST	**Host interface** Provides an interface for control and configuration.
CLK & Reset	**Clocks & resets** Provides a central location for clock trees and reset logic.

Table 201: Ethernet MAC features

Feature	Description
MIIM	**MII management** Provides control/status path to MII and RMII PHYs.
STAT	**Statistics module** Counts and saves Ethernet statistics.
SAL	**Station address logic** Performs destination address filtering.
MII	**Media Independent Interface** Provides the interface from the MAC core to a PHY that supports the MII (as described in the IEEE 802.3 standard).
RMII	**Reduced Media Independent Interface** Provides the interface from the MAC core to a PHY that supports RMII. **Advisory:** Note that the NS9750 RMII interface incorrectly handles packets with dribble. (Dribble occurs when extra data is detected on the end of a packet, but there is insufficient data to form a new byte.) In some cases, packets with dribble will be passed through with the extra data truncated; this is the correct handling, and the packet is treated as a normal packet without error. In other cases, packets with dribble will be passed through with an extra byte at the end. In these situations, the packet is rejected correctly because it appears to have an invalid FCS. In addition, the dribble bit (RXDR) in the status field of RX Ethernet packets and in the Ethernet Receive Status register (see "Ethernet Receive Status register" on page 347) may be falsely set for packets that do not have any dribble bits. For RMII, the dribble bit should be ignored. For RMII, ignore the Receive Alignment Error Counter and the Receive FCS Error Counter. Set the M1RFC and M1RAL bits in the Carry Register 1 Mask register to 1 so no interrupts will be caused when these counters overflow.

Table 201: Ethernet MAC features

Table 202 shows how the different PHY interfaces are mapped to the external IO. In addition to these signals, NS9750 has a dedicated interrupt input for the external PHY (enet_phy_int).

External IO	MII	RMII
RXD[3]	RXD[3]	N/C Pull low external to NS9750
RXD[2]	RXD[2]	N/C Pull low external to NS9750
RXD[1]	RXD[1]	RXD[1]
RXD[0]	RXD[0]	RXD[0]
RX_DV	RX_DV	N/C Pull low external to NS9750
RX_ER	RX_ER	RX_ER Optional signal; pull low external to NS9750 if not being used
RX_CLK	RX_CLK	REF_CLK
TXD[3]	TXD[3]	N/C
TXD[2]	TXD[2]	N/C
TXD[1]	TXD[1]	TXD[1]
TXD[0]	TXD[0]	TXD[0]
TX_EN	TX_EN	TX_EN
TX_ER	TX_ER	N/C
TX_CLK	TX_CLK	N/C Pull low external to NS9750
CRS	CRS	CRS_DV
COL	COL	N/C Pull low external to NS9750
MDC	MDC	MDC
MDIO	MDIO	MDIO

Table 202: PHY interface mappings to external IO

Station address logic (SAL)

The station address logic module examines the destination address field of incoming frames, and filters the frames before they are stored in the Ethernet front-end module. The filtering options, listed next, are programmed in the Station Address Filter register (see page 366).

- Accept frames to destination address programmed in the SA1, SA2, and SA3 registers (Station Address registers, beginning on page 364)
- Accept all frames
- Accept all multicast frames
- Accept all multicast frames using HT1 and HT2 registers (Hash Table registers, beginning on page 366)
- Accept all broadcast frames

The filtering conditions are independent of each other; for example, the Station Address Logic register can be configured to accept all broadcast frames, and frames to the programmed destination address.

The MAC receiver provides the station address logic with a 6-bit CRC value that is the upper 6 bits of a 32-bit CRC calculation performed on the 48-bit multicast destination address. This 6-bit value addresses the 64-bit multicast hash table created in the HT1 and HT2 registers (see "Register Hash Tables" on page 366). If the current receive frame is a multicast frame and the 6-bit CRC addresses a bit in the hash table that is set to 1, the receive frame is accepted; otherwise, the frame is rejected. See "Sample hash table code," beginning on page 397, for sample C code to calculate hash table entries.

Statistics module

The Statistics module counts and saves Ethernet statistics in several counters (see "Statistics registers" on page 368).

The Ethernet General Control Register #2 contains three statistics module configuration bits:

- **AUTOZ.** Enable statistics counter clear on read.
- **CLRCNT.** Clear statistics counters.
- **STEN.** Enable statistics counters.

If any of the counters roll over, an associated carry bit is set in the Carry 1 (CAR1) or Carry 2 (CAR2) registers (see "General Statistics registers," beginning on page 377). Any statistics counter overflow can cause the STOVFL bit in the Ethernet Interrupt Status register (see page 385) to be set if its associated mask bit is not set in Carry Mask Register 1 or Carry Mask Register 2 (see "General Statistics registers," beginning on page 377).

The counters support a *clear on read* capability that is enabled when AUTOZ is set to 1 in the Ethernet General Control Register #2.

Ethernet front-end module

Figure 65 shows the Ethernet front-end module (EFE).

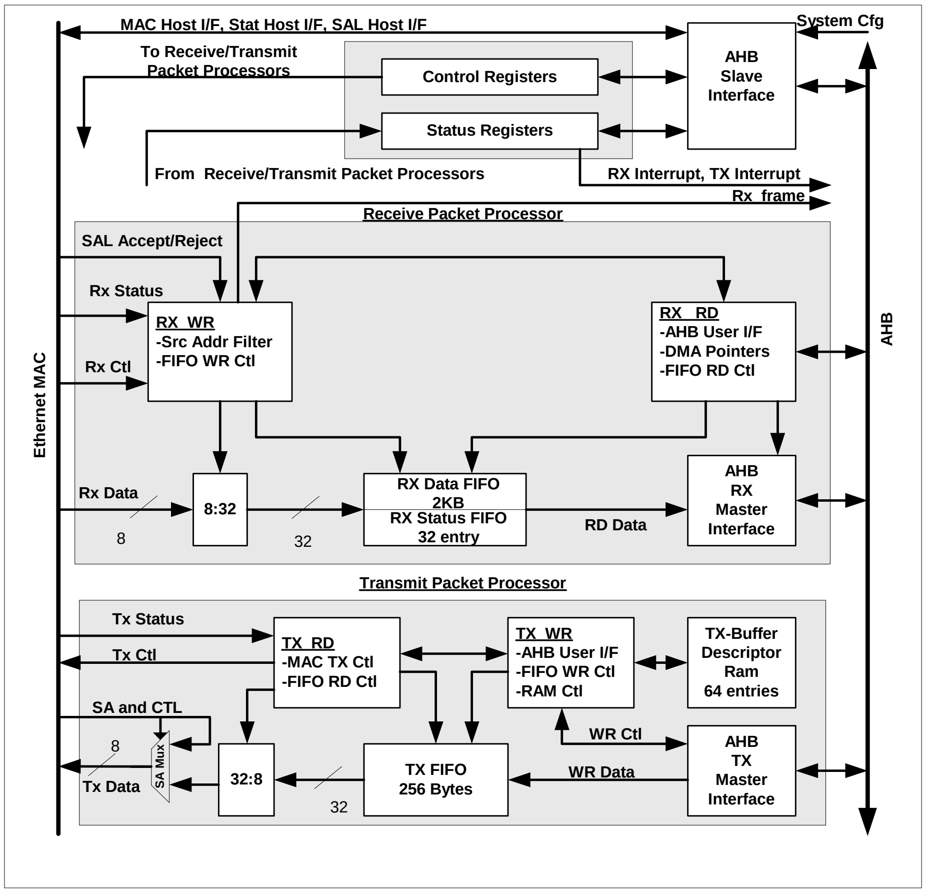

MAC Host I/F, Stat Host I/F, SAL Host I/F
System Cfg
To Receive/Transmit Packet Processors
Control Registers
Status Registers
AHB Slave Interface
From Receive/Transmit Packet Processors
RX Interrupt, TX Interrupt
Rx_frame
Receive Packet Processor
SAL Accept/Reject
Rx Status
Rx Ctl
RX_WR -Src Addr Filter -FIFO WR Ctl
RX_RD -AHB User I/F -DMA Pointers -FIFO RD Ctl
AHB
Ethernet MAC
Rx Data
8
8:32
32
RX Data FIFO 2KB
RX Status FIFO 32 entry
RD Data
AHB RX Master Interface
Transmit Packet Processor
Tx Status
Tx Ctl
TX_RD -MAC TX Ctl -FIFO RD Ctl
TX_WR -AHB User I/F -FIFO WR Ctl -RAM Ctl
TX-Buffer Descriptor Ram 64 entries
SA and CTL
8
SA Mux
Tx Data
32:8
32
TX FIFO 256 Bytes
WR Ctl
WR Data
AHB TX Master Interface

Figure 65: Ethernet front-end module block diagram

The EFE module includes a set of control and status registers, a receive packet processor, and a transmit packet processor. On one side, the Ethernet front end interfaces to the MAC and provides all control and status signals required by the MAC. On the other side, the Ethernet front end interfaces to the system.

The receive packet processor accepts good Ethernet frames (for example, valid checksum and size) from the Ethernet MAC and commits them to external system memory. Bad frames (for example, invalid checksum or code violation) and frames with unacceptable destination addresses are discarded.

The 2K byte RX_FIFO allows the entire Ethernet frame to be buffered while the receive byte count is analyzed. The receive byte count is analyzed by the receive packet processor to select the optimum-sized buffer for transferring the received frame to system memory. The processor can use one of four different-sized receive buffers in system memory.

The transmit packet processor transfers frames constructed in system memory to the Ethernet MAC. The software initializes a buffer descriptor table in a local RAM that points the transmit packet processor to the various frame segments in system memory. The 256-byte TX_FIFO decouples the data transfer to the Ethernet MAC from the AHB bus fill rate.

Receive packet processor

As a frame is received from the Ethernet MAC, it is stored in the receive data FIFO. At the end of the frame, an accept/reject decision is made based on several conditions. If the packet is rejected, it is essentially flushed from the receive data FIFO.

If a frame is accepted, status signals from the MAC, including the receive size of the frame, are stored in a separate 32-entry receive status FIFO; the RX_RD logic is notified that a good frame is in the FIFO.

If the RX_WR logic tries to write a full receive data FIFO anytime during the frame, it flushes the frame from the receive data FIFO and sets RXOVFL_DATA (RX data FIFO overflowed) in the Ethernet Interrupt Status register. For proper operation, reset the receive packet processor using the ERX bit in the Ethernet General Control Register #1 when this condition occurs. If the RX_WR logic tries to write a full receive status FIFO at the end of the frame, the RX_WR logic flushes the frame from the receive data FIFO and sets RXOVFL_STAT (RX status FIFO overflowed) in the Ethernet Interrupt Status register.

Power down mode

The RX_WR logic supports the NS9750's system power down and recovery functionality. In this mode, the RX clock to the MAC and the RX_WR logic are still active, but the clock to the RX_RD and AHB interface is disabled. This allows frames to be received and written into the receive FIFO, but the frame remains in the FIFO until the system wakes up. Normal frame filtering is still performed.

When a qualified frame is inserted into the receive FIFO, the receive packet processor notifies the system power controller, which performs the *wake up* sequence. The frame remains in the receive FIFO until the system wakes up.

Transferring a frame to system memory

The RX_RD logic manages the transfer of a frame in the RX_FIFO to system memory. The transfer is enabled by setting the ERXDMA (enable receive DMA) bit in Ethernet General Control Register #1.

Transferring a frame in the receive FIFO to system memory begins when the RX_WR logic notifies the RX_RD logic that a good frame is in the receive FIFO. Frames are transferred to system memory using up to four rings (that is, 1, 2, or 3 rings can also be used) of buffer descriptors that point to buffers in system memory. The maximum frame size that each ring can accept is programmable. The first thing the RX_RD logic does, then, is analyze the frame length in the receive status FIFO to determine which buffer descriptor to use.

The RX_RD logic goes through the four buffer descriptors looking for the optimum buffer size. It searches the enabled descriptors starting with A, then B, C, and finally D; any pools that are full (that is, the F bit is set in the buffer descriptor) are skipped. The search stops as soon as the logic encounters an available buffer that is large enough to hold the entire receive frame.

The pointers to the first buffer descriptor in each of the four pools are found in the related Buffer Descriptor Pointer register (RXAPTR, RXBPTR, RXCPTR, RXDPTR). Pointers to subsequent buffer descriptors are generated by adding an offset of 0x10 from this pointer for each additional buffer used.

Figure 66 shows the format of the buffer descriptors. The current buffer descriptor for each pool is kept in local registers. The current buffer descriptor registers are initialized to the buffer descriptors pointed to by the Buffer Descriptor Pointer registers, by setting the ERXINIT (enable initialization of RX buffer descriptor registers) bit in Ethernet General Control Register #1. The initialization process is complete when RXINIT (RX initialization complete) is set in the Ethernet General Status register. At the end of a frame, the next buffer descriptor for the ring just used is read from system memory and stored in the registers internal to the RX_RD logic.

	31	30	29	28		16	15		0
OFFSET + 0	Source Address								
OFFSET + 4	Buffer Length (11 lower bits used)								
OFFSET + 8	Destination Address (not used)								
OFFSET + C	W	I	E	F	Reserved		Status		

Figure 66: Receive buffer descriptor format

Field	Description
W	WRAP bit, which, when set, tells the RX_RD logic that this is the last buffer descriptor in the ring. In this situation, the next buffer descriptor is found using the appropriate Buffer Descriptor Pointer register. When the WRAP bit is not set, the next buffer descriptor is found using an offset of 0x10 from the current buffer descriptor pointer.
I	When set, tells the RX_RD logic to set RXBUFC in the Ethernet Interrupt Status register (see page 385) after the frame has been transferred to system memory.
E	ENABLE bit, which, when set, tells the RX_RD logic that this buffer descriptor is enabled. When a new frame is received, pools that do not have the E bit set in their next buffer descriptor are skipped when deciding in which pool to put the frame. The receive processor can use up to four different-sized receive buffers in system memory.
Buffer pointer	32-bit pointer to the start of the buffer in system memory. This pointer must be aligned on a 32-bit boundary.
Status	Lower 16 bits of the Ethernet Receive Status register. The status is taken from the receive status FIFO and added to the buffer descriptor after the last word of the frame is written to system memory.
F	When set, indicates the buffer is full. The RX_RD logic sets this bit after filling a buffer. The system software clears this bit, as required, to free the buffer for future use. When a new frame is received, pools that have the F bit set in their next buffer descriptor are skipped when deciding in which pool to put the frame.

Field	Description
Buffer length	This is a dual use field: ■ When the buffer descriptor is read from system memory, buffer length indicates the maximum sized frame, in bytes, that can be stored in this buffer ring. ■ When the RX_RD logic writes the descriptor back from the receive status FIFO into system memory at the end of the frame, the buffer length is the actual frame length, in bytes.Only the lower 11 bits of this field are valid, since the maximum legal frame size for Ethernet is 1522 bytes.

Transmit packet processor

Transmit frames are transferred from system memory to the transmit packet processor into a 256-byte TX_FIFO. Because various parts of the transmit frame can reside in different buffers in system memory, several buffer descriptors can be used to transfer the frame.

All buffer descriptors (that is, up to 64) are found in a local TX buffer descriptor RAM. Figure 67 shows the transmit buffer descriptor format.

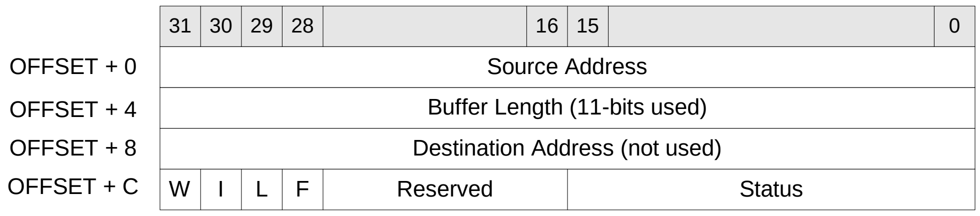


Figure 67: Transmit buffer descriptor format

Field	Description
W	WRAP bit, which, when set, tells the TX_WR logic that this is the last buffer descriptor within the continuous list of descriptors in the TX buffer descriptor RAM. The next buffer descriptor is found using the initial buffer descriptor pointer in the TX Buffer Descriptor Pointer register (TXPTR; see "TX Buffer Descriptor Pointer register," beginning on page 389). When the WRAP bit is not set, the next buffer descriptor is located at the next entry in the TX buffer descriptor RAM.

Field	Description
I	When set, tells the TX_WR logic to set TXBUFC in the Ethernet Interrupt Status register (see page 385) when the buffer is closed due to a normal channel completion.
Buffer pointer	32-bit pointer to the start of the buffer in system memory. This pointer can be aligned on any byte of a 32-bit word.
Status	Lower 16 bits of the Ethernet Transmit Status register. The status is returned from the Ethernet MAC at the end of the frame and written into the last buffer descriptor of the frame.
L	When set, tells the TX_WR logic that this buffer descriptor is the last descriptor that completes an entire frame. This bit allows multiple descriptors to be chained together to make up a frame.
F	When set, indicates the buffer is full. The TX_WR logic clears this bit after emptying a buffer. The system software sets this bit as required, to signal that the buffer is ready for transmission. If the TX_WR logic detects that this bit is not set when the buffer descriptor is read, it does one of two things: ■ If a frame is not in progress, the TX_WR logic sets the TXIDLE bit in the Ethernet Interrupt Status register. ■ If a frame is in progress, the TXBUFNR bit in the Ethernet Interrupt Status register is set. In either case, the TX_WR logic stops processing frames until TCLER (clear transmit logic) in Ethernet General Control Register #2 is toggled from low to high. TXBUFNR is set only for frames that consist of multiple buffer descriptors and contain a descriptor — *not* the first descriptor — that does not have the F bit set after frame transmission has begun.
Buffer length	This is a dual use field: ■ When the buffer descriptor is read from the TX buffer descriptor RAM, buffer length indicates the length of the buffer, in bytes. The TX_WR logic uses this information to identify the end of the buffer. For proper operation of the TX_WR logic, all transmit frames must be at least 34 bytes in length. ■ When the TX_WR logic updates the buffer descriptor at the end of the frame, it writes the length of the frame, in bytes, into this field for the last buffer descriptor of the frame. If the MAC is configured to add the CRC to the frame (that is, CRCEN in MAC Configuration Register #2 is set to 1), this field will include the four bytes of CRC. This field is set to 0x000 for jumbo frames that are aborted (see "TXAJ" on page 346) Only the lower 11 bits of this field are valid, since the maximum legal frame size for Ethernet is 1522 bytes.

Setting the EXTDMA (enable transmit DMA) bit in Ethernet General Control Register #1 starts the transfer of transmit frames from the system memory to the TX_FIFO. The TX_WR logic reads the first buffer descriptor in the TX buffer descriptor RAM.

- If the F bit is set, it transfers data from system memory to the TX_FIFO using the buffer pointer as the starting point. This process continues until the end of the buffer is reached. The address for each subsequent read of the buffer is incremented by 32 bytes (that is, 0x20). The buffer length field in the buffer descriptor is decremented by this same value, each transfer, to identify when the end of the buffer is reached.
- If the L field in the buffer descriptor is 0, the next buffer descriptor in the RAM continues the frame transfer until the L field in the current buffer descriptor is 1. This identifies the current buffer as the last buffer of a transmit frame.

After the entire frame has been written to the TX_FIFO, the TX_WR logic waits for a signal from the TX_RD logic indicating that frame transmission has completed at the MAC. The TX_WR logic updates the buffer length, status, and F fields of the current buffer descriptor (that is, the last buffer descriptor for the frame) in the TX buffer descriptor RAM when the signal is received.

The TX_WR logic examines the status received from the MAC after it has transmitted the frame.

- If the frame was transmitted successfully, the TX_WR logic sets TXDONE (frame transmission complete) in the Ethernet Interrupt Status register and reads the next buffer descriptor. If a new frame is available (that is, the F bit is set), the TX_WR starts transferring the frame. If a new frame is not available, the TX_WR logic sets the TXIDLE (TX_WR logic has no frame to transmit) bit in the Ethernet Interrupt Status register and waits for the software to toggle TCLER (clear transmit logic), in Ethernet General Control Register #2, from low to high to resume processing. When TCLER is toggled, transmission starts again with the buffer descriptor pointed to by the Transmit Recover Buffer Descriptor Pointer register. Software should update this register before toggling TCLER.

  The Transmit Buffer Descriptor Pointer Offset register will be valid only if the previous buffer completed normally. In the case of an error that requires that software manually throw away a packet by clearing out buffer descriptors, the Transmit Buffer Descriptor Pointer Offset register will not contain the correct value. In this situation, software must keep track of the location of the next buffer descriptor to be kicked off.

- If the TX_WR logic detects that the frame was aborted or had an error, the logic updates the current buffer descriptor as described in the previous paragraph. If the frame was aborted before the last buffer descriptor of the frame was accessed, the result is a situation in which the status field of a buffer descriptor, which is not the last buffer descriptor in a frame, has a non-zero value. The TX_WR logic stops processing frames until TCLER (clear transmit logic) in Ethernet General Control Register #2 is toggled from low to high to resume processing. The TX_WR logic also sets TXERR (last frame not transmitted successfully) in the Ethernet Interrupt Status register and loads the TX buffer descriptor RAM address of the current buffer descriptor in the TX Error Buffer Descriptor Pointer register (see page 390). This allows identification of the frame that was not transmitted successfully. As part of the recovery procedure, software must read the TX Error Buffer Descriptor Pointer register and then write the 8-bit address of the buffer descriptor to resume transmission into the Transmit Recover Buffer Descriptor Pointer register.

Transmitting a frame to the Ethernet MAC

The TX_RD logic is responsible for reading data from the TX_FIFO and sending it to the Ethernet MAC. The logic does not begin reading a new frame until the TX_FIFO is full. This scheme decouples the data transfer to the Ethernet MAC from the fill rate from the AHB bus. For short frames that are less than 256 bytes, the transmit process begins when the end-of-frame signal is received from the TX_WR logic.

When the MAC completes a frame transmission, it returns status bits that are stored in the Ethernet Transmit Status register (see page 344) and written into the status field of the current buffer descriptor.

Ethernet Slave Interface

The AHB slave interface supports only single 32-bit transfers. The slave interface also supports limiting CSR and RAM accesses to CPU “privileged mode” accesses. Use the internal register access mode bit 0 in the Miscellaneous System Configuration register to set access accordingly (see "Miscellaneous System Configuration and Status register," beginning on page 296).

The slave also generates an AHB ERROR if the address is not aligned on a 32-bit boundary, and the misaligned bus address response mode is set in the Miscellaneous System Configuration register (see "Miscellaneous System Configuration and Status register," beginning on page 296). In addition, accesses to non-existent addresses result in an AHB ERROR response.

Interrupts

Separate RX and TX interrupts are provided back to the system. Table 203 shows all interrupt sources and the interrupts to which they are assigned.

Interrupt condition	Description	Interrupt
RX data FIFO overflow	RX data FIFO overflowed. For proper operation, reset the receive packet processor using the ERX bit in the Ethernet General Control Register #1 when this condition occurs.	RX
RX status FIFO overflow	RX status overflowed.	RX
Receive buffer closed	I bit set in receive buffer descriptor and buffer closed.	RX
Receive complete (Pool A)	Complete receive frame stored in pool A of system memory.	RX
Receive complete (Pool B)	Complete receive frame stored in pool B of system memory.	RX
Receive complete (Pool C)	Complete receive frame stored in pool C of system memory.	RX
Receive complete (Pool D)	Complete receive frame stored in pool D of system memory.	RX
No receive buffers	No buffer is available for this frame because all 4 buffer rings are disabled, full, or no available buffer is big enough for the frame.	RX
Receive buffers full	No buffer is available for this frame because all 4 buffers are disabled or full.	RX
RX buffer ready	Frame available in RX_FIFO. (Used for diagnostics.)	RX
Statistics counter overflow	One of the statistics counters has overflowed. Individual counters can be masked using the CAM1 and CAM2 registers.	TX
Transmit buffer closed	I bit set in Transmit buffer descriptor and buffer closed.	TX

Table 203: Ethernet interrupt conditions

Interrupt condition	Description	Interrupt
Transmit buffer not ready	F bit not set in transmit buffer descriptor when read from TX buffer descriptor RAM, for a frame in progress.	TX
Transmit complete	Frame transmission complete.	TX
TXERR	Frame not transmitted successfully.	TX
TXIDLE	TX_WR logic in idle mode because there are no frames to send.	TX

Table 203: Ethernet interrupt conditions

The status bits for all interrupts are available in the Ethernet Interrupt Status register, and the associated enables are available in the Ethernet Interrupt Enable register. Each interrupt status bit is cleared by writing a 1 to it.

Resets

Table 204 provides a summary of all resets used for the Ethernet front-end and MAC, as well as the modules the resets control.

Bit field	Register	Active state	Default state	Modules reset
ERX	Ethernet General Control Register #1	0	0	RX_RD, RX_WR
ETX	Ethernet General Control Register #1	0	0	TX_RD, TX_WR
MAC_HRST	Ethernet General Control Register #1	1	0	MAC, STAT, RMII, RX_WR, TX_RD, programmable registers in Station Address Logic
SRST	MAC1	1	1	MAC (except programmable registers), Station Address Logic (except programmable registers), RMII, RX_WR, TX_RD
RPERFUN	MAC1	1	0	MAC RX logic
RPEMCST	MAC1	1	0	MAC PEMCS (TX side)
RPETFUN	MAC1	1	0	MAC TX logic

Table 204: Reset control

Bit field	Register	Active state	Default state	Modules reset
RMIIM	MII Management Configuration register	1	0	MAC MIIM logic
RPERMII	PHY Support register	1	0	RMII

Table 204: Reset control

External CAM filtering

NS9750 supports external Ethernet CAM filtering, which requires an external CAM controller to operate in conjunction with the MAC inside NS9750. The interface to the CAM controller is provided through GPIO in NS9750. External CAM filtering uses these bits:

- GPIO[19] configured as an output and for function 0
- GPIO[18] configured as an input and for function 0

For MII PHYs, the CAM_REQ (GPIO[19]) signal is driven high by NS9750, to identify the beginning of each Ethernet frame being transferred to NS9750. The signal is driven high coincident with the 6th nibble of the packet from the frame. The external CAM hardware must monitor the MII receive interface between the PHY and the MAC waiting for the CAM_REQ assertion. When CAM_REQ is asserted, the CAM hardware can extract the destination address field from the MII receive bus. As an alternative, the external CAM hardware can use the RX_DV signal from the MII PHY to detect the start of a frame.

After performing the necessary destination address lookup, the incoming frame can be rejected by CAM filtering hardware by asserting the CAM_REJECT (GPIO[20]) input high. This signal must be asserted no later than the 4th nibble from the end of the frame. Once it is asserted, it must remain asserted until three RX_CLKs after the end of the frame, to guarantee that the RX_WR logic has captured it. For example, a 64-byte frame contains 128 nibbles of data on the MII interface. CAM_REJECT must be valid by the 123rd nibble of data (first nibble is 0th nibble).

Figure 68 shows the timing relationship between the CAM_REQ, CAM_REJECT, and MII receive interface signals when using an MII PHY.

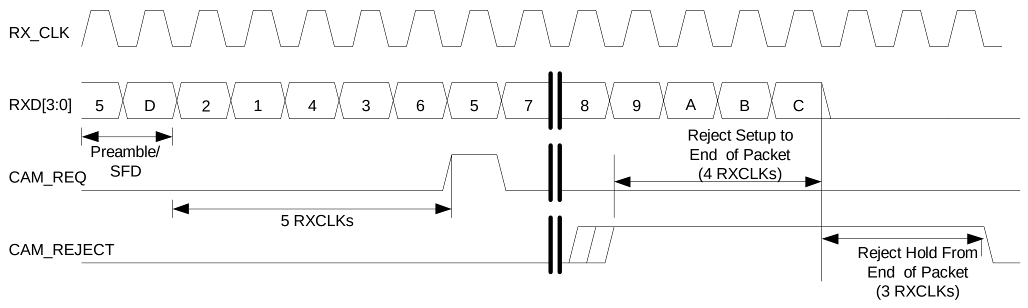

RX_CLK
RXD[3:0]
5
D
2
1
4
3
6
5
7
8
9
A
B
C
Preamble/ SFD
CAM_REQ
5 RXCLKs
Reject Setup to End of Packet (4 RXCLKs)
CAM_REJECT
Reject Hold From End of Packet (3 RXCLKs)

Figure 68: External Ethernet CAM filtering for MII PHY

In this example, the MII receive interface is transferring a frame whose first 6 nibbles have the values 1, 2, 3, 4, 5, and 6. The external CAM hardware uses the CAM_REQ signal to find the alignment for the destination address. After lookup is performed, the CAM hardware can assert the CAM_REJECT signal to discard the frame. The CAM_REJECT signal must be asserted no later than the 4th nibble from the end of the frame.

For RMII PHYs, the external CAM filtering logic is different, because the PHY interface is 2 bits at 50 MHz rather than the 4 bits at 25 MHz for a MII PHY. Because the CAM_REQ signal is generated from the 25 MHz clock, it cannot be used reliably with external 50 MHz logic to identify the start of a new frame. The external logic instead should use the RMII PHY receive interface signals (that is, RXD[1:0], CRS_DV) to find the start of a frame, as shown in Figure 69. The RMII specification defines the start of a frame preamble when CRS_DV is high and RXD[1:0] transitions from 00 to 01. Per the specification, CRS_DV is asserted asynchronously to REF_CLK, to indicate the CRS function. When RXD[1:0] transitions from 00, however, CRS_DV performs the data valid function, and is negated and asserted synchronous to REF_CLK until the end of the frame.

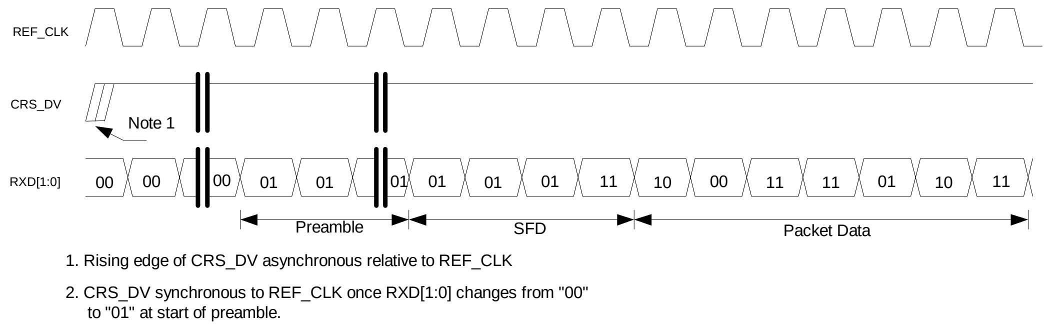

REF_CLK
CRS_DV
Note 1
RXD[1:0]
00 00 00 01 01 01 01 01 01 11 10 00 11 11 01 10 11
Preamble
SFD
Packet Data
1. Rising edge of CRS_DV asynchronous relative to REF_CLK
2. CRS_DV synchronous to REF_CLK once RXD[1:0] changes from "00" to "01" at start of preamble.

Figure 69: RMII PHY receive interface

After performing the necessary destination address lookup, the incoming frame can be rejected by the CAM filtering hardware by asserting the CAM_REJECT(GPIO[20]) input high, as shown in Figure 70. CAM_REJECT(GPIO[20]) must be asserted no later than one di-bit nibble before the end of the frame (that is, when CRS_DV is negated). Once the signal is asserted, it must remain asserted until 16 REF_CLKs (for 100 Mbps) or 128 REF_CLKS (for 10 Mbps) after the end of the frame, to guarantee that the RX_WR logic has captured it. For example, a 64-byte frame contains 256 di-bits (that is, 2 bits) of data on the RMII interface. CAM_REJECT must be valid by the 254th di-bit of data (the first di-bit is 0th di-bit).

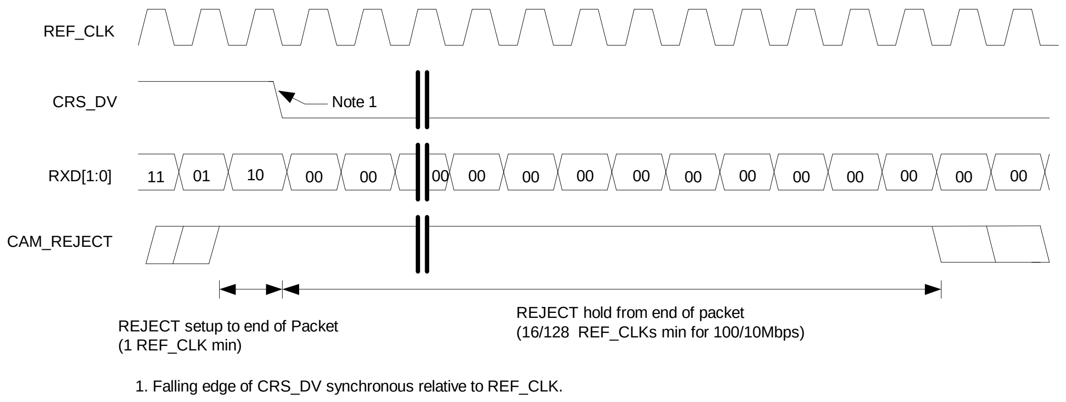

REF_CLK
CRS_DV
Note 1
RXD[1:0]
11 01 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00
CAM_REJECT
REJECT setup to end of Packet
(1 REF_CLK min)
REJECT hold from end of packet
(16/128 REF_CLKs min for 100/10Mbps)
1. Falling edge of CRS_DV synchronous relative to REF_CLK.

Figure 70: External Ethernet CAM filtering for RMII PHY

Ethernet Control and Status registers

Table 205 shows the address for each Ethernet controller register. All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Address	Register	Description
A060 0000	EGCR1	Ethernet General Control Register #1
A060 0004	EGCR2	Ethernet General Control Register #2
A060 0008	EGSR	Ethernet General Status register
A060 000C–A060 0014		Reserved
A060 0018	ETSR	Ethernet Transmit Status register
A060 001C	ERSR	Ethernet Receive Status register
A060 0400	MAC1	MAC Configuration Register #1
A060 0404	MAC2	MAC Configuration Register #2
A060 0408	IPGT	Back-to-Back Inter-Packet-Gap register
A060 040C	IPGR	Non-Back-to-Back Inter-Packet-Gap register
A060 0410	CLRT	Collision Window/Retry register
A060 0414	MAXF	Maximum Frame register
A060 0418	SUPP	PHY Support register
A060 041C	Reserved	
A060 0420	MCFG	MII Management Configuration register
A060 0424	MCMD	MII Management Command register
A060 0428	MADR	MII Management Address register
A060 042C	MWTD	MII Management Write Data register
A060 0430	MRDD	MII Management Read Data register
A060 0434	MIND	MII Management Indicators register
A060 0440	SA1	Station Address Register #1
A060 0444	SA2	Station Address Register #2

Table 205: Ethernet Control and Status register map

Address	Register	Description
A060 0448	SA3	Station Address register #3
A060 0500	SAFR	Station Address Filter register
A060 0504	HT1	Hash Table Register #1
A060 0508	HT2	Hash Table Register #2
A060 0680	STAT	Statistics Register Base (45 registers)
A060 0A00	RXAPTR	RX_A Buffer Descriptor Pointer register
A060 0A04	RXBPTR	RX_B Buffer Descriptor Pointer register
A060 0A08	RXCPTR	RX_C Buffer Descriptor Pointer register
A060 0A0C	RXDPTR	RX_D Buffer Descriptor Pointer register
A060 0A10	EINTR	Ethernet Interrupt Status register
A060 0A14	EINTREN	Ethernet Interrupt Enable register
A060 0A18	TXPTR	TX Buffer Descriptor Pointer register
A060 0A1C	TXRPTR	TX Recover Buffer Descriptor Pointer register
A060 0A20	TXERBD	TX Error Buffer Descriptor Pointer register
A060 0A24	Reserved	
A060 0A28	RXAOFF	RX_A Buffer Descriptor Pointer Offset register
A060 0A2C	RXBOFF	RX_B Buffer Descriptor Pointer Offset register
A060 0A30	RXCOFF	RX_C Buffer Descriptor Pointer Offset register
A060 0A34	RXDOFF	RX_D Buffer Descriptor Pointer Offset register
A060 0A38	TXOFF	Transmit Buffer Descriptor Pointer Offset register
A060 0A3C	RXFREE	RX Free Buffer register
A060 1000	TXBD	TX Buffer Descriptor RAM (256 locations)

Table 205: Ethernet Control and Status register map

Ethernet General Control Register #1

Address: A060 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ERX	ERX DMA	Rsvd	ERX SHT	Not used				ETX	ETX DMA	Not used		ERX INIT	Reserved		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHY_MODE		Rsvd	Not used		RX ALIGN	MAC_ HRST	ITXA	Reserved							

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	ERX	0	**Enable RX packet processing** (see "Receive packet processor" on page 324) 0 Reset RX 1 Enable RX Used as a soft reset for the RX. When cleared, resets all logic in the RX and flushes the FIFO. The ERX bit must be set active high to allow data to be received from the MAC receiver.
D30	R/W	ERXDMA	0	**Enable receive DMA** 0 Disable receive DMA data request (use to stall receiver) 1 Enable receive DMA data request Must be set active high to allow the RX_RD logic to request the AHB bus to DMA receive frames into system memory. Set this bit to zero to temporarily stall the receive side Ethernet DMA. The RX_RD logic stalls on frame boundaries.
D29	N/A	Reserved	N/A	N/A

Table 206: Ethernet General Control Register #1

Bits	Access	Mnemonic	Reset	Description
D28	R/W	ERXSHT	0	**Accept short (<64) receive frames** 0 Do not accept short frames 1 Accept short frames When set, allows frames that are smaller than 64 bytes to be accepted by the RX_WR logic. ERXSHT is typically set for debugging only.
D27:24	R/W	Not used	0	Always write as 0.
D23	R/W	ETX	0	**Enable TX packet processing** (see "Transmit packet processor" on page 327) 0 Reset TX 1 Enable TX Used as a soft reset for the TX. When cleared resets all logic in the TX and flushes the FIFOs. ETX must be set active high to allow data to be sent to the MAC and to allow processor access to the TX buffer descriptor RAM.
D22	R/W	ETXDMA	0	**Enable transmit DMA** 0 Disable transmit DMA data request (use to stall transmitter) 1 Enable transmit DMA data request Must be set active high to allow the transmit packet processor to issue transmit data requests to the AHB interface. Set this bit to 0 to temporarily stall frame transmission, which always stalls at the completion of the current frame. The 8-bit address of the next buffer descriptor to be read in the TX buffer descriptor RAM is loaded into the TXSPTR register when the transmit process ends. If the transmit packet processor already is stalled and waiting for TCLER (see "TCLER" on page 343),clearing ETXDMA will not take effect until TCLER has been toggled. This bit generally should be set after the Ethernet transmit parameters (for example, buffer pointer descriptor) are programmed into the transmit packet processor.
D21	R/W	Not used	1	Always write as 1.
D20	R/W	Not used	0	Always write as 0.

Table 206: Ethernet General Control Register #1

Bits	Access	Mnemonic	Reset	Description
D19	R/W	ERXINIT	0	**Enable initialization of RX buffer descriptors** 0 Do not initialize 1 Initialize When set, causes the RX_RD logic to initialize the internal buffer descriptor registers for each of the four pools from the buffer descriptors pointed to by RXAPTR, RXBPTR, RXCPTR, and RXDPTR. This is done as part of the RX initialization process. RXINIT is set in the Ethernet General Status register (see page 344) when the initialization process is complete, and ERXINIT must be cleared before enabling frame reception from the MAC. The delay from ERXINIT set to RXINIT set is less than five microseconds.
D18:16	N/A	Reserved	N/A	N/A
D15:14	R/W	PHY_MODE	00	**Ethernet interface mode** 00 10/100 Mbit MII mode 01 10/100 Mbit RMII mode 10 Reserved 11 Reserved Identifies what type of Ethernet PHY is attached to NS9750. NS9750 supports two styles of Ethernet PHY: MII and RMII. This field should be changed only while the MAC is reset.
D13	N/A	Reserved	N/A	N/A
D12:11	R/W	Not used	0	Always write as 0.
D10	R/W	RXALIGN	0	**Align RX data** 0 Standard receive format. The data block immediately follows the 14-byte header block. 1 The receiver inserts a 2-byte padding between the 14-byte header and the data block, causing longword alignment for both the header and data blocks.
D09	R/W	MAC_HRST	1	**MAC host interface soft reset** 0 Restore MAC, STAT, SAL, RX_WR, and TX_RD to normal operation. 1 Reset MAC, STAT, programmable registers in SAL, RX_WR, and TX_RD. Keep high for minimum of 5µsec to guarantee that all functions get reset.

Table 206: Ethernet General Control Register #1

Bits	Access	Mnemonic	Reset	Description
D08	R/W	ITXA	0	**Insert transmit source address** 0 Source address for Ethernet transmit frame taken from data in TX_FIFO. 1 Insert the MAC Ethernet source address into the Ethernet transmit frame source address field. Set to force the MAC to automatically insert the Ethernet MAC source address into the Ethernet transmit frame source address. The SA1, SA2, and SA3 registers provide the address information. When the ITXA bit is cleared, the Ethernet MAC source address is taken from the data in the TX_FIFO.
D07:00	N/A	Reserved	N/A	N/A

Table 206: Ethernet General Control Register #1

Ethernet General Control Register #2

Address: A060 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used												T CLER	AUTO Z	CLR CNT	STEN

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	R/W	Not used	0	Always write as 0.

Table 207: Ethernet General Control Register #2

Bits	Access	Mnemonic	Reset	Description
D03	R/W	TCLER	0	**Clear transmit error** 0->1 transition: Clear transmit error. Clears out conditions in the transmit packet processor that have caused the processor to stop and require assistance from software before the processor can be restarted (for example, an AHB bus error or the TXBUFNR bit set in the Ethernet Interrupt Status register (see page 385)). Toggle this bit from low to high to restart the transmit packet processor.
D02	R/W	AUTOZ	0	**Enable statistics counter clear on read** 0 No change in counter value after read 1 Counter cleared after read When set, configures all counters in the Statistics module to clear on read. If AUTOZ is not set, the counters retain their value after a read. The counters can be cleared by writing all zeros.
D01	R/W	CLRCNT	1	**Clear statistics counters** 0 Do not clear all counters 1 Clear all counters When set, synchronously clears all counters in the Statistics module.
D00	R/W	STEN	0	**Enable statistics counters** 0 Counters disabled 1 Counters enabled When set, enables all counters in the Statistics module. If this bit is cleared, the counters will not update.

Table 207: Ethernet General Control Register #2

Ethernet General Status register

Address: A060 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											RX INIT	Reserved			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:21	N/A	Reserved	N/A	N/A
D20	R/C	RXINIT	0x0	**RX initialization complete** Set when the RX_RD logic has completed the initialization of the local buffer descriptor registers requested when ERXINIT in Ethernet General Control Register #1 (see page 339) is set. The delay from ERXINIT set to RXINIT set is less than five microseconds.
D19:00	N/A	Reserved	N/A	N/A

Table 208: Ethernet General Status register

Ethernet Transmit Status register

Address: A060 0018

The Ethernet Status register contains the status for the last transmit frame. The TXDONE bit in the Ethernet Interrupt Status register (see page 385) is set upon completion of a transmit frame and the Ethernet Transmit Status register is loaded at the same time. Bits [15:0] are also loaded into the Status field of the last transmit buffer descriptor for the frame.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX OK	TX BR	TX MC	TX AL	TX AED	TX AEC	TX AUR	TX AJ	Not used	TX DEF	TX CRC	Not used	TXCOLC			

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A
D15	R	TXOK	0x0	**Frame transmitted OK** When set, indicates that the frame has been delivered to and emptied from the transmit FIFO without problems.
D14	R	TXBR	0x0	**Broadcast frame transmitted** When set, indicates the frame's destination address was a broadcast address.
D13	R	TXMC	0x0	**Multicast frame transmitted** When set, indicates the frame's destination address was a multicast address.
D12	R	TXAL	0x0	**TX abort — late collision** When set, indicates that the frame was aborted due to a collision that occurred beyond the collision window set in the Collision Window/Retry register (see page 366). If this bit is set, the TX_WR logic stops processing frames and sets the TXERR bit in the Ethernet Interrupt Status register.
D11	R	TXAED	0x0	**TX abort — excessive deferral** When set, indicates that the frame was deferred in excess of 6071 nibble times in 100 Mbps or 24,287 times in 0 Mbps mode. This causes the frame to be aborted if the *excessive deferral bit* is set to 0 in MAC Configuration Register #2 (see page 351). If TXAED is set, the TX_WR logic stops processing frames and sets the TXERR bit in the Ethernet Interrupt Status register.

Table 209: Ethernet Transmit Status register

Bits	Access	Mnemonic	Reset	Description
D10	R	TXAEC	0x0	**TX abort — excessive collisions** When set, indicates that the frame was aborted because the number of collisions exceeded the value set in the Collision Window/Retry register. If this bit is set, the TX_WR logic stops processing frames and sets the TXERR bit in the Ethernet Interrupt Status register.
D09	R	TXAUR	0x0	**TX abort — underrun** When set, indicates that the frame was aborted because the TX_FIFO had an underrun. If this bit is set, the TX_WR logic stops processing frames and sets the TXERR bit in the Ethernet Interrupt Status register.
D08	R	TXAJ	0x0	**TX abort — jumbo** When set, indicates that the frame's length exceeded the value set in the Maximum Frame register (see page 357). TXAJ is set only if the HUGE bit in MAC Configuration Register #2 (see page 351) is set to 0. Jumbo frames result in the TX buffer descriptor buffer length field (see "Buffer length" on page 327) being set to 0x000. If the HUGE bit is set to 0, the frame is truncated. If TXAJ is set, the TX_WR logic stops processing frames and sets the TXERR bit in the Ethernet Interrupt Status register.
D07	R	Not used	0x0	Always set to 0.
D06	R	TXDEF	0x0	**Transmit frame deferred** When set, indicates that the frame was deferred for at least one attempt, but less than the maximum number for an excessive deferral. TXDEF is also set when a frame was deferred due to a collision. This bit is not set for late collisions.
D05	R	TXCRC	0x0	**Transmit CRC error** When set, indicates that the attached CRC in the frame did not match the internally-generated CRC. This bit is not set if the MAC is inserting the CRC in the frame (that is, the CRCEN bit is set in MAC Configuration Register #2). If TXCRC is set, the TX_WR logic stops processing frames and sets the TXERR bit in the Ethernet Interrupt Status register.

Table 209: Ethernet Transmit Status register

Bits	Access	Mnemonic	Reset	Description
D04	R	Not used	0x0	Always set to 0.
D03:00	R	TXCOLC	0x0	**Transmit collision count** Number of collisions the frame incurred during transmission attempts.

Table 209: Ethernet Transmit Status register

Ethernet Receive Status register

Address: A060 001C

The Ethernet Receive Status register contains the status for the last completed receive frame. The RXBR bit in the Ethernet Interrupt Status register (see page 385) is set whenever a receive frame is completed and the Ethernet Receive Status register is loaded at the same time. Bits [15:0] are also loaded into the status field of the receive buffer descriptor used for the frame.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					RXSIZE										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCE	RXDV	RXOK	RXBR	RXMC	Rsvd	RXDR	Reserved								

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:27	N/A	Reserved	N/A	N/A
D26:16	R	RXSIZE	0x000	**Receive frame size in bytes** Length of the received frame, in bytes.

Table 210: Ethernet Receive Status register

Bits	Access	Mnemonic	Reset	Description

D15	R	RXCE	0x0	**Receive carrier event previously seen**
				When set, indicates that a *carrier event* activity (an activity on the receive channel that does not result in a frame receive attempt being made) was found at some point since the last receive statistics. A carrier event results when the interface signals to the PHY have the following values:
				MRXER = 1
				MRXDV = 0
				RXD = 0xE
				The event is being reported with this frame, although it is not associated with the frame.

D14	R	RXDV	0x0	**Receive data violation event previously seen** Set when the last receive event was not long enough to be a valid frame.
D13	R	RXOK	0x0	**Receive frame OK** Set when the frame had a valid CRC and no symbol errors.
D12	R	RXBR	0x0	**Receive broadcast frame** Set when the frame has a valid broadcast address.
D11	R	RXMC	0x0	**Receive multicast frame** Set when the frame has a valid multicast address.
D10	N/A	Reserved	N/A	N/A
D09	R	RXDR	0x0	**Receive frame has dribble bits** Set when an additional 1–7 bits are received after the end of the frame. **Note:** Ignore this bit with RMII applications.
D08:00	N/A	Reserved	N/A	N/A

Table 210: Ethernet Receive Status register

MAC Configuration Register #1

Address: A060 0400

MAC Configuration Register #1 provides bits that control functionality within the Ethernet MAC block.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRST	Not used	Reserved		Not used	RPER FUN	RPE MCST	RPET FUN	Reserved			LOOP BK	Not used			RXEN

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A
D15	R/W	SRST	1	**Soft reset** Set this bit to 1 to reset the RX_WR, TX_RD, MAC (except host interface), SAL (except host interface), and RMII modules.
D14	R/W	Not used	0	Always write as 0.
D13:12	N/A	Reserved	N/A	N/A
D11	N/A	Not used	0	Always write as 0.
D10	R/W	RPERFUN	0	**Reset PERFUN** Set this bit to 1 to put the MAC receive logic into reset.
D09	R/W	RPEMCST	0	**Reset PEMCS/TX** Set this bit to 1 to put the MAC control sublayer/ transmit domain logic into reset.
D08	R/W	RPETFUN	0	**Reset PETFUN** Set this bit to 1 to put the MAC transmit logic into reset.
D07:05	N/A	Reserved	N/A	N/A
D04	R/W	LOOPBK	0	**Internal loopback** Set this bit to 1 to cause the MAC transmit interface to be internally looped back to the MAC receive interface. Clearing this bit results in normal operation.
D03:01	R/W	Not used	0	Always write as 0.

Table 211: MAC Configuration Register #1

Bits	Access	Mnemonic	Reset	Description
D00	R/W	RXEN	0	**Receive enable** Set this bit to 1 to allow the MAC receiver to receive frames.

Table 211: MAC Configuration Register #1

MAC Configuration Register #2

Address: A060 0404

MAC Configuration Register #2 provides additional bits that control functionality within the Ethernet MAC block.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd	EDE FER	Not used	NOBO	Reserved		LONGP	PUREP	AUTOP	VLANP	PADEN	CRCEN	Not used	HUGE	Not used	FULLD

Register bit assignment

Bits	Access	Mnemonic	Reset	Definition
D31:15	N/A	Reserved	N/A	N/A
D14	R/W	EDEFER	0	**Excess deferral** 0 The MAC aborts when the excessive deferral limit is reached (that is, 6071 nibble times in 100 Mbps mode or 24,287 bit times in 10 Mbps mode). 1 Enables the MAC to defer to carrier indefinitely, as per the 802.3u standard.
D13	R/W	Not used	0	Always write to 0.
D12	R/W	NOBO	0	**No backoff** When this bit is set to 1, the MAC immediately retransmits following a collision, rather than using the binary exponential backoff algorithm (as specified in the 802.3u standard).
D11:10	N/A	Reserved	N/A	N/A
D09	R/W	LONGP	0	**Long preamble enforcement** 0 Allows any length preamble (as defined in the 802.3u standard). 1 The MAC allows only receive frames that contain preamble fields less than 12 bytes in length.

Table 212: MAC Configuration Register #2

Bits	Access	Mnemonic	Reset	Definition
D08	R/W	PUREP	0	**Pure preamble enforcement** 0 No preamble checking is performed 1 The MAC certifies the content of the preamble to ensure that it contains 0x55 and is error-free.
D07	R/W	AUTOP	0	**Auto detect pad enable** When set to 1, this bit causes the MAC to detect automatically the type of transmit frame, either tagged or untagged, by comparing the two octets following the source address with the 0x8100 VLAN protect ID and pad accordingly. **Note:** This bit is ignored if PADEN is set to 0. See "PAD operation table for transmit frames" on page 353 for more information.
D06	R/W	VLANP	0	**VLAN pad enable** Set to 1 to have the MAC pad all short transmit frames to 64 bytes and to append a valid CRC. This bit is used in conjunction with auto detect pad enable (AUTOP) and pad/CRC enable (PADEN). See "PAD operation table for transmit frames" on page 353. **Note:** This bit is ignored if PADEN is set to 0.
D05	R/W	PADEN	0	**Pad/CRC enable** 0 Short transmit frames not padded. 1 The MAC pads all short transmit frames. This bit is used in conjunction with auto detect pad enable (AUTOP) and VLAN pad enable (VLANP). See "PAD operation table for transmit frames" on page 353.
D04	R/W	CRCEN	0	**CRC enable** 0 Transmit frames presented to the MAC contain a CRC. 1 Append a CRC to every transmit frame, whether padding is required or not. CRCEN must be set if PADEN is set to 1.
D03	R/W	Not used	0	Always write as 0.

Table 212: MAC Configuration Register #2

Bits	Access	Mnemonic	Reset	Definition
D02	R/W	HUGE	0	**Huge frame enable** 0 Transmit and receive frames are limited to the MAXF value in the Maximum Frame register (see "Maximum Frame register" on page 357). 1 Frames of any length are transmitted and received.
D01	R/W	Not used	0	Always write as 0.
D00	R/W	FULLD	0	**Full-duplex** 0 The MAC operates in half-duplex mode. 1 The MAC operates in full-duplex mode.

Table 212: MAC Configuration Register #2

PAD operation table for transmit frames

Type	AUTOP	VLANP	PADEN	Action
Any	X	X	0	No pad; check CRC
Any	0	0	1	Pad to 60 bytes; append CRC
Any	X	1	1	Pad to 64 bytes; append CRC
Any	1	0	1	If untagged, pad to 60 bytes; append CRC If VLAN tagged, pad to 64 bytes; append CRC

Back-to-Back Inter-Packet-Gap register

Address: A060 0408

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									IPGT						

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:07	N/A	Reserved	N/A	N/A
D06:00	R/W	IPGT	0x00	**Back-to-back inter-packet-gap** Programmable field that indicates the nibble time offset of the minimum period between the end of any transmitted frame to the beginning of the next frame. **Full-duplex mode** ■ Register value should be the appropriate period in nibble times minus 3. ■ Recommended setting is 0x15 (21d), which represents the minimum IPG of 0.96 uS (in 100 Mbps) or 9.6uS (in 10 Mbps). **Half-duplex mode** ■ Register value should be the appropriate period in nibble times minus 6. ■ Recommended setting is 0x12 (18d), which represents the minimum IPG of 0.96 uS (in 100 Mbps) or 9.6 uS (in 10 Mbps).

Table 213: Back-to-Back Inter-Packet-Gap register

Non Back-to-Back Inter-Packet-Gap register

Address: A060 040C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd	IPGR1							Rsvd	IPGR2						

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:15	N/A	Reserved	N/A	N/A
D14:08	R/W	IPGR1	0x00	**Non back-to-back inter-packet-gap part 1** Programmable field indicating optional carrierSense window (referenced in IEEE 8.2.3/4.2.3.2.1). ■ If carrier is detected during the timing of IPGR1, the MAC defers to carrier. ■ If carrier comes after IPGR1, the MAC continues timing IPGR2 and transmits — knowingly causing a collision. This ensures fair access to the medium. IPGR1's range of values is 0x0 to IPGR2. The recommended value is 0xC.
D07	N/A	Reserved	N/A	N/A
D06:00	R/W	IPGR2	0x00	**Non back-to-back inter-packet-gap part 2** Programmable field indicating the non back-to-back inter-packet-gap. The recommended value for this field is 0x12 (18d), which represents the minimum IPG of 0.96 µS in 100 Mbps or 9.6 µS in 10 Mbps.

Table 214: Non Back-to-Back Inter-Packet-Gap register

Collision Window/Retry register

Address: A060 0410

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CWIN						Reserved				RETX			

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:14	N/A	Reserved	N/A	N/A
D13:08	R/W	CWIN	0x37	**Collision window** Programmable field indicating the slot time or collision window during which collisions occur in properly configured networks. Because the collision window starts at the beginning of transmissions, the preamble and SFD (start-of-frame delimiter) are included. The default value (0x37 (55d)) corresponds to the frame byte count at the end of the window.
D07:04	N/A	Reserved	N/A	N/A
D03:00	R/W	RETX	0xF	**Retransmission maximum** Programmable field specifying the number of retransmission attempts following a collision before aborting the frame due to excessive collisions. The 802.3u standard specifies the attemptLimit to be 0xF (15d).

Table 215: Collision Window/Retry register

Maximum Frame register

Address: A060 0414

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXF															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A
D15:00	R/W	MAXF	0x0600	**Maximum frame length** Default value of 0x600 represents a maximum receive frame of 1536 octets. An untagged maximum-size Ethernet frame is 1518 octets. A tagged frame adds four octets for a total of 1522 octets. To use a shorter maximum length restriction, program this field accordingly. **Note:** If a proprietary header is allowed, this field should be adjusted accordingly. For example, if 4-byte proprietary headers are prepended to the frames, the MAXF value should be set to 1526 octets. This allows the maximum VLAN tagged frame plus the 4-byte header.

Table 216: Maximum Frame register

PHY Support register

Address: A060 0418

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPER MII	Not used						SPEED	Not used							

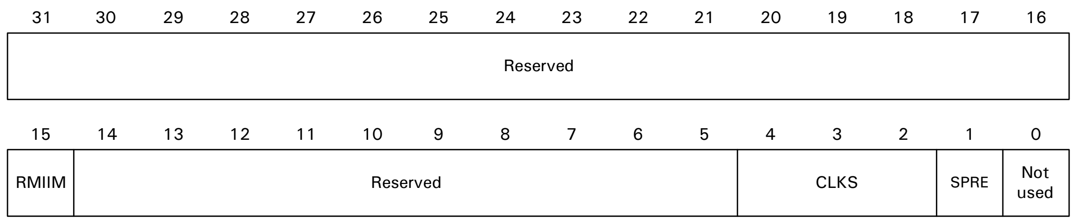
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A
D15	R/W	RPERMII	0	**Reset RMII module** Set to 1 to reset the RMII PHY interface module logic.
D14:09	R/W	Not used	0x08	Always write 0x08.
D08	R/W	SPEED	0	**Speed select (RMII**) 0 RMII PHY interface logic is configured for 10 Mbps 1 RMII PHY interface logic is configured for 100 Mbps
D07:00	R?W	Not used	0x00	Always write 0x00.

Table 217: PHY Support register

MII Management Configuration register

Address: A060 0420



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A
D15	R/W	RMIIM	0	**Reset MII management block** Set this bit to 1 to reset the MII Management module.
D14:05	N/A	Reserved	N/A	N/A
D04:02	R/W	CLKS	0x0	**Clock select** Used by the clock divide logic in creating the MII management clock, which (per the IEEE 802.3u standard) can be no faster than 2.5 MHz. **Note:** Some PHYs support clock rates up to 12.5 MHz. The AHB bus clock is used as the input to the clock divide logic. "Clocks field settings," on page 360, shows the settings that are supported.
D01	R/W	SPRE	0	**Suppress preamble** 0 Causes normal cycles to be performed 1 Causes the MII Management module to perform read/write cycles without the 32-bit preamble field. (Preamble suppression is supported by some PHYs.)
D00	R/W	Not used	0	Always write to 0.

Table 218: MII Management Configuration register

Clocks field settings

CLKS field	Divisor	AHB bus clock for 2.5 MHz	AHB bus clock for 12.5 MHz
000	4		50 MHz
001	4		50 MHz
010	6		75 MHz
011	8		100 MHz
100	10		
101	20	50 MHz	
110	30	75 MHz	
111	40	100 MHz	

MII Management Command register

Address: A060 0424

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SCAN	READ

Register bit assignment

Note: If both SCAN and READ are set, SCAN takes precedence.

Bits	Access	Mnemonic	Reset	Description
D31:02	N/A	Reserved	N/A	N/A

Table 219: MII Management Command register

Bits	Access	Mnemonic	Reset	Description
D01	R/W	SCAN	0	**Automatically scan for read data** Set to 1 to have the MII Management module perform read cycles continuously. This is useful for monitoring link fail, for example. **Note:** SCAN must transition from a 0 to a 1 to initiate the continuous read cycles.
D00	R/W	READ	0	**Single scan for read data** Set to 1 to have the MII Management module perform a single read cycle. The read data is returned in the MII Management Read Data register after the BUSY bit in the MII Management Indicators register has returned to a value of 0. **Note:** READ must transition from a 0 to a 1 to initiate a single read cycle.

Table 219: MII Management Command register

MII Management Address register

Address: A060 0428

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DADR					Reserved			RADR				

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:13	N/A	Reserved	N/A	N/A

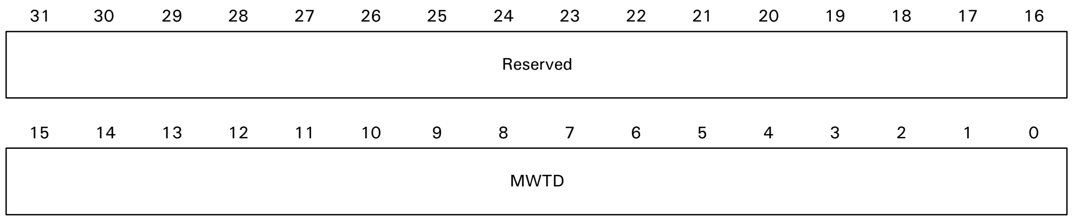
Table 220: MII Management Address register

Bits	Access	Mnemonic	Reset	Description
D12:08	R/W	DADR	0x00	**MII PHY device address** Represents the 5-bit PHY device address field for management cycles. Up to 32 different PHY devices can be addressed.
D07:05	N/A	Reserved	N/A	N/A
D04:00	R/W	RADR	0x00	**MII PHY register address** Represents the 5-bit PHY register address field for management cycles. Up to 32 registers within a single PHY device can be addressed.

Table 220: MII Management Address register

MII Management Write Data register

Address: A060 042C



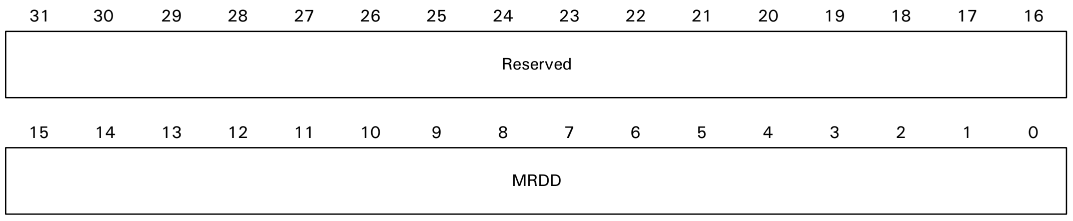
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A
D15:00	R/W	MWTD	0x0000	**MII write data** When this register is written, an MII Management write cycle is performed using this 16-bit data along with the preconfigured PHY device and PHY register addresses defined in the MII Management Address register (see page 361). The write operation completes when the BUSY bit in the MII Management Indicators register (see page 363) returns to 0.

Table 221: MII Management Write Data register

MII Management Read Data register

Address: A060 0430



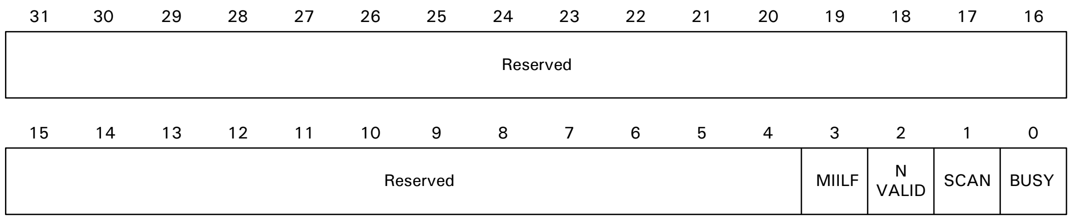
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A
D15:00	R	MRDD	0x0000	**MII read data** Read data is obtained by reading from this register after an MII Management read cycle. An MII Management read cycle is executed by loading the MII Management Address register, then setting the READ bit to 1 in the MII Management Command register (see page 360). Read data is available after the BUSY bit in the MII Management Indicators register (see page 363) returns to 0.

Table 222: MII Management Read Data register

MII Management Indicators register

Address: A060 0434



Register bit assignment

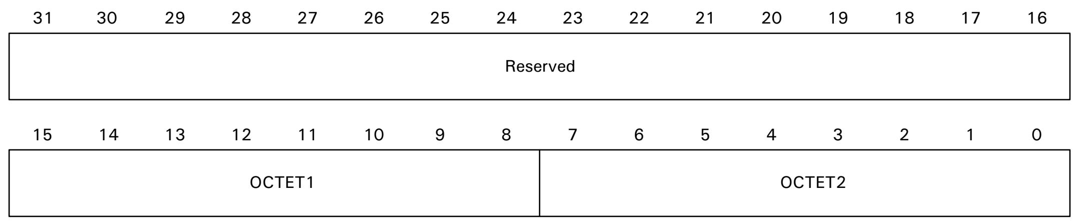
Bits	Access	Mnemonic	Reset	Description
D31:04	N/A	Reserved	N/A	N/A
D03	R	MIILF	0	**MII link failure** When set to 1, indicates that the PHY currently has a link fail condition.
D02	R	NVALID	0	**Read data not valid** When set to 1, indicates that the MII Management read cycle has not completed and the read data is not yet valid. Also indicates that SCAN READ is not valid for automatic scan reads.
D01	R	SCAN	0	**Automatically scan for read data in progress** When set to 1, indicates that continuous MII Management scanning read operations are in progress.
D00	R	BUSY	0	**MII interface BUSY with read/write operation** When set to 1, indicates that the MII Management module currently is performing an MII Management read or write cycle. This bit returns to 0 when the operation is complete.

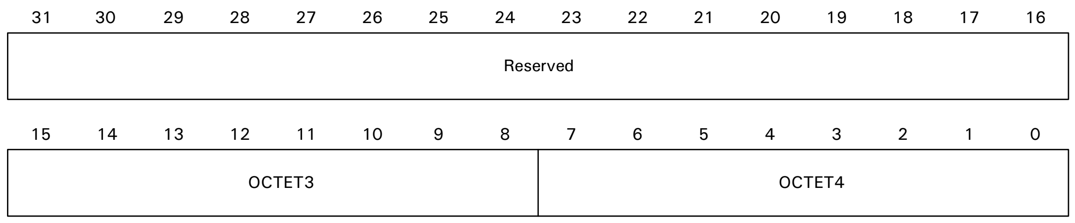
Table 223: MII Management Indicators register

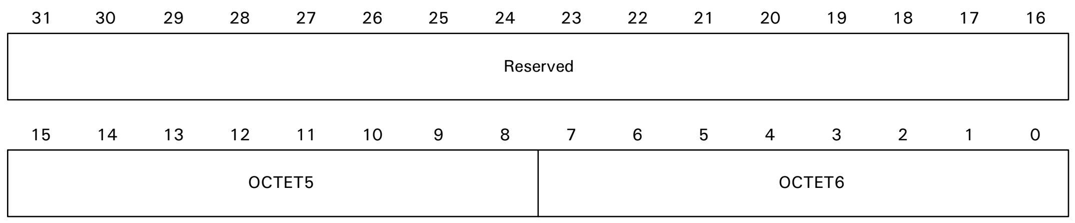
Station Address registers

Address: A060 0440 / 0444 / 0448

The 48-bit station address is loaded into Station Address Register #1, Station Address Register #2, and Station Address Register #3, for use by the station address logic (see "Station address logic (SAL)" on page 321).







Bits	Access	Mnemonic	Reset	Description
Station Address Register #1				
D31:16	N/A	Reserved	N/A	N/A
D15:08	R/W	OCTET1	0	Station address octet #1 (stad[7:0])
D07:00	R/W	OCTET2	0	Station address octet #2 (stad[15:8])
Station Address Register #2				
D31:16	N/A	Reserved	N/A	N/A
D15:08	R/W	OCTET3	0	Station address octet #3 (stad[23:16])
D07:00	R/W	OCTET4	0	Station address octet #4 (stad[31:24])
Station Address Register #3				
D31:16	N/A	Reserved	N/A	N/A
D15:08	R/W	OCTET5	0	Station address octet #5 (stad[39:32])
D07:00	R/W	OCTET6	0	Station address octet #6 (stad[47:40])

Table 224: Station Address registers

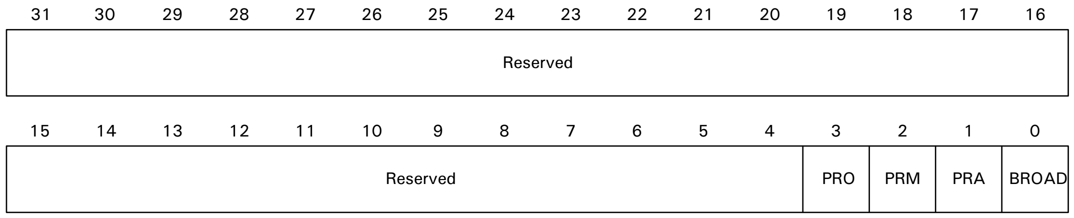
Note: Octet #6 is the first byte of a frame received from the MAC. Octet #1 is the last byte of the station address received from the MAC.

Station Address Filter register

Address: A060 0500

The Station Address Filter register contains several filter controls. The register is located in the station address logic (see "Station address logic (SAL)" on page 321).

All filtering conditions are independent of each other. For example, the station address logic can be programmed to accept all multicast frames, all broadcast frames, and frames to the programmed destination address.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	N/A	Reserved	N/A	N/A
D03	R/W	PRO	0	Enable promiscuous mode; receive all frames
D02	R/W	PRM	0	Accept all multicast frames
D01	R/W	PRA	0	Accept multicast frames using the hash table
D00	R/W	BROAD	0	Accept all broadcast frames

Table 225: Station Address Filter register

Register Hash Tables

The MAC receiver provides the station address logic with a 6-bit CRC value that is the upper six bits of a 32-bit CRC calculation performed on the 48-bit multicast destination address. This 6-bit value addresses the 64-bit multicast hash table created in HT1 (hash table 1) and HT2 (hash table 2). If the current receive frame is a multicast frame and the 6-bit CRC addresses a bit in the hash table that is set to 1, the receive frame will be accepted; otherwise, the receive frame is rejected.

HT1 stores enables for the lower 32 CRC addresses; HT2 stores enables for the upper 32 CRC addresses.

HT1

Address: A060 0504

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT1															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	HT1	0x00000000	CRC 31:00

Table 226: Hash Table Register 1

HT2

Address: A060 0508

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT2															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT2															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	HT2	0x00000000	CRC 63:32

Table 227: Hash Table Register 2

Statistics registers

Address: A060 0680 (base register)

The Statistics module has 39 counters and 4 support registers that count and save Ethernet statistics. The Ethernet General Control Register #2 contains three Statistics module configuration bits: AUTOZ, CLRCNT, and STEN. The counters support a “clear on read” capability that is enabled when AUTOZ is set to 1.

Combined transmit and receive statistics counters

The combined transmit and receive statistics counters, listed in Table 228, are incremented for each good or bad frame, transmitted and received, that falls within the specified frame length limits of the counter (for example, TR127 counts 65–127 byte frames). The frame length excludes framing bits and includes the FCS (checksum) bytes. All counters are 18 bits, with this bit configuration:

D31:18	R		Reserved
D17:00	R/W	Reset = 0x00000	Count (R/W)

Address	Register	Transmit and receive counters			R/W
A060_0680	TR64	Transmit & receive 64		Byte frame counter	R/W
A060_0684	TR127	Transmit & receive 65	to	127 Byte frame counter	R/W
A060_0688	TR255	Transmit & receive 128	to	255 Byte frame counter	R/W
A060_068C	TR511	Transmit & receive 256	to	511 Byte frame counter	R/W
A060_0690	TR1K	Transmit & receive 512	to	1023 Byte frame counter	R/W
A060_0694	TRMAX	Transmit & receive 1024	to	1518 Byte frame counter	R/W
A060_0698	TRMGV	Transmit & receive 1519 count	to 1522 Byte good VLAN frame		R/W

Table 228: Combined transmit and receive statistics counters address map

Receive statistics counters

Address	Register	Receive counters	R/W
A060_069C	RBYT	Receive byte counter	R/W
A060_06A0	RPKT	Receive packet counter	R/W
A060_06A4	RFCS	Receive FCS error counter	R/W
A060_06A8	RMCA	Receive multicast packet counter	R/W
A060_06AC	RBCA	Receive broadcast packet counter	R/W
A060_06B0	RXCF	Receive control frame packet counter	R/W
A060_06B4	RXPF	Receive PAUSE frame packet counter	R/W
A060_06B8	RXUO	Receive unknown OPCODE counter	R/W
A060_06BC	RALN	Receive alignment error counter	R/W
A060_06C0	Reserved	N/A	N/A
A060_06C4	RCDE	Receive code error counter	R/W
A060_06C8	RCSE	Receive carrier sense error counter	R/W
A060_06CC	RUND	Receive undersize packet counter	R/W
A060_06D0	ROVR	Receive oversize packet counter	R/W
A060_06D4	RFRG	Receive fragments counter	R/W
A060_06D8	RJBR	Receive jabber counter	R/W
A060-06DC	Reserved	N/A	N/A

Table 229: Receive statistics counters address map

Receive byte counter (A060 069C)

Incremented by the byte count of frames received with 0 to 1518 bytes, including those in bad packets, excluding framing bits but including FCS bytes.

D31:24	R	Reset = Read as 0	Reserved
D23:00	R/W	Reset = 0x000000	RBYT

Receive packet counter (A060 06A0)

Incremented for each received frame (including bad packets, and all unicast, broadcast, and multicast packets).

D31:18	R	Reset = Read as 0	Reserved
D17:00	R/W	Reset = 0x00000	RPKT

Receive FCS error counter (A060 06A4)

Incremented for each frame received with a length of 64 to 1518 bytes, and containing a frame check sequence (FCS) error. FCS errors are not counted for VLAN frames that exceed 1518 bytes or for any frames with dribble bits.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	RFCS

Ignore this counter for RMII applications as it does not operate properly.

Receive multicast packet counter (A060 06A8)

Incremented for each good multicast frame with a length no greater than 1518 bytes (non-VLAN) or 1522 bytes (VLAN), excluding broadcast frames. This counter does not look at range/length errors.

D31:18	R	Reset = Read as 0	Reserved
D17:00	R/W	Reset = 0x00000	RMCA

Receive broadcast packet counter (A060 06AC)

Incremented for each good broadcast frame with a length no greater than 1518 bytes (non-VLAN) or 1522 bytes (VLAN), excluding multicast frames. This counter does not look at range/length errors.

D31:18	R	Reset = Read as 0	Reserved
D17:00	R/W	Reset = 0x00000	RBCA

Receive control frame packet counter (A060 06B0)

Incremented for each MAC control frame received (PAUSE and unsupported).

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	RXCF

Receive PAUSE frame packet counter (A060 06B4)

Incremented each time a valid PAUSE control frame is received.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	RXPF

Receive unknown OPCODE packet counter (A060 06B8)

Incremented each time a MAC control frame is received with an OPCODE other than PAUSE.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	RBUO

Receive alignment error counter (A060 06BC)

Incremented for each received frame, from 64 to 1518 bytes, that contains an invalid FCS and has dribble bits (that is, is not an integral number of bytes).

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	RALN

Ignore this counter for RMII applications as it does not operate properly.

Receive code error counter (A060 06C4)

Incremented each time a valid carrier was present and at least one invalid data symbol was found.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	RCDE

Receive carrier sense error counter (A060 06C8)

Incremented each time a false carrier is found during idle, as defined by a 1 on RX_ER and an 0xE on RXD. The event is reported with the statistics generated on the next received frame. Only one false carrier condition can be detected and logged between frames.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	RCSE

Receive undersize packet counter (A060 06CC)

Incremented each time a frame is received that is less than 64 bytes in length, contains a valid FCS, and is otherwise well-formed. This counter does not look at range/length errors.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	RUND

Receive oversize packet counter (A060 06D0)

Incremented each time a frame is received that exceeds 1518 bytes (non-VLAN) or 1522 bytes (VLAN), contains a valid FCS, and is otherwise well-formed. This counter does not look at range/length errors. This counter is not incremented when a packet is truncated because it exceeds the MAXF value.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	ROVR

Receive fragments counter (A060 06D4)

Incremented for each frame received that is less than 64 bytes in length and contains an invalid FCS; this includes integral and non-integral lengths.

D31:12	R		Reserved
D11:00	R/W	Reset = 0x000	RFRG

Receive jabber counter (A060 06D8)

Incremented for frames received that exceed 1518 bytes (non-VLAN) or 1522 bytes (VLAN) and contain an invalid FCS, including alignment errors. This counter does not increment when a packet is truncated to 1518 (non-VLAN) or 1522 (VLAN) bytes by MAXF.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	RJBR

Transmit statistics counters

Address	Register	Transmit counters	R/W
A060_06E0	TBYT	Transmit byte counter	R/W
A060_06E4	TPKT	Transmit packet counter	R/W
A060_06E8	TMCA	Transmit multicast packet counter	R/W
A060_06EC	TBCA	Transmit broadcast packet counter	R/W
A060_06F0	Reserved	N/A	N/A
A060_06F4	TDFR	Transmit deferral packet counter	R/W
A060_06F8	TEDF	Transmit excessive deferral packet counter	R/W
A060_06FC	TSCL	Transmit single collision packet counter	R/W
A060_0700	TMCL	Transmit multiple collision packet counter	R/W
A060_0704	TLCL	Transmit late collision packet counter	R/W
A060_0708	TXCL	Transmit excessive collision packet counter	R/W
A060_070C	TNCL	Transmit total collision counter	R/W
A060_0710	Reserved	N/A	N/A
A060_0714	Reserved	N/A	N/A
A060_0718	TJBR	Transmit jabber frame counter	R/W
A060_071C	TFCS	Transmit FCS error counter	R/W
A060_0720	Reserved	N/A	N/A
A060_0724	TOVR	Transmit oversize frame counter	R/W
A060_0728	TUND	Transmit undersize frame counter	R/W
A060_072C	TFRG	Transmit fragments frame counter	R/W

Table 230: Transmit statistics counters address map

Transmit byte counter (A060 06E0)

Incremented by the number of bytes that were put on the wire, including fragments of frames that were involved with collisions. This count does not include preamble/ SFD or jam bytes.

D31:24	R	Reset = Read as 0	Reserved
D23:00	R/W	Reset = 0x000000	TBYT

Transmit packet counter (A060 06E4)

Incremented for each transmitted packet (including bad packets, excessive deferred packets, excessive collision packets, late collision packets, and all unicast, broadcast, and multicast packets).

D31:18	R	Reset = Read as 0	Reserved
D17:00	R/W	Reset = 0x00000	TPKT

Transmit multicast packet counter (A060 06E8)

Incremented for each multicast valid frame transmitted (excluding broadcast frames).

D31:18	R	Reset = Read as 0	Reserved
D17:00	R/W	Reset = 0x00000	TMCA

Transmit broadcast packet counter (A060 06EC)

Incremented for each broadcast frame transmitted (excluding multicast frames).

D31:18	R	Reset = Read as 0	Reserved
D17:00	R/W	Reset = 0x00000	TBCA

Transmit deferral packet counter (A060 06F4)

Incremented for each frame that was deferred on its first transmission attempt. This counter does not include frames involved in collisions.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	TDFR

Transmit excessive deferral packet counter (A060 06F8)

Incremented for frames aborted because they were deferred for an excessive period of time (3036 byte times).

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	TEDF

Transmit single collision packet counter (A060 06FC)

Incremented for each frame transmitted that experienced exactly one collision during transmission.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	TSCL

Transmit multiple collision packet counter (A060 0700)

Incremented for each frame transmitted that experienced 2–15 collisions (including any late collisions) during transmission.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	TMCL

Transmit late collision packet counter (A060 0704)

Incremented for each frame transmitted that experienced a late collision during a transmission attempt. Late collisions are defined using the CWIN[13:08] field of the Collision Window/Retry register.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	TLCL

Transmit excessive collision packet counter (A060 0708)

Incremented for each frame transmitted that experienced excessive collisions during transmission, as defined by the RETX [03:00] field of the Collision Window/Retry register, and was aborted.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	TXCL

Transmit total collision packet counter (A060 070C)

Incremented by the number of collisions experienced during the transmission of a frame.

Note: This register does not include collisions that result in an excessive collision count or late collisions.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	TNCL

Transmit jabber frame counter (A060 0718)

Incremented for each oversized transmitted frame with an incorrect FCS value.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	TJBR

Transmit FCS error counter (A060 071C)

Incremented for every valid-sized packet with an incorrect FCS value.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	TFCS

Transmit oversize frame counter (A060 0724)

Incremented for each transmitted frame that exceeds 1518 bytes (NON_VLAN) or 1522 bytes (VLAN) and contains a valid FCS.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	TOVR

Transmit undersize frame counter (A060 0728)

Incremented for every frame less than 64 bytes, with a correct FCS value. This counter also is incremented when a jumbo packet is aborted (see "TXAJ" on page 346) and the MAC is not checking the FCS (see "CRCEN" on page 352), because the frame is reported as having a length of 0 bytes.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	TUND

Transmit fragment counter (A060 072C)

Incremented for every frame less than 64 bytes, with an incorrect FCS value.

D31:12	R	Reset = Read as 0	Reserved
D11:00	R/W	Reset = 0x000	TFRG

General Statistics registers

Table 231 lists the General Statistics registers.

Address	Register	General registers	R/W
A060_0730	CAR1	Carry Register 1	R
A060_0734	CAR2	Carry Register 2	R
A060_0738	CAM1	Carry Register 1 Mask register	R/W
A060_073C	CAM2	Carry Register 2 Mask register	R/W

Table 231: General Statistics register address map

Carry Register 1 (CAR1) and Carry Register 2 (CAR2) have carry bits for all of the statistics counters. These carry bits are set when the associated counter reaches a rollover condition.

These carry bits also can cause the STOVFL (statistics counter overflow) bit in the Ethernet Interrupt Status register (see "Ethernet Interrupt Status register" on page 385) to be set. Carry Register 1 Mask register (CAM1) and Carry Register 2 Mask register (CAM2) have individual mask bits for each of the carry bits. When set, the mask bit prevents the associated carry bit from setting the STOVFL bit.

Carry Register 1

Address: A060 0730

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C164	C1127	C1255	C1511	C11K	C1 MAX	C1 MGV	Reserved								C1 RBY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1 RPK	C1 RFC	C1 RMC	C1 RBC	C1 RXC	C1RXP	C1 RXU	C1RAL	Rsvd	C1 RCD	C1 RCS	C1 RUN	C1 ROV	C1 RFR	C1 RJB	Rsvd

Bits	Access	Mnemonic	Reset	Description
D31	R/C	C164	0	Carry register 1 TR64 counter carry bit
D30	R/C	C1127	0	Carry register 1 TR127 counter carry bit
D29	R/C	C1255	0	Carry register 1 TR255 counter carry bit
D28	R/C	C1511	0	Carry register TR511 counter carry bit
D27	R/C	C11K	0	Carry register 1 TR1K counter carry bit
D26	R/C	C1MAX	0	Carry register 1 TRMAX counter carry bit
D25	R/C	C1MGV	0	Carry register 1 TRMGV counter carry bit
D24:17	N/A	Reserved	N/A	N/A
D16	R/C	C1RBY	0	Carry register 1 RBYT counter carry bit
D15	R/C	C1RPK	0	Carry register 1 RPKT counter carry bit
D14	R/C	C1RFC	0	Carry register 1 RFCS counter carry bit
D13	R/C	C1RMC	0	Carry register 1 RMCA counter carry bit
D12	R/C	C1RBC	0	Carry register 1 RBCA counter carry bit
D11	R/C	C1RXC	0	Carry register 1 RXCF counter carry bit
D10	R/C	C1RXP	0	Carry register 1 RXPF counter carry bit
D09	R/C	C1RXU	0	Carry register 1 RXUO counter carry bit
D08	R/C	C1RAL	0	Carry register 1 RALN counter carry bit
D07	N/A	Reserved	N/A	N/A
D06	R/C	C1RCD	0	Carry register 1 RCDE counter carry bit

Table 232: Carry Register 1

Bits	Access	Mnemonic	Reset	Description
D05	R/C	C1RCS	0	Carry register 1 RCSE counter carry bit
D04	R/C	C1RUN	0	Carry register 1 RUND counter carry register
D03	R/C	C1ROV	0	Carry register 1 ROVR counter carry bit
D02	R/C	C1RFR	0	Carry register 1 RFRG counter carry bit
D01	R/C	C1RJB	0	Carry register 1 RJBR counter carry bit
D00	N/A	Reserved	N/A	N/A

Table 232: Carry Register 1

Carry Register 2

Address: A060 0734

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												C2 JTB	C2 TFC	Rsvd	C2 TOV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C2 TUN	C2 TFG	C2 TBY	C2 TPK	C2TMC	C2TBC	Rsvd	C2TDF	C2 TED	C2 TSC	C2 TMA	C2 TLC	C2 TXC	C2 TNC	Reserved	

Bits	Access	Mnemonic	Reset	Description
D31:20	N/A	Reserved	N/A	N/A
D19	R/C	C2TJB	0	Carry register 2 TJBR counter carry bit
D18	R/C	C2TFC	0	Carry register 2 TFCS counter carry bit
D17	N/A	Reserved	N/A	N/A
D16	R/C	C2TOV	0	Carry register 2 TOVR counter carry bit
D15	R/C	C2TUN	0	Carry register 2 TUND counter carry bit
D14	R/C	C2TFG	0	Carry register 2 TFRG counter carry bit
D13	R/C	C2TBY	0	Carry register 2 TBYT counter carry bit
D12	R/C	C2TPK	0	Carry register 2 TPKT counter carry bit
D11	R/C	C2TMC	0	Carry register 2 TMCA counter carry bit

Table 233: Carry Register 2

Bits	Access	Mnemonic	Reset	Description
D10	R/C	C2TBC	0	Carry register 2 TBCA counter carry bit
D09	N/A	Reserved	N/A	N/A
D08	R/C	C2TDF	0	Carry register 2TDFR counter carry bit
D07	R/C	C2TED	0	Carry register 2 TEDF counter carry bit
D06	R/C	C2TSC	0	Carry register 2 TSCL counter carry bit
D05	R/C	C2TMA	0	Carry register 2 TMCL counter carry bit
D04	R/C	C2TLC	0	Carry register 2 TLCL counter carry bit
D03	R/C	C2TXC	0	Carry register 2 TXCL counter carry bit
D02	R/C	C2TNC	0	Carry register 2 TNCL counter carry bit
D01:00	N/A	Reserved	N/A	N/A

Table 233: Carry Register 2

Carry Register 1 Mask register

Address: A060 0738

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M164	M1127	M 1255	M 1511	M C11K	M1 MAX	M1 MGV	Reserved								M1 RBY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1 RPK	M1 RFC	M1 RMC	M1 RBC	M1 RXC	M1RXP	M1 RXU	M1RAL	Not used	M1 RCD	M1 RCS	M1 RUN	M1 ROV	M1 RFR	M1 RJB	Not used

Bits	Access	Mnemonic	Reset	Description
D31	R/W	M164	1	Mask register 1 TR64 counter carry bit mask
D30	R/W	M1127	1	Mask register 1 TR127 counter carry bit mask
D29	R/W	M1255	1	Mask register 1 TR255 counter carry bit mask
D28	R/W	M1511	1	Mask register 1 TR511 counter carry bit mask
D27	R/W	M11K	1	Mask register 1 TR1K counter carry bit mask
D26	R/W	M1MAX	1	Mask register 1 TRMAX counter carry bit mask

Table 234: Carry Register 1 Mask register

Bits	Access	Mnemonic	Reset	Description
D25	R/W	M1MGV	1	Mask register 1 TRMGV counter carry bit mask
D24:17	N/A	Reserved	N/A	N/A
D16	R/W	M1RBY	1	Mask register 1 RBYT counter carry bit mask
D15	R/W	M1RPK	1	Mask register 1 RPKT counter carry bit mask
D14	R/W	M1RFC	1	Mask register 1 RFCS counter carry bit mask. Set this bit to 1 for RMII applications.
D13	R/W	M1RMC	1	Mask register 1 RMCA counter carry bit mask
D12	R/W	M1RBC	1	Mask register 1 RBCA counter carry bit mask
D11	R/W	M1RXC	1	Mask register 1 RXCF counter carry bit mask
D10	R/W	M1RXP	1	Mask register 1 RXPF counter carry bit mask
D09	R/W	M1RXU	1	Mask register 1 RXUO counter carry bit mask
D08	R/W	M1RAL	1	Mask register 1 RALN counter carry bit mask. Set this bit to 1 for RMII applications.
D07	N/A	Not used	1	Always write as 1.
D06	R/W	M1RCD	1	Mask register 1 RCDE counter carry bit mask
D05	R/W	M1RCS	1	Mask register 1 RCSE counter carry bit mask
D04	R/W	M1RUN	1	Mask register 1 RUND counter carry bit mask
D03	R/W	M1ROV	1	Mask register 1 ROVR counter carry bit mask
D02	R/W	M1RFR	1	Mask register 1 RFRG counter carry bit mask
D01	R/W	M1RJB	1	Mask register 1 RJBR counter carry bit mask
D00	R/W	Not used	1	Always write as 1.

Table 234: Carry Register 1 Mask register

Carry Register 2 Mask register

Address: A060 073C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												M2 JTB	M2 TFC	Not used	M2 TOV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M2 TUN	M2 TFG	M2 TBY	M2 TPK	M2 TMC	M2TBC	Not used	M2TDF	M2 TED	M2 TSC	M2 TMA	M2 TLC	M2 TXC	M2 TNC	Not used	

Bits	Access	Mnemonic	Reset	Description
D31:20	N/A	Reserved	N/A	N/A
D19	R/W	M2TJB	1	Mask register 2 TJBR counter carry bit mask
D18	R/W	M2TFC	1	Mask register 2 TFCS counter carry bit mask
D17	R/W	Not used	1	Always write as 1.
D16	R/W	M2TOV	1	Mask register 2 TOVR counter carry bit mask
D15	R/W	M2TUN	1	Mask register 2 TUND counter carry bit mask
D14	R/W	M2TFG	1	Mask register 2 TFRG counter carry bit mask
D13	R/W	M2TBY	1	Mask register 2 TBYT counter carry bit mask
D12	R/W	M2TPK	1	Mask register 2 TPKT counter carry bit mask
D11	R/W	M2TMC	1	Mask register 2 TMCA counter carry bit mask
D10	R/W	M2TBC	1	Mask register 2 TBCA counter carry bit mask
D09	R/W	Not used	1	Always write as 1.
D08	R/W	M2TDF	1	Mask register 2 TDFR counter carry bit mask
D07	R/W	M2TED	1	Mask register 2 TEDF counter carry bit mask
D06	R/W	M2TSC	1	Mask register 2 TSCL counter carry bit mask
D05	R/W	M2TMA	1	Mask register 2 TMCL counter carry bit mask
D04	R/W	M2TLC	1	Mask register 2 TLCL counter carry bit mask
D03	R/W	M2TXC	1	Mask register 2 TXCL counter carry bit mask
D02	R/W	M2TNC	1	Mask register 2 TNCL counter carry bit mask
D01:00	R/W	Not used	11	Always write as "11."

Table 235: Carry Register 2 Mask register

RX_A Buffer Descriptor Pointer register

Address: A060 0A00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXAPTR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXAPTR															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	RXAPTR	0x00000000	**RX_A Buffer Descriptor Pointer** Contains a pointer to the initial receive buffer descriptor for the A pool of buffers.

Table 236: RX_A Buffer Descriptor Pointer register

RX_B Buffer Descriptor Pointer register

Address: A060 0A04

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXBPTR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXBPTR															

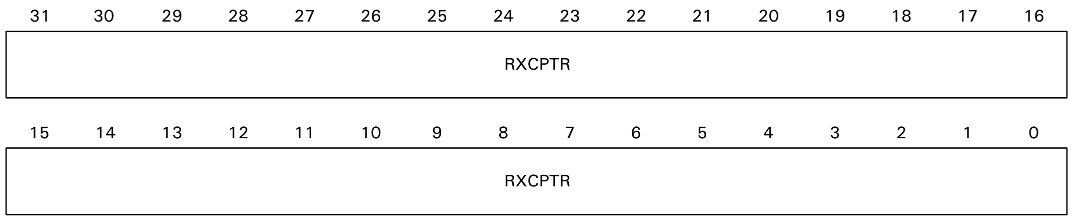
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	RXBPTR	0x00000000	**RX_B Buffer Descriptor Pointer** Contains a pointer to the initial receive buffer descriptor for the B pool of buffers.

Table 237: RX_B Buffer Descriptor Pointer register

RX_C Buffer Descriptor Pointer register

Address: A060 0A08



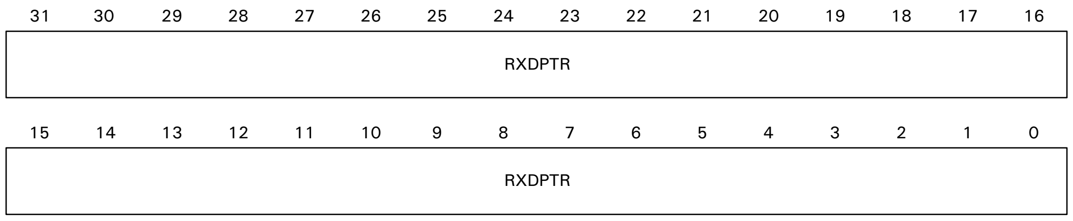
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	RXCPTR	0x00000000	**RX_C Buffer Descriptor Pointer** Contains a pointer to the initial receive buffer descriptor for the C pool of buffers.

Table 238: RX_C Buffer Descriptor Pointer

RX_D Buffer Descriptor Pointer register

Address: A060 0A0C



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	RXDPTR	0x00000000	**RX_D Buffer Descriptor Pointer** Contains a pointer to the initial receive buffer descriptor for the D pool of buffers.

Table 239: RX_D Buffer Descriptor Pointer register

Ethernet Interrupt Status register

Address: A060 0A10

The Ethernet Interrupt Status register contains status bits for all of the Ethernet interrupt sources. Each interrupt status bit is assigned to either the RX or TX Ethernet interrupt; bits D25:16 are assigned to the RX interrupt and D06:00 are assigned to the TX interrupt.

The bits are set to indicate an interrupt condition, and are cleared by writing a 1 to the appropriate bit. All interrupts bits are enabled using the Ethernet Interrupt Enable register (EINTREN). If any enabled bit in the Ethernet Interrupt Status register is set, its associated Ethernet interrupt to the system is set. The interrupt to the system is negated when all active interrupt sources have been cleared. If an interrupt source is active at the same time the interrupt bit is being cleared, the interrupt status bit remains set and the interrupt signal remains set.

Note: For diagnostics, software can cause any of these interrupt status bits to be set by writing a 1 to a bit that is 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						RX OVFL_ DATA	RX OVFL_ STAT	RX BUFC	RX DONE A	RX DONE B	RX DONE C	RX DONE D	RXNO BUF	RX BU FFUL	RXBR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									ST OVFL	Not used	TX BUFC	TX BUF NR	TX DONE	TX ERR	TX IDLE

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:26	N/A	Reserved	N/A	N/A
D25	R/C	RXOVFL_DATA	0	Assigned to RX interrupt. RX data FIFO overflowed. For proper operation, reset the receive packet processor using the ERX bit in the Ethernet General Control register when an overflow condition occurs.

Table 240: Ethernet Interrupt Status register

Bits	Access	Mnemonic	Reset	Description
D24	R/C	RXOVFL_STAT	0	Assigned to RX interrupt. RX status FIFO overflowed.
D23	R/C	RXBUFC	0	Assigned to RX interrupt. I bit set in receive Buffer Descriptor and buffer closed.
D22	R/C	RXDONEA	0	Assigned to RX interrupt. Complete receive frame stored in pool A of system memory.
D21	R/C	RXDONEB	0	Assigned to RX interrupt. Complete receive frame stored in pool B of system memory.
D20	R/C	RXDONEC	0	Assigned to RX interrupt. Complete receive frame stored in pool C of system memory.
D19	R/C	RXDONED	0	Assigned to RX interrupt. Complete receive frame stored in pool D of system memory.
D18	R/C	RXNOBUF	0	Assigned to RX interrupt. No buffer is available for this frame due to one of these conditions: ■ All four buffer rings being disabled ■ All four buffer rings being full ■ No available buffer big enough for the frame
D17	R/C	RXBUFFUL	0	Assigned to RX interrupt. No buffer is available for this frame because all four buffer rings are disabled or full.
D16	R/C	RXBR	0	Assigned to RX interrupt. New frame available in the RX_FIFO. This bit is used for diagnostics.
D15:07	N/A	Reserved	N/A	N/A

Table 240: Ethernet Interrupt Status register

Bits	Access	Mnemonic	Reset	Description
D06	R/C	STOVFL	0	Assigned to TX interrupt. Statistics counter overflow. Individual counters can be masked using the Carry Register 1 and 2 Mask registers. The source of this interrupt is cleared by clearing the counter that overflowed, and by clearing the associated carry bit in either Carry Register 1 or Carry Register 2 by writing a 1 to the bit.
D05	R	Not used	0	Always write as 0.
D04	R/C	TXBUFC	0	Assigned to TX interrupt. I bit set in the Transmit Buffer Descriptor and buffer closed.
D03	R/C	TXBUFNR	0	Assigned to TX interrupt. F bit not set in the Transmit Buffer Descriptor when read from the TX Buffer descriptor RAM.
D02	R/C	TXDONE	0	Assigned to TX interrupt. Frame transmission complete.
D01	R/C	TXERR	0	Assigned to TX interrupt. Last frame not transmitted successfully. See "Ethernet Interrupt Status register" on page 385 for information about restarting the transmitter when this bit is set.
D00	R/C	TXIDLE	0	Assigned to TX interrupt. TX_WR logic has no frame to transmit. See "Ethernet Interrupt Status register" on page 385 for information about restarting the transmitter when this bit is set.

Table 240: Ethernet Interrupt Status register

Ethernet Interrupt Enable register

Address: A060 0A14

The Ethernet Interrupt Enable register contains individual enable bits for each of the bits in the Ethernet Interrupt Status register. When these bits are cleared, the corresponding bit in the Ethernet Interrupt Status register cannot cause the interrupt signal to the system to be asserted when it is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						EN_RX OVFL_ DATA	EN_RX OVFL_ STAT	EN_ RX BUFC	EN_RX DONE A	EN_RX DONE B	EN_RX DONE C	EN_RX DONE D	EN_ RXNO BUF	EN_RX BUF FUL	EN_ RXBR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									EN_ST OVFL	Not used	EN_TX BUFC	EN_TX BUF NR	EN_ TX DONE	EN_ TX ERR	EN_ TX IDLE

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:26	N/A	Reserved	N/A	N/A
D25	R/W	EN_RXOVFL_DATA	0	Enable the RXOVFL_DATA interrupt bit.
D24	R/W	EN_RXOVFL_STAT	0	Enable the RXOVFL_STATUS interrupt bit.
D23	R/W	EN_RXBUFC	0	Enable the RXBUFC interrupt bit.
D22	R/W	EN_RXDONEA	0	Enable the RXDONEA interrupt bit.
D21	R/W	EN_RXDONEB	0	Enable the RXDONEB interrupt bit.
D20	R/W	EN_RXDONEC	0	Enable the RXDONEC interrupt bit.
D19	R/W	EN_RXDONED	0	Enable the RXDONED interrupt bit.
D18	R/W	EN_RXNOBUF	0	Enable the RXNOBUF interrupt bit.
D17	R/W	EN_RXBUFFUL	0	Enable the RXBUFFUL interrupt bit.
D16	R/W	EN_RXBR	0	Enable the RXBR interrupt bit.
D15:07	N/A	Reserved	N/A	N/A
D06	R/W	EN_STOVFL	0	Enable the STOVFL interrupt bit.
D05	R/W	Not used	0	Always write as 0.
D04	R/W	EN_TXBUFC	0	Enable the TXBUFC interrupt bit.
D03	R/W	EN_TXBUFNR	0	Enable the TXBUFNR interrupt bit.
D02	R/W	EN_TXDONE	0	Enable the TXDONE interrupt bit.
D01	R/W	EN_TXERR	0	Enable the TXERR interrupt bit.
D00	R/W	EN_TXIDLE	0	Enable the TXIDLE interrupt bit.

Table 241: Ethernet Interrupt Enable register

TX Buffer Descriptor Pointer register

Address: A060 0A18

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXPTR							

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A
D07:00	R/W	TXPTR	0x00	Contains a pointer to the initial transmit buffer descriptor in the TX buffer descriptor RAM. **Note:** This pointer is the 8-bit physical address of the TX buffer descriptor RAM, and points to the first location of the four-location buffer descriptor. The byte offset of this buffer descriptor can be calculated by multiplying this value by 4.

Table 242: TX Buffer Descriptor Pointer register

Transmit Recover Buffer Descriptor Pointer register

Address: A060 0A1C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXRPTR							

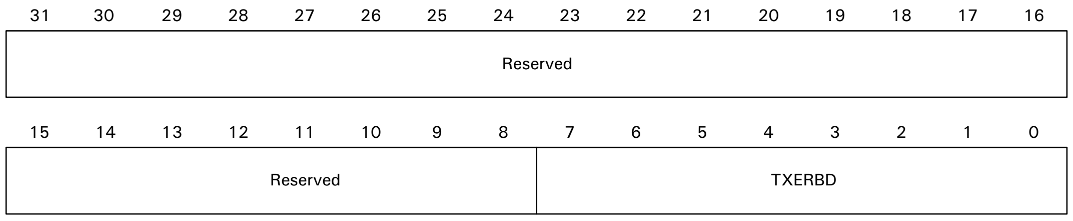
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A
D07:00	R/W	TXRPTR	0x00	Contains a pointer to a buffer descriptor in the TX buffer descriptor RAM. **Note:** This pointer is the 8-bit physical address of the TX buffer descriptor RAM, and points to the first location of the four-location buffer descriptor. The byte offset of this buffer descriptor can be calculated by multiplying this value by 4. This is the buffer descriptor at which the TX_WR logic resumes processing when TCLER is toggled from low to high in Ethernet General Control Register #2 (see page 342).

Table 243: Transmit Recover Buffer Descriptor Pointer register

TX Error Buffer Descriptor Pointer register

Address: A060 0A20



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A

Table 244: TX Error Buffer Descriptor Pointer register

Bits	Access	Mnemonic	Reset	Description
D07:00	R	TXERBD	0x00	Contains the pointer (in the TX buffer descriptor RAM) to the last buffer descriptor of a frame that was not successfully transmitted. TXERBD is loaded by the TX_WR logic when a transmit frame is aborted by the MAC or when the MAC finds a CRC error in a frame. TXERBD also is loaded if a buffer descriptor that is not the first buffer descriptor in a frame does not have its F bit set. **Note:** This pointer is the 8-bit physical address of the TX buffer descriptor RAM, and points to the first location of the four-location buffer descriptor. The byte offset of this buffer descriptor can be calculated by multiplying this value by 4. **Note:** Software uses TXERBD to identify frames that were not transmitted successfully.

Table 244: TX Error Buffer Descriptor Pointer register

RX_A Buffer Descriptor Pointer Offset register

Address: A060 0A28

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RXAOFF										

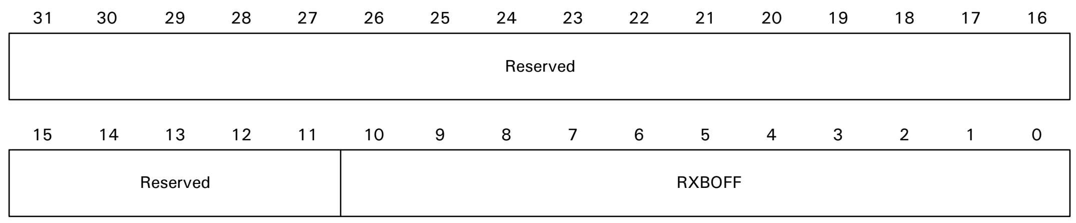
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:11	N/A	Reserved	N/A	N/A
D10:00	R	RXAOFF	0x000	Contains an 11-bit byte offset from the start of the pool A ring. The offset is updated at the end of the RX packet, and will have the offset to the next buffer descriptor that will be used. RXAOFF can be used to determine where the RX_RD logic will put the next packet.

Table 245: RX_A Buffer Descriptor Pointer Offset register

RX_B Buffer Descriptor Pointer Offset register

Address: A060 0A2C



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:11	N/A	Reserved	N/A	N/A
D10:00	R	RXBOFF	0x000	Contains an 11-bit byte offset from the start of the pool B ring. The offset is updated at the end of the RX packet, and will have the offset to the next buffer descriptor that will be used. RXBOFF can be used to determine where the RX_RD logic will put the next packet.

Table 246: RX_B Buffer Descriptor Pointer Offset register

RX_C Buffer Descriptor Pointer Offset register

Address: A060 0A30

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RXCOFF										

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:11	N/A	Reserved	N/A	N/A
D10:00	R	RXCOFF	0x000	Contains an 11-bit byte offset from the start of the pool C ring. The offset is updated at the end of the RX packet, and will have the offset to the next buffer descriptor that will be used. RXCOFF can be used to determine where the RX_RD logic will put the next packet.

Table 247: RX_C Buffer Descriptor Pointer Offset register

RX_D Buffer Descriptor Pointer Offset register

Address: A060 0A34

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RXDOFF										

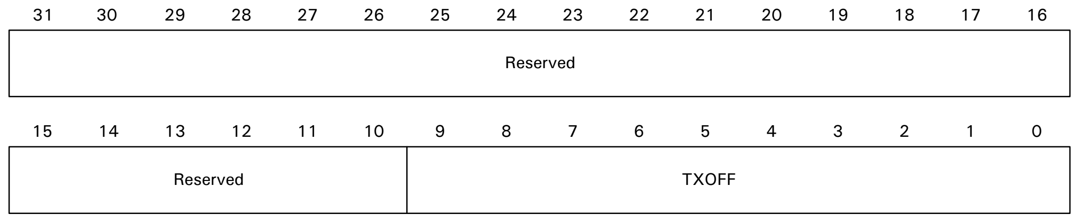
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:11	N/A	Reserved	N/A	N/A
D10:00	R	RXDOFF	0x000	Contains an 11-bit byte offset from the start of the pool D ring. The offset is updated at the end of the RX packet, and will have the offset to the next buffer descriptor that will be used. RXDOFF can be used to determine where the RX_RD logic will put the next packet.

Table 248: RX_D Buffer Descriptor Pointer Offset register

Transmit Buffer Descriptor Pointer Offset register

Address: A060 0A38



Register bit assignment

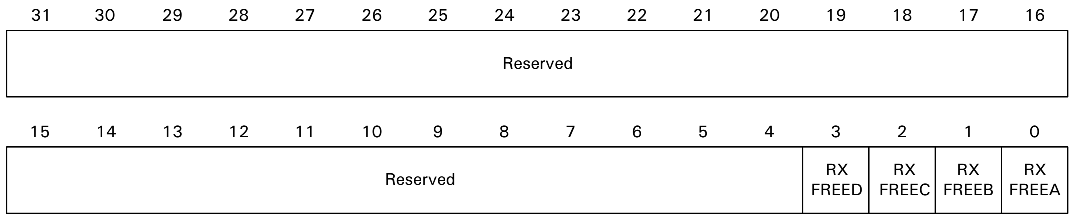
Bits	Access	Mnemonic	Reset	Description
D31:10	N/A	Reserved	N/A	N/A
D09:00	R	TXOFF	0x000	Contains a 10-bit byte offset from the start of the transmit ring in the TX buffer descriptor RAM. The offset is updated at the end of the TX packet, and will have the offset to the next buffer descriptor that will be used. TXOFF can be used to determine from where the TX_WR logic will grab the next packet.

Table 249: TX Buffer Descriptor Pointer Offset register

RX Free Buffer register

Address: A060 0A3C

So the RX_RD logic knows when the software is freeing a buffer for reuse, the software writes to the RXFREE register each time it frees a buffer in one of the pools. RXFREE has an individual bit for each pool; this bit is set to 1 when the register is written. Reads to RXFREE always return all 0s.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	N/A	Reserved	N/A	N/A
D03	W	RXFREED	0	Pool D free bit
D02	W	RXFREEC	0	Pool C free bit
D01	W	RXFREEB	0	Pool B free bit
D00	W	RXFREEA	0	Pool A free bit

Table 250: RX Free Buffer register

TX buffer descriptor RAM

Address: A060 1000

The TX buffer descriptor RAM holds 64 transmit buffer descriptors on-chip. Each buffer descriptor occupies four locations in the RAM, and the RAM is implemented as a 256x32 device. This is the format of the TX buffer descriptor RAM:

Offset+0

D31:00	R/W	Source address

Offset+4

D31:11	R/W	Not used
D10:00	R/W	Buffer length

Offset+8

D31:00	R/W	Destination address (not used)

Offset+C

D31	R/W	W	Wrap
D30	R/W	I	Interrupt on buffer completion
D29	R/W	L	Last buffer on transmit frame
D28	R/W	F	Buffer full
D27:16	R/W	Reserved	N/A
D15:00	R/W	Status	Transmit status from MAC

See Figure 67, "Transmit buffer descriptor format," on page 327, for more information about the fields in *Offset+C*.

Sample hash table code

This sample C code describes how to calculate hash table entries based on 6-byte Ethernet destination addresses and a hash table consisting of two 32-bit registers (HT1 and HT2). HT1 contains locations 31:0 of the hash table; HT2 contains locations 63:32 of the hash table.

The pointer to the hash table is bits [28:23] of the Ethernet destination address CRC. The polynomial is the same as that used for the Ethernet FCS:

G(x) = *x*^32+*x*^26+*x*^23+*x*^22+*x*^16+*x*^12+*x*^11+*x*^10+*x*^8+*x*^7+*x*^5+*x*^4+*x*^2+*x*+1

```
static ETH_ADDRESS mca_address[MAX_MCA];          /*list of MCA addresses*/
static INT16 mca_count;                           /*# of MCA addresses*/

/     *
      *
      * Function: void eth_load_mca_table (void)
      *
      * Description:
      *
      * This routine loads the MCA table. It generates a hash table for
      * the MCA addresses currently registered and then loads this table
      * into the registers HT1 and HT2.
      *
      * Parameters:
      *
      *        none
      *
      * Return Values:
      *
      *        none
      *
      */

static void eth_load_mca_table (void)

{
              WORD32 has_table[2];
```

```
        // create hash table for MAC address
        eth_make_hash_table (hash_table);

            (*MERCURY_EFE) .ht2.bits.data = SWAP32(hash_table[1]);
            (*MERCURY_EFE) .ht1.bits.data = SWAP32(hash_table[0]);
}


/   *
    *
    * Function: void eth_make_hash_table (WORD32 *hash_table)
    *
    * Description:
    *
    *           This routine creates a hash table based on the CRC values of
    *           the MAC addresses setup by set_hash_bit(). The CRC value of
    *           each MAC address is calculated and the lower six bits are used
    *           to generate a value between 0 and 64. The corresponding bit in
    *           the 64-bit hash table is then set.
    *
    *     Parameters:
    *
    *           hash_table                pointer to buffer to store hash table in.
    *
    * Return Values:
    *
    *           none
    *
    */

static void eth_make_hash_table (WORD32 *hash_table)

{
        int index;

        memset (hash_table, 0, 8);                            /* clear hash table*/

        for (index = 0; index < mca_count; index++)           /*for each mca address*/
```

```
            {
                set_hash_bit ((BYTE *) hash_table, calculate_hash_bit (mca_address
                [index]));
            }
}


/    *
     *
     * Function: void set_hash_bit (BYTE *table, int bit)
     *
     * Description:
     *
     *          This routine sets the appropriate bit in the hash table.
     *
     * Parameters:
     *
     *          table               pointer to hash table
     *          bit                 position of bit to set
     *
     * Return Values:
     *
     *          none
     *
     */

static void set_hash_bit (BYTE *table, int bit)

{
            int byte_index, bit_index;

            byte-index = bit >> 3;
            bit_index = bit & 7;
            table [byte_index] |= (1 << bit_index);
}


/    *
     *
```

```
    * Function: int calculate_hash_bit (BYTE *mca)
    *
    * Description:
    *           This routine calculates which bit in the CRC hash table needs
    *           to be set for the MERCURY to recognize incoming packets with
    *           the MCA passed to us.
    *
    * Parameters:
    *
    *           mca                     pointer to multi-cast address
    *
    * Return Values:
    *
    *           bit position to set in hash table
    *
    */

#define POLYNOMIAL 0x4c11db6L

static int calculate_hash_bit (BYTE *mca)

{
          WORD32 crc;
          WORD16 *mcap, bp, bx;
          int result, index, mca_word, bit_index;
          BYTE lsb;
          WORD16 copy_mca[3]
          memcpy (copy_mca,mca,sizeof(copy_mca));
          for (index = 0; index < 3; index++)
          {
              copy_mca [index] = SWAP16 (copy_mca [index]);
          }

          mcap = copy_mca;
          crc = 0xffffffffL;

          for (mca_word = 0; mca_word < 3; mca_word++)
          {
              bp = *mcap;
```

```
        mcap++;
        for (bit_index = 0; bit_index < 16; bit_index++)
        {
                bx = (WORD16) (crc >> 16);              /* get high word of crc*/
                bx = rotate (bx, LEFT, 1);              /* bit 31 to lsb*/
                bx ^= bp;                               /* combine with incoming*/
                crc <<= 1;                              /* shift crc left 1 bit*/
                bx &= 1;                                /* get control bit*/
                if (bx)                                 /* if bit set*/
                {
                 crc ^= POLYNOMIAL;                     /* xero crc with polynomial*/
                }
                crc |= bx:                              /* or in control bit*/
                bp = rotate (bp, RIGHT, 1);
        }
    }

    // CRC calculation done. The 6-bit result resides in bit
    // locations 28:23

    result = (crc >> 23) & 0x3f;

    return result;

}
```

Sample hash table code

PCI-to-AHB Bridge

C H A P T E R 7

The PCI-to-AHB bridge provides connections between PCI-based modules/devices and the NS9750 AHB bus.

Important: This chapter presumes knowledge of PCI system standards and architecture, and explains how PCI works in relation to the AHB bus. If you have questions regarding PCI terminology or concepts, please refer to your PCI documentation.

About the PCI-to-AHB Bridge

The PCI-to-AHB bridge provides these features:

- Supports PCI specification 2.1 and 2.2 protocol
- AHB master and slave interfaces
- PCI master and target interfaces
- Open drain interrupt output for PCI bus
- Interrupt to AHB bus for AHB and PCI bus errors
- Supports 32-bit address and data on both busses
- Supports AHB core clock up to 100 MHz, and a PCI core clock of 33 MHz
- Performs all AHB-to-PCI reads as SPLIT transactions on the AHB bus (improves bus use)
- Supports AHB burst transfers up to 8 words
- AHB master supports all AHB slave responses for upstream (initiated on the PCI bus) PCI-to-AHB traffic
- Supports early burst termination on the AHB bus
- Dual 64-byte write buffers in both directions
- Single 64-byte read buffers in both directions
- Supports PCI configuration cycles
- Supports target retry, target disconnect, and target abort on PCI bus
- Supports all PCI parity generation and parity error detection
- Includes all PCI-specific configuration registers
- Supports configuration of internal PCI configuration registers using the AHB bus
- Supports PCI-to-AHB address translation
- Supports AHB-to-PCI address translation

Note: Use the AHB DMA function to move blocks of data between the ARM CPU and the PCI bus. Do not use load and store multiple commands to the PCI-to- AHB bridge and do not cache PCI memory.

PCI-to-AHB bridge functionality

Figure 71 shows the PCI-to-AHB bridge. *Downstream* transactions are those initiated on the AHB bus; *upstream* transactions are those initiated on the PCI bus.

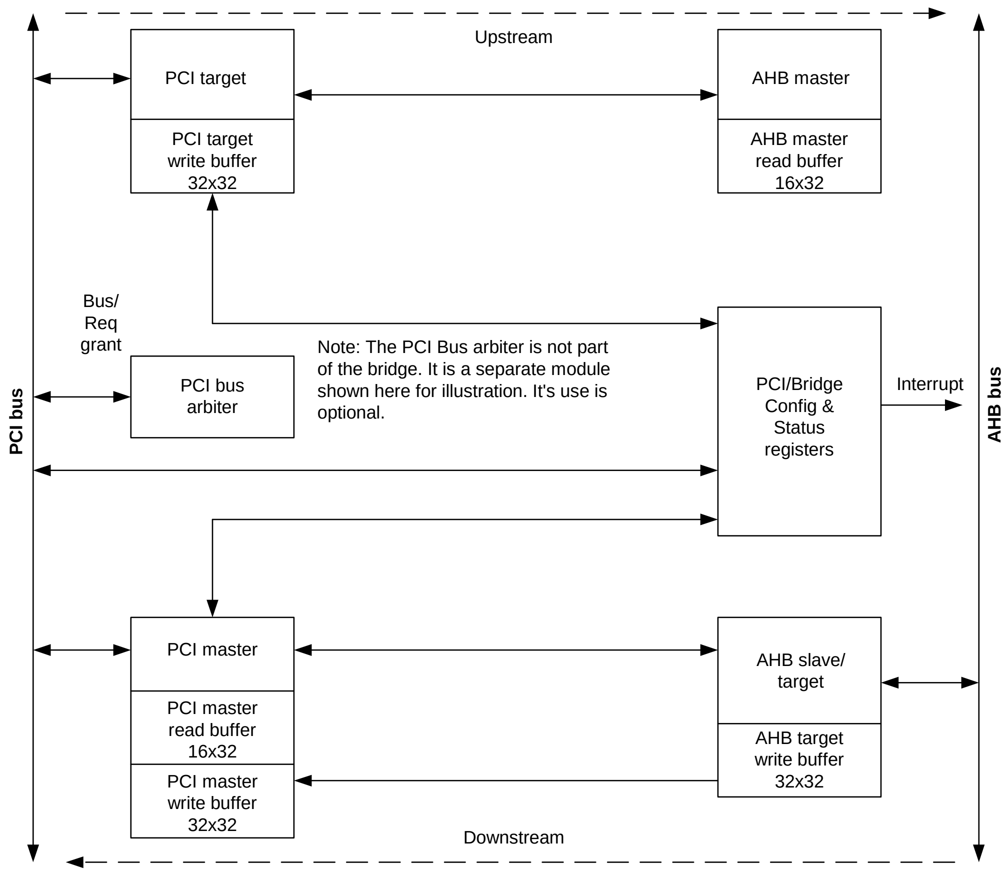

Upstream
PCI target
PCI target write buffer 32x32
AHB master
AHB master read buffer 16x32
Bus/ Req grant
PCI bus arbiter
Note: The PCI Bus arbiter is not part of the bridge. It is a separate module shown here for illustration. It's use is optional.
PCI/Bridge Config & Status registers
Interrupt
PCI bus
AHB bus
PCI master
PCI master read buffer 16x32
PCI master write buffer 32x32
AHB slave/ target
AHB target write buffer 32x32
Downstream

Figure 71: PCI-to-AHB bridge diagram

AHB master interface

The AHB master interface block controls the bridge's access to the AHB bus as a master, and is used for reads and writes to the AHB bus that are initiated on the PCI bus by an external PCI bus master. The requests are transferred to the AHB master from the PCI target interface. PCI writes are posted in the dual 64-byte PCI target write buffer. AHB read data is stored in the 64-byte AHB master read buffer before being sent back to the PCI bus. The AHB master interface supports both single and burst transactions.

AHB slave/target interface

The AHB slave/target interface block controls the AHB target access to the bridge, and is used for reads and writes to the PCI bus that are initiated on the AHB bus. The requests are transferred to the PCI master interface. Writes are posted in the dual 64-byte AHB target write buffer and then transferred to the dual 64-byte PCI master write buffer. Reads are delayed through the AHB SPLIT transaction protocol. Read data is stored in the 64-byte PCI master read buffer. The AHB slave/target interface supports both single and burst transactions.

The AHB slave provides the AHB interface to the PCI/Bridge configuration registers (see page 411), using the CONFIG_ADDR and CONFIG_DATA memory spaces.

PCI target interface

The PCI target interface block controls the PCI bus access to the AHB bus, and is used for reads and writes that are initiated on the PCI bus by an external PCI bus master. The requests are transferred to the AHB master interface. Writes are posted in the dual 64-byte PCI target write buffer. The PCI target interface supports both single and burst transactions.

PCI master interface

The PCI master interface block controls the bridge's access to the PCI bus as a master, and is used for reads and writes to the PCI bus that are initiated on the AHB bus. The PCI master interfaces to the AHB slave/target interface, and receives its requests from the AHB target interface. The PCI master has both a 64-byte read buffer and a dual
64-byte write buffer. Both single and burst transactions are supported.

PCI/bridge configuration and status registers

The PCI/bridge configuration and status registers block contains standard PCI configuration and status registers. All registers can be accessed using the PCI and AHB buses (see "Configuration registers," beginning on page 411).

PCI bus arbiter

The PCI bus arbiter (also referred to as *PCI arbiter*), although embedded in NS9750, is not part of the PCI-to-AHB bridge protocol. See "PCI bus arbiter," beginning on page 418, for information about the PCI arbiter. The arbiter's use is optional.

Cross-bridge transaction error handling

The PCI-to-AHB bridge supports several error-handling mechanisms. All mechanisms can cause an interrupt to the system unless they are masked. Table 251 describes the errors and resulting action.

Error	Action taken
PCI-to-AHB write A write to the AHB bus does not complete due to receipt of an AHB bus error	■ The AHBERR bit in the PCI Bridge Interrupt Status register is set. ■ The AHB address is stored in the AHB Error Address register in the bridge.
PCI-to-AHB read A read to the AHB bus does not complete due to receipt of an AHB bus error	■ The AHBERR bit in the PCI Bridge Interrupt Status register is set. ■ The AHB address is stored in the AHB Error Address register in the bridge. ■ A PCI target abort is issued to release the PCI bus ■ The SIGNALED TARGET ABORT bit in the PCI Status register is set.
AHB-to-PCI write A write to the PCI bus does not complete due to receipt of a PCI bus error.	■ Bits [15:11] and [04] in the PCI Status register will indicate the source of the PCI bus error. ■ The PCI address is stored in the PCI Error Address register in the bridge.
AHB-to-PCI read A read to the PCI bus does not complete due to receipt of a PCI bus error.	■ Bits [15:11] and [04] in the PCI Status register will indicate the source of the PCI bus error ■ The PCI address is stored in the PCI Error Address register in the bridge.

Table 251: PCI-to-AHB error handling

PCI target error filtering

If the PCI target address or data parity checking logic finds an error during upstream transactions, the transaction is not passed to the AHB master. In this situation, the

DETECTED PARITY ERROR bit in the PCI Status register is set. For address parity errors, the SIGNALED TARGET ABORT bit in the PCI Status register is set.

For data parity checking on writes, the entire burst is discarded if any word in the burst has a parity error.

AHB address decoding and translation

The PCI-to-AHB bridge supports these four AHB address spaces:

- PCI memory (0x8000_0000->0x8FFF_FFFF; 256 MB)
- PCI IO (0xA000_0000->0xA00F_FFFF; 1 MB)
- PCI CONFIG_ADDR register (0xA010_0000)
- PCI CONFIG_DATA register (0xA020_0000)

The bridge supports AHB to PCI memory address translation using the PCI Bridge AHB to PCI Memory Address Translate 0/1 (see page 437 and page 438) and PCI Bridge Address Translation Control (see page 441) registers. The address translation scheme breaks the 256 MB memory window from AHB to PCI into eight 32 MB subwindows that can be translated individually.

The PALTxVAL fields in the PCI Bridge AHB to PCI Memory Address Translate 0/1 registers control the translation for each of the eight subwindows. For example, if PALT0VAL is set to 0x75, an access to 0x8000_0000 on the AHB bus is mapped to 0xEA00_0000 in the PCI bus. The PALT_EN bit in the PCI Bridge Address Translation Control register determines whether AHB to PCI address translation is enabled:

- When set to 1, PALT_EN enables address translation.
- When set to 0, no address translation takes place, and the AHB and PCI addresses are identical.

Address translation also is provided for accesses to PCI IO space from the AHB bus. The translation process is similar to memory address translation, with this exception: the PALT8VAL field translates the 1 MB window dedicated to PCI IO space to another 1 MB IO window on the PCI bus. When PALT_EN is set to 1, IO translation is enabled.

PCI address decoding and mapping

The PCI-to-AHB bridge uses six Base Address registers (BAR), defined in the PCI Configuration register (see Table 254, "PCI/bridge configuration registers," on page 413), to determine the range of PCI addresses to which the bridge responds. The window size of each Base Address register is hardwired (see Table 257 on page 417), but each register can be enabled or disabled using the ENBAR0–ENBAR5 bits in the PCI Miscellaneous Support register (see page 426) in the PCI arbiter.

The bridge supports PCI to AHB memory address translation using the PCI Bridge PCI to AHB Memory Address Translate 0/1(see page 439) and PCI Bridge Address Translation Control (see page 441) registers. The address translation scheme provides a separate translation value for each of the six Base Address registers. The translation window size is the same as the size of the corresponding register.

The MALTxVAL fields in the PCI Bridge PCI to AHB Memory Address Translate 0/1 registers (see page 439 and page 440) control the translation for each of the six Base Address registers. For example, if MALT1VAL is set to 0x08, an access to 0xFC00_0000 on the PCI bus that hits Base Address register 1 is mapped to 0x2000_0000 on the AHB bus. The MALT_EN bit in the PCI Bridge Address Translation Control register determines whether PCI to AHB address translation is enabled:

- When set to 1, MALT_EN enables address translation.
- When set to 0, no address translation takes place, and the AHB and PCI addresses are identical.

The external PCI bus is allowed access only to NS9750's system memory. The MALTxVAL values, therefore, should be programmed only to map addresses in the lower 2 GB of NS9750's 4 GB address space (0x0000_0000 -> 0x7FFF_FFFF)

Interrupts

The bridge generates an interrupt to the AHB bus, for either AHB or PCI bus errors. An AHB bus error interrupt is generated when the AHB master receives an ERROR response to a transaction it initiated. The status bit for this interrupt, AHBERR, is in the PCI Bridge Interrupt Status register (see page 434).

A PCI bus error interrupt is generated for any of these PCI conditions:

- Address or data parity error detected (DPE, see "Detected parity error" on page 415)
- Bridge-generated system error (SERR#, see "Signaled system error" on page 415)
- Bridge receives a master abort (RMA, see "Received master abort" on page 415)

- Bridge receives a target abort (RTA, see "Received target abort" on page 415)
- Bridge signals a target abort (STA, see "Signaled target abort" on page 415)
- Bridge master finds a parity error on read data or detects the target asserting a master data parity error (PERR#, see "Master data parity error" on page 415) and the parity error response bit in the PCI Command register (see page 414) is set.

The PCI Status register (see page 415) contains the status bits for the interrupts caused by PCI bus errors.

Use the PCI Bridge Interrupt Enable register (see page 435) to enable or disable interrupt sources. Clearing an enable bit (setting the bit to 0) prevents the associated interrupt status bit from asserting the external interrupt to the system.

When an AHB bus error occurs, the AHB address that caused the bus error is saved in the AHB Address Error register (see "PCI Bridge AHB Error Address register" on page 433). Because multiple errors can occur before the software services the interrupt, no new addresses are saved in the register until AHBERR (in the PCI Bridge Interrupt Status register) is cleared.

When a PCI bus error occurs, the PCI address that caused the bus error is saved in the PCI Address Error register ("PCI Bridge PCI Error Address register" on page 433). Because multiple bus errors can occur before the software services the interrupt, no new addresses are saved in the register until all error bits are cleared in the PCI Status register.

The bridge can drive an interrupt to the PCI bus. This interrupt is driven from the INTA2PCI bit in the PCI Miscellaneous Support register (see page 426) in the PCI arbiter. This interrupt is used only in systems in which NS9750 is not processing PCI interrupts, and is set by software.

Transaction ordering

The AHB-to-PCI bridge maintains the request order in each direction. Transactions are sent to the destination bus in the order in which they are received on the source bus. No order is maintained between upstream and downstream transactions.

Endian configuration

The PCI bus is defined as little endian and the AHB bus can be defined as either Big or little endian. The PCI-to-AHB bridge supports byte-swapping only when the AHB bus is configured as a big endian bus. Byte-swapping is selected using the endian mode bit in the Miscellaneous System Configuration register (see "Miscellaneous System Configuration and Status register," beginning on page 296). Table 252 shows the byte-swapping scheme used.

PCI bus byte	AHB bus byte
Data[31:24]	AHB_Data[7:0]
Data[23:16]	AHB_Data[15:8]
Data[15:08]	AHB_Data[23:16]
Data[07:00]	AHB_Data[31:24]

Table 252: Big endian byte-swapping

Configuration registers

The Configuration registers within the PCI-to-AHB bridge are accessed using PCI configuration cycles. Two registers are used to access the configuration registers from the AHB side: Configuration Address Port (CONFIG_ADDR) and Configuration Address Data Port (CONFIG_DATA).

Table 253 describes the fields in the Configuration Address Port register. The Configuration Address Data Port register has no specific format; it contains the read or write configuration data.

Bits	Access	Mnemonic	Reset	Description
D31	R/W	ENABLE	0x0	**Enable translation** 0 Disabled (default) 1 Enabled Enables translation of a subsequent access to the CONFIG_DATA register to a PCI configuration cycle.
D30:24	N/A	Reserved	N/A	N/A

Table 253: CONFIG_ADDR register

Bits	Access	Mnemonic	Reset	Description
D23:16	R/W	BUS_NUMBER	0x00	**Target PCI bus number** ■ **Bus 0.** Considered a local bus, so a Type 0 configuration is performed. ■ **All other bus numbers.** Result in a Type 1 cycle that targets an external bus (that is, a bus on the other side of a PCI-to-PCI bridge).
D15:11	R/W	DEVICE_NUMBER	0x00	**Target PCI device number** For Type 0 cycles, the bridge uses this field to determine which bit of AD[31:11] is the only bit set during the configuration cycle on the PCI bus. This will be the bit that is connected to the IDSEL input of the device assigned to this device number. **Note:** This value must be 0 for internal registers. **Example:** If DEVICE_NUMBER = 0x01, AD[31:11] = 0x000002
D10:08	R/W	FUNCTION_NUMBER	0x0	**Target function number within PCI device** This value is 0x0 for all accesses to bridge registers. This value is mapped to AD[10:08] during the configuration cycle.
D07:02	R/W	REGISTER_NUMBER	0x00	**Target register address within the PCI** 32-bit register, whose value is mapped to AD[07:02] during the configuration cycle. **Example:** If the PCI Vendor ID register is being accessed, the REGISTER_NUMBER will be 0x00. (See Table 254, “PCI/bridge configuration registers,” on page 413 for more information.)
D01:00	R/W	TYPE	0x0	**Type field** The value in this field must be 00 for Type 0 cycles and 01 for Type 1 cycles. This value is mapped to AD[01:00] during the configuration cycle.

Table 253: CONFIG_ADDR register

Bridge Configuration registers

Table 254 shows the standard PCI configuration registers that are supported by the PCI-to-AHB bridge. These registers can be 8-, 16-, or 32-bits wide, as indicated in the table. The size of the transfer on the AHB bus determines which bytes are written.

All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

The registers are described briefly in this chapter. For more information about each register, see your PCI documentation.

Note: The register number refers to the REGISTER_NUMBER field in the Configuration Address Port register (see Table 253, "CONFIG_ADDR register," on page 411).

Register number	[31:24]	[23:16]	[15:08]	[07:00]
0x00	Device ID[1]		Vendor ID[1]	
0x01	Status		Command	
0x02	Class code[1]			Revision ID[1]
0x03	BIST	Header	Latency timer	Cache size
0x04	Base address 0			
0x05	Base address 1			
0x06	Base address 2			
0x07	Base address 3			
0x08	Base address 4			
0x09	Base address 5			
0x0A	CardBus CIS pointer			
0x0B	Subsystem ID[1]		Subsystem vendor ID[1]	
0x0C	Expansion ROM			
0x0D	Reserved			
0x0E	Reserved			
0x0F	Max_Lat[1]	Min_Gnt[1]	Interrupt pin[1]	Interrupt Line

Table 254: PCI/bridge configuration registers

Register number	[31:24]	[23:16]	[15:08]	[07:00]

1 These entries are standard read-only PCI configuration registers that are initialized using registers in the PCI arbiter (see "PCI bus arbiter," beginning on page 418).

Table 254: PCI/bridge configuration registers

PCI Vendor ID register

Read-only value. To change this value, use the VENDOR_ID field in the PCI Configuration 0 register in the PCI arbiter (see page 428).

PCI Device ID register

Read-only value. To change this value, use the DEVICE_ID field in the PCI Configuration 0 register in the PCI arbiter (see page 428).

PCI Command register

The default value of this register is 0x0000. Table 255 describes the register fields.

Bits	Description	Type
D15:10	Reserved	Hardwired to 0
D09	Fast back-to-back	Hardwired to 0; the device cannot generate fast back-to-back cycles.
D08	SERR#	R/W
D07	Address stepping	Hardwired to 0
D06	Parity error response	R/W
D05	VGA palette snooping	Hardwired to 0
D04	Master MWI (set to 0 for NS9750)	R/W
D03	Special cycle response	Hardwired to 0
D02	Bus master (set to 1 for NS9750)	R/W
D01	Memory enable (set to 1 for NS9750)	R/W
D00	IO enable (set to 0 for NS9750)	R/W

Table 255: Command register

PCI Status register

Table 256 describes the PCI Status register fields.

Bits	Access	Mnemonic	Reset	Description
D15	R/C	DPE	0	**Detected parity error** Device detected parity error. Used as an interrupt source to AHB bus.
D14	R/C	SERR#	0	**Signaled system error** Device generated system error (SERR#). Used as an interrupt source to AHB bus.
D13	R/C	RMA	0	**Received master abort** Master aborted transaction. Used as an interrupt source to AHB bus.
D12	R/C	RTA	0	**Received target abort** Master received target abort. Used as an interrupt source to AHB bus.
D11	R/C	STA	0	**Signaled target abort** Master signaled the target abort as target. Used as an interrupt source to AHB bus.
D10:09	Hard-wired to 10	DEVSEL	10	**DEVSEL timing for target** 00 Fast 01 Medium 10 Slow 11 Reserved
D08	R/C	PERR#	0	**Master data parity error** The master detected a parity error and the following conditions exist: ■ Master initiated transaction ■ Master set PERR# (read) or detected PERR# asserted by target (write) ■ Parity error response bit set in the PCI Command register Used as an interrupt source to the AHB bus.

Table 256: PCI Status register

Bits	Access	Mnemonic	Reset	Description
D07	Hard-wired to 1	FBBC	1	**Fast back-to-back capable** 0 No support 1 Support Device supports fast back-to-back transactions as a target only.
D06	N/A	Not used	0	Hardwired to 0.
D05	Hard-wired to 0	BS66	0	**66MHz capable** Bus speed: 0 33 MHz 1 66MHz
D04:00	N/A	Not used	0	Always set to 0.

Table 256: PCI Status register

PCI Revision ID register

Read-only value. To change this value, use the REVISION_ID field in the PCI Configuration 1 register in the PCI arbiter (see page 429).

PCI Class Code register

Read only value. To change this value, use the CLASS_CODE field in the PCI Configuration 1 register in the PCI arbiter (see page 429).

PCI Cache Size register

Read/write value that should always be set to 0x00. The bridge ignores this value.

PCI Latency Timer register

Read/write field programmed by the device driver.

PCI Header register

Read-only value, hardwired to 0x0.

PCI BIST register

Read-only value, hardwired to 0x0.

PCI Base Address registers [5:0]

The PCI-to-AHB bridge supports the six Base Address registers defined by PCI. Table 257 defines the memory space size decoded by each register.

Base Address register	Memory size decoded
0	256 MB
1	64 MB
2	16 MB
3	4 MB
4	1 MB
5	256 KB

Table 257: Base Address register decoding sizes

Each Base Address register is enabled using the ENBAR0–ENBAR5 bits in the PCI Miscellaneous Support register (see page 426) in the PCI arbiter. Note that the bridge forces the four least significant bits (LSBs) of each Base Address register to 0x0. As such, PCI defines each register with the following characteristics:

- Memory space indicator
- Located anywhere in the 32-bit address space
- Not prefetchable

PCI CardBus CIS Pointer register

Read-only value, hardwired to 0x0.

PCI Subsystem Vendor ID register

Read-only value. To change this value, use the SUBVENDOR_ID field in the PCI Configuration 2 register (see page 430) in the PCI arbiter.

PCI Subsystem ID register

Read-only value. To change this value, use the SUBSYSTEM_ID field in the PCI Configuration 2 register (see page 430) in the PCI arbiter.

PCI Expansion ROM register

Read-only value, hardwired to 0x00000000.

PCI Interrupt Line register

Read/write value indicating to which line of an interrupt controller the PCI interrupt generated by the bridge is connected. This register is used only in those systems in which NS9750 is not handling PCI interrupts.

PCI Interrupt Pin register

Read-only value programmed using the INTERRUPT_PIN field in the PCI Configuration 3 register (see page 431) in the PCI arbiter. Set this value to 0x1 (default value) when NS9750 drives INTA#.

PCI Min Grant register

Read-only value programmed using the MIN_GRANT field in the PCI Configuration 3 register (see page 431) in the PCI arbiter.

PCI Max Latency register

Read-only value programmed using the MAX_LATENCY field in the PCI Configuration 3 register (see page 431) in the PCI arbiter.

PCI bus arbiter

NS9750 provides an embedded PCI bus arbiter that supports up to three external PCI masters and the internal PCI-to-AHB bridge. The arbiter uses a rotating priority scheme. An AHB slave is integrated with the PCI bus arbiter to access programmable registers, to support system configuration and error reporting.

NS9750 can be configured to use either the embedded PCI arbiter or an external arbiter through the bootstrap initialization scheme used during powerup (see "Bootstrap initialization" on page 272). The RTCK pin selects the source of the arbiter:

- The internal arbiter is used if RTCK = 1.
- If a pulldown resistor is placed on the RTCK bit, an external arbiter is used. When an external arbiter is used, the REQ#/GNT# (request/grant) signals for the internal PCI-to-AHB bridge are brought to external pins on NS9750 (see Figure 73, "System connections to NS9750 — External arbiter and central resources," on page 457).

PCI arbiter functional description

The PCI bus arbiter supports up to four PCI masters, including the PCI-to-AHB bridge, using the rotating priority scheme shown in Table 258. With rotating priority, the priority of a master depends on its relative position to the last granted master. After reset, the arbiter defaults to the PCI-to-AHB bridge having the highest priority.

Last granted master	Highest priority	2nd priority	3rd priority	Lowest priority	Parked master
PCI-to-AHB bridge	External master 1	External master 2	External master 3	PCI-to-AHB bridge	PCI-to-AHB bridge
External master 1	External master 2	External master 3	PCI-to-AHB bridge	External master 1	External master 1
External master 2	External master 3	PCI-to-AHB bridge	External master 1	External master 2	External master 2
External master 3	PCI-to-AHB bridge	External master 1	External master 2	External master 3	External master 3

Table 258: Rotating priority scheme

Each master has a set of REQ#/GNT# signals used for bus arbitration. The master asserts its REQ# when it needs to execute a bus transaction. The arbiter then asserts the GNT# to the requester with the highest priority. Until the bus is idle (that is, FRAME# and IRDY# are both inactive), the arbiter continually arbitrates and the asserted GNT# can change each clock cycle. One GNT# can be negated coincident with another GNT# being asserted in this situation. When the bus goes idle, the arbiter stops arbitrating and the master with the asserted GNT# is allowed to start a transaction.

If there are no new requesters when the current bus master completes its transaction, the bus ownership stays with the most recent bus master (*bus parking*). If a REQ# is asserted from any of the other masters, there must be a one clock cycle delay between the negation of the GNT# to the parked bus master and the assertion of the GNT# to the bus master requesting the bus. If the granted bus master does not start its bus transaction within 16 PCI clocks of the bus being idle, the PCI arbiter sets the PCIBRK_Mx bit for that master (in the PCI Arbiter Interrupt Status register; see page 424) and negates GNT#. The bus ownership can then be granted to one of the other bus masters. For the three external masters, REQ# from the broken master is ignored until the interrupt service routine re-enables it by toggling its PCIEN_Mx bit from low-to-high in the PCI Arbitration Configuration register (see page 423). (Although a broken master condition for the PCI-to-AHB bridge is logged using the PCIBRK_Mx bit, it is never taken out of service.) The PCIEN_Mx bits are also used to enable or disable the requests from the three external masters during normal operation.

Slave interface

The PCI bus arbiter slave interface supports single 32-bit transfers only.

The system can be configured such that all CSRs can be accessed using only “privileged mode” accesses (HPROT=xx1x) or only user mode accesses (HPROT=xx0x). Use internal register access mode bit 0 in the Miscellaneous System Configuration register to set access accordingly (see "Miscellaneous System Configuration and Status register" on page 296).

The slave generates a AHB bus error if the address is not aligned on a 32-bit boundary and Misaligned Bus Address Response Mode is set in the Miscellaneous System Configuration and Status register. Accesses to non-existent addresses also result in an AHB bus error response.

PCI Arbiter Configuration registers

Table 259 provides the PCI bus arbiter register map.

Address Offset	Register	Description
0xA030 0000	PARBCFG	PCI Arbiter Configuration
0xA030 0004	PARBINT	PCI Arbiter Interrupt Status
0xA030 0008	PARBINTEN	PCI Arbiter Interrupt Enable
0xA030 000C	PMISC	PCI Miscellaneous Support
0xA030 0010	PCFG0	PCI Configuration 0
0xA030 0014	PCFG1	PCI Configuration 1
0xA030 0018	PCFG2	PCI Configuration 2
0xA030 001C	PCFG3	PCI Configuration 3
0xA030 0020	PAHBCFG	PCI Bridge Configuration
0xA030 0024	PAHBERR	PCI Bridge AHB Error Address
0xA030 0028	PCIERR	PCI Bridge PCI Error Address
0xA030 002C	PINTR	PCI Bridge Interrupt Status
0xA030 0030	PINTEN	PCI Bridge Interrupt Enable
0xA030 0034	PALTMEM0	PCI Bridge AHB to PCI Memory Address Translate 0
0xA030 0038	PALTMEM1	PCI Bridge AHB to PCI Memory Address Translate 1
0xA030 003C	PALTIO	PCI Bridge AHB to PCI IO Address Translate
0xA030 0040	PMALT0	PCI Bridge PCI to AHB Memory Address Translate 0
0xA030 0044	PMALT1	PCI Bridge PCI to AHB Memory Address Translate 1
0xA030 0048	PALTCTL	PCI Bridge Address Translation Control
0xA030 004C	CMISC	CardBus Miscellaneous Support
0xA030 004C–0xA030 0FFC		Reserved (all read accesses return 0x0 value)
0xA030 1000	CSKTEV	CardBus Socket Event
0xA030 1004	CSKTMSK	CardBus Socket Mask
0xA030 1008	CSKTPST	CardBus Socket Present State
0xA030 100C	CSKTFEV	CardBus Socket Force Event
0xA030 1010	CSKTCTL	CardBus Socket Control

Table 259: PCI arbiter register map

Address Offset	Register	Description
0xA030 1014–0xA030 1FFC		Reserved (all read accesses return 0x0 value)

Table 259: PCI arbiter register map

PCI Arbiter Configuration register

Address: A030 0000

The PCI Arbiter Configuration register enables and disables each of the three external PCI bus masters. The internal PCI-to-AHB bridge is always enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PCI EN_ M3	PCI EN_ M2	PCI EN_ M1	PCI_ CTL_ RSC_n

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	Read only; hard-wired to 0	Reserved	N/A	N/A
D03	R/W	PCIEN_M3	0	**External master 3 enable** 0 Disable (default) 1 Enable If the master becomes broken, toggle low -> high to re-enable.
D02	R/W	PCIEN_M2	0	**External master 2 enable** 0 Disable (default) 1 Enable If the master becomes broken, toggle low -> high to re-enable.
D01	R/W	PCIEN_M1	0	**External Master 1 Enable** 0 Disable (default) 1 Enable If the master becomes broken, toggle low-> high to re-enable.

Table 260: PCI Arbiter Configuration register

Bits	Access	Mnemonic	Reset	Description
D00	R	PCI_CTL_RSC_n	N/A	**PCI_CENTRAL_RSC_n input to NS9750** (NS9750 has internal pulldown) 0 NS9750 provides PCI central resource functions (pulldown) 1 NS9750 does not provide PCI central resource functions

Table 260: PCI Arbiter Configuration register

PCI Arbiter Interrupt Status register

Address: A030 0004

The PCI Arbiter Interrupt Status register reports broken masters (that is, masters that do not respond in 16 clocks after being granted the bus) and PCI system errors from external PCI agents (that is, SERR# asserted for 1 clock cycle). There is a separate bit for each of the interrupt sources, and each bit can cause an interrupt if the associated bit in the PCI Arbiter Interrupt Enable register is set to 1.

Note: For diagnostics, software can cause an interrupt by writing a 1 to a bit that is set to 0. Otherwise, in normal operation, the software writes a 1 to a bit that is set to clear the bit and the interrupt from the PCI arbiter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CCLK RUN	PCI SERR	PCI BRK_ M3	PCI BRK_ M2	PCI BRK_ M1	PCI BRK_ M0

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:06	Read only; hard-wired to 0	Reserved	N/A	N/A

Table 261: PCI Arbiter Interrupt Status register

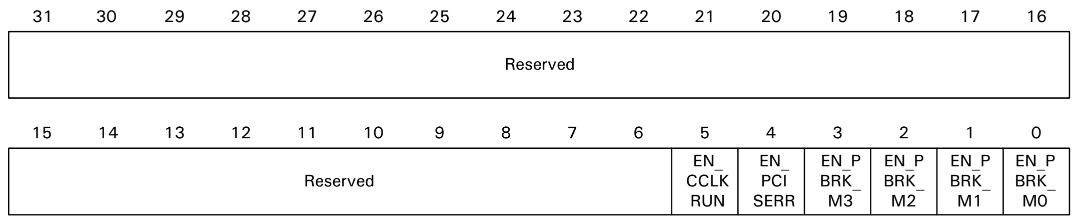
Bits	Access	Mnemonic	Reset	Description
D05	R/C	CCLKRUN	0	**Restart CardBus clock** Used for CardBus Applications only. Indicates that an external CardBus card has asserted CardBus CCLKRUN# to request that the CardBus clock be restarted.
D04	R/C	PCISERR	0	An SERR signal has been received from an external PCI agent.
D03	R/C	PCIBRK_M3	0	External master 3 broken
D02	R/C	PCIBRK_M2	0	External master 2 broken
D01	R/C	PCIBRK_M1	0	External master 1 broken
D00	R/C	PCIBRK_M0	0	PCI-to-AHB bridge broken

Table 261: PCI Arbiter Interrupt Status register

PCI Arbiter Interrupt Enable register

Address: A030 0008

The PCI Arbiter Interrupt Enable register has an enable bit for each of the interrupt status bits in the PCI Arbiter Interrupt Status register. Set these bits to 1 to allow the associated interrupt status bit to cause an interrupt to the system.



Register bit assignment

BIts	Access	Mnemonic	Reset	Description
D31:06	Read only; hard-wired to 0	Reserved	N/A	N/A

Table 262: PCI Arbiter Interrupt Enable register

BIts	Access	Mnemonic	Reset	Description
D05	R/W	EN_CCLKRUN	0	**Enable CCLKRUN# interrupt** 0 Disable (default) 1 Enable
D04	R/W	EN_PCISERR	0	**Enable SERR received from external PCI agent** 0 Disable (default) 1 Enable
D03	R/W	EN_PBRK_M3	0	**Enable external master 3 broken** 0 Disable (default) 1 Enable
D02	R/W	EN_PBRK_M2	0	**Enable external master 2 broken** 0 Disable (default) 1 Enable
D01	R/W	EN_PBRK_M1	0	**Enable external master 1 broken** 0 Disable (default) 1 Enable
D00	R/W	EN_PBRK_M0	0	**Enable PCI-to-AHB bridge broken** 0 Disable (default) 1 Enable

Table 262: PCI Arbiter Interrupt Enable register

PCI Miscellaneous Support register

Address: A030 000C

The PCI Miscellaneous Support register contains miscellaneous PCI functions that are required in NS9750.

- Change the EN_BARx fields only during system initialization, when there is no PCI activity.
- In a system where NS9750 is not the host, the EN_BARx fields must be programmed within 2^{25} PCI clocks of RST# being negated. This is the time allowed from RST# negated to the first configuration cycle on the PCI bus.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						EN_ BAR5	EN_ BAR4	EN_ BAR3	EN_ BAR2	EN_ BAR1	EN_ BAR0	Reserved			INTA2 PCI

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:10	Read only; hard-wired to 0	Reserved	N/A	N/A
D09	R/W	EN_BAR5	0	**Enable bridge PCI Base Address register 5** 0 Disable (default) 1 Enable **Note:** Although BAR_x can still be accessed when EN_BAR5 is 1, the address range defined by BAR_x will not be decoded.
D08	R/W	EN_BAR4	0	**Enable bridge PCI Base Address register 4** 0 Disable (default) 1 Enable **Note:** Although BAR_x can still be accessed when EN_BAR4 is 1, the address range defined by BAR_x will not be decoded.
D07	R/W	EN_BAR3	0	**Enable bridge PCI Base Address register 3** 0 Disable (default) 1 Enable **Note:** Although BAR_x can still be accessed when EN_BAR3 is 1, the address range defined by BAR_x will not be decoded.

Table 263: PCI Miscellaneous Support register

Bits	Access	Mnemonic	Reset	Description
D06	R//W	EN_BAR2	0	**Enable bridge PCI Base Address register 2** 0 Disable (default) 1 Enable **Note:** Although BAR_x can still be accessed when EN_BAR2 is 1, the address range defined by BAR_x will not be decoded.
D05	R/W	EN_BAR1	0	**Enable bridge PCI Base Address register 1** 0 Disable (default) 1 Enable **Note:** Although BAR_x can still be accessed when EN_BAR1 is 1, the address range defined by BAR_x will not be decoded.
D04	R/W	EN_BAR0	0	**Enable bridge PCI Base Address register 0** 0 Disable (default) 1 Enable **Note:** Although BAR_x can still be accessed when EN_BAR0 is 1, the address range defined by BAR_x will not be decoded.
D03:01	Read only; hard-wired to 0	Reserved	N/A	N/A
D00	R/W	INTA2PCI	0	The inverted value of this bit drives the INTA# output pin. INTA# is configured as a pseudo-open drain output. 0 INTA# high impedance (default) INTA# must be in high impedance state for CardBus applications and for PCI applications when NS9750 is the host, and provides the interrupt controller for PCI interrupts. 1 Assert INTA# low

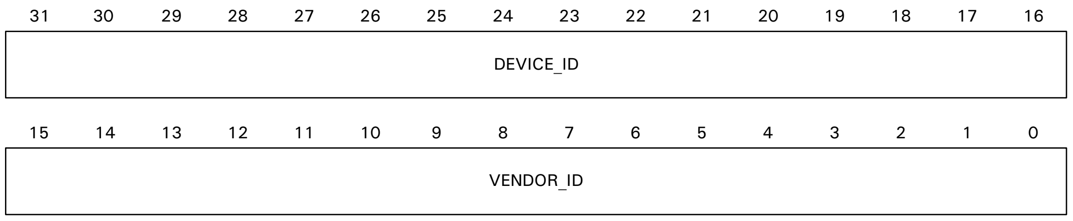
Table 263: PCI Miscellaneous Support register

PCI Configuration 0 register

Address: A030 0010

The PCI Configuration 0 register contains the values that will be read from the PCI Device ID and PCI Vendor ID registers.

Change these fields only during system initialization, when there is no PCI activity. In a system where NS9750 is not the host, these fields must be programmed within 2^{25} PCI clocks of RST# being negated. This is the time allowed from RST# negated to the first configuration cycle on the PCI bus.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	R/W	DEVICE_ID	0x00C4	**Device ID value** Value to be inserted into the PCI Device ID register. Defaults to the assigned device ID (0x00C4).
D15:00	R/W	VENDOR_ID	0x114F	**Vendor ID value** Value to be inserted into the PCI Vendor ID register. Defaults to the assigned vendor ID (0x114F).

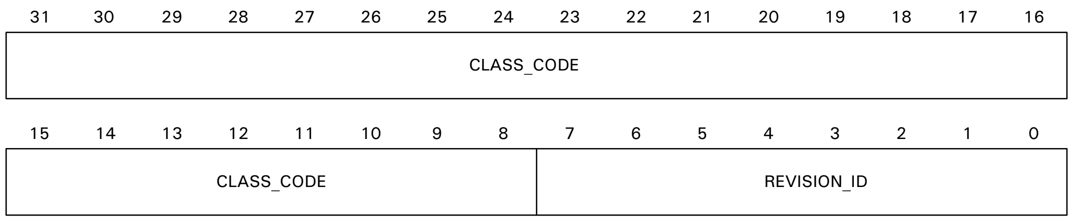
Table 264: PCI Configuration 0 register

PCI Configuration 1 register

Address: A030 0014

The PCI Configuration 1 register contains the values that will be read from the PCI Class Code and PCI Revision ID registers.

Change these fields only during system initialization, when there is no PCI activity. In a system where NS9750 is not the host, these fields must be programmed within 2^{25} PCI clocks of RST# being negated. This is the time allowed from RST# negated to the first configuration cycle on the PCI bus.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	R/W	CLASS_CODE	0x060000	**Class code value** Value to be inserted into PCI Class Code register. Defaults to class code for a host/PCI bridge (0x060000).
D07:00	R/W	REVISION_ID	0x00	**Revision ID value** Value to be inserted into the PCI Revision ID register. Defaults to 0x00.

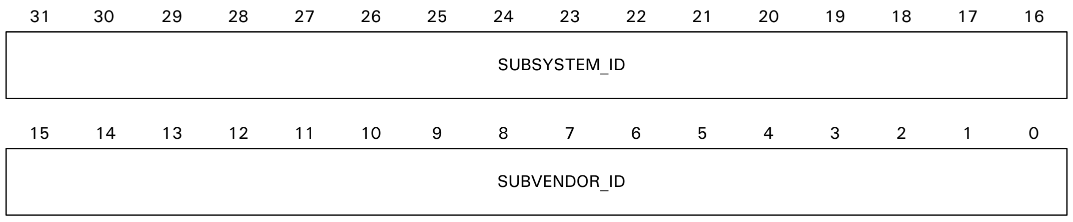
Table 265: PCI Configuration 1 register

PCI Configuration 2 register

Address: A030 0018

The PCI Configuration 2 register contains the values that will be read from the PCI Subsystem ID and PCI Subsystem Vendor ID registers.

Change these fields only during system initialization, when there is no PCI activity. In a system where NS9750 is not the host, these fields must be programmed within 2^{25} PCI clocks of RST# being negated. This is the time allowed from RST# negated to the first configuration cycle on the PCI bus.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	R/W	SUBSYSTEM_ID	0x0000	**Subsystem ID value** Value to be inserted into the PCI Subsystem ID register. Defaults to 0x0000.
D15:00	R/W	SUBVENDOR_ID	0x0000	**Subvendor ID value** Value to be inserted into the PCI Subvendor ID register. Defaults to 0x0000.

Table 266: PCI Configuration 2 register

PCI Configuration 3 register

Address: A030 001C

The PCI Configuration 3 register contains the values that will be read from the PCI Max_Lat, PCI Min_Gnt, and PCI Interrupt Pin registers.

Change these fields only during system initialization, when there is no PCI activity. In a system where NS9750 is not the host, these fields must be programmed within 2^{25} PCI clocks of RST# being negated. This is the time allowed from RST# negated to the first configuration cycle on the PCI bus.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								MAX_LATENCY							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIN_GRANT								INTERRUPT_PIN							

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:24	Read only; hard-wired to 0	Reserved	N/A	N/A

Table 267: PCI Configuration 3 register

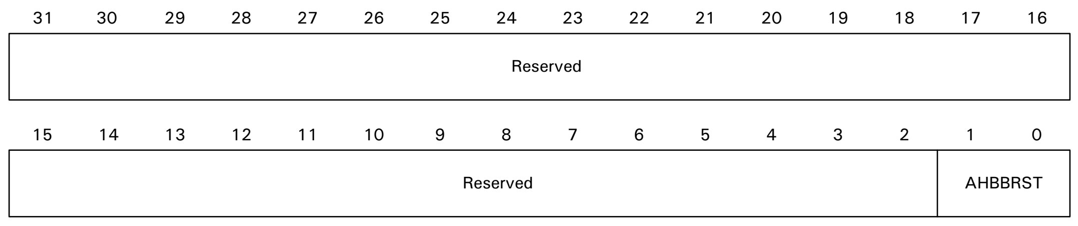
Bits	Access	Mnemonic	Reset	Description
D23:16	R/W	MAX_LATENCY	0x00	**Max latency value** Value to be inserted into the PCI Max_Lat register. Defaults to 0x00.
D15:08	R/W	MIN_GRANT	0x00	**Min grant value** Value to be inserted into the PCI Min_Gnt register. Defaults to 0x00.
D07:00	R/W	INTERRUPT_PIN	0x01	**Interrupt pin value** Value to be inserted onto the PCI Interrupt Pin register. Defaults to 0x01, which is the encoding for INTA#.

Table 267: PCI Configuration 3 register

PCI Bridge Configuration register

Address: A030 0020

The PCI Bridge Configuration register controls the bandwidth allocated to the bridge. Change the AHBBRST field only during system initialization, when there is no traffic to or from the bridge. Because the setting of this register affects NS9750's bandwidth allocation, changes will have an effect on system performance.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:02	Hardwired to 0	Reserved	N/A	N/A

Table 268: PCI Bridge Configuration register

Bits	Access	Mnemonic	Reset	Description
D01:00	R/W	AHBBRST	0x1	**AHB burst length control** Determines the type of burst cycles done when the bridge acts as AHB master: 00 16 01 32 (default) 10 64 11 Reserved

Table 268: PCI Bridge Configuration register

PCI Bridge AHB Error Address register

Address: A030 0024

The PCI Bridge AHB Error Address register stores the address of the AHB transaction that received an AHB ERROR response.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHBEADR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHBEADR															

Register bit assignment

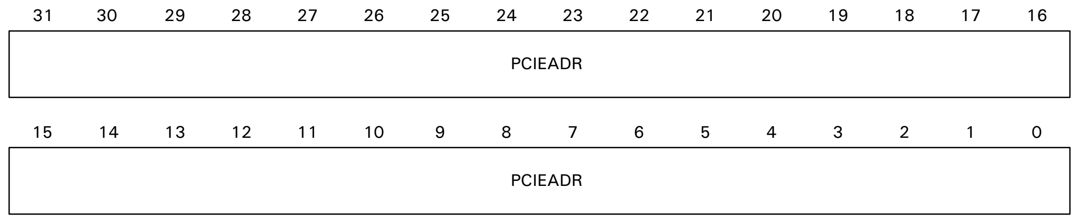
Bits	Access	Mnemonic	Reset	Description
D31:00	R	AHBEADR	0x00000000	**AHB error address** Holds the AHB address that caused the error, when AHBERR is set in the PCI Bridge Interrupt Status register. No further updates are allowed to this register until AHBERR is cleared,

Table 269: PCI Bridge AHB Error Address register

PCI Bridge PCI Error Address register

Address: A030 0028

The PCI Bridge PCI Error Address register stores the address of the PCI transaction that received a PCI bus error response.



Register bit assignment

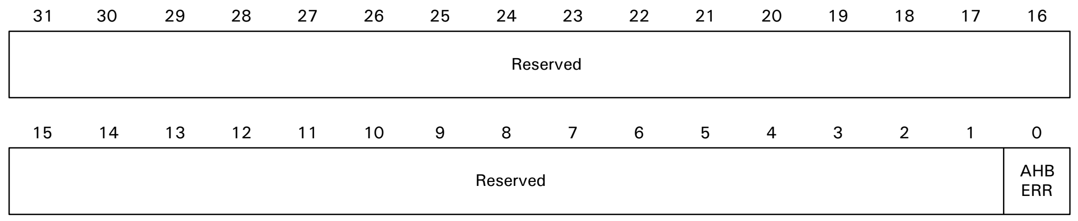
Bits	Access	Mnemonic	Reset	Description
D31:00	R	PCIEADR	0x00000000	**PCI error address** Holds the PCI address that caused an error, when any of the PCI error bits are set in the PCI Status register. No further updates are allowed to this register until all error bits are cleared.

Table 270: PCI Bridge PCI Error Address register

PCI Bridge Interrupt Status register

Address: A030 002C

The PCI Bridge Interrupt Status register shows the status of the AHB bus error interrupt.



Register bit assignment

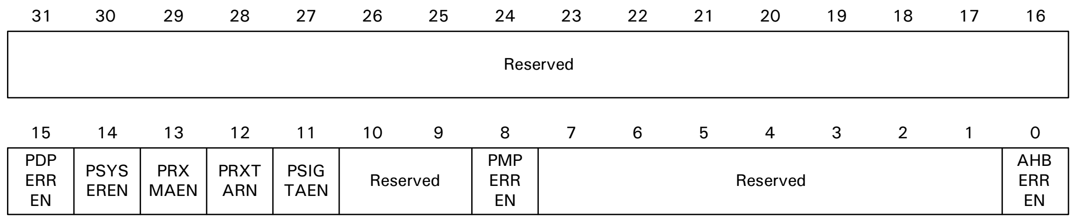
Bits	Access	Mnemonic	Reset	Description
D31:01	Hardwired to 0	Reserved	N/A	N/A
D00	R/C	AHBERR	0	AHB bus error

Table 271: PCI Bridge Interrupt Status register

PCI Bridge Interrupt Enable register

Address: A030 0030

The PCI Bridge Interrupt Enable register stores the enables for all interrupt sources.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	Hardwired to 0	Reserved	N/A	N/A
D15	R/W	PDPERREN	0	**PCI detected parity error enable** 0 Interrupt disabled 1 Interrupt enabled Bit 15 of PCI Status register
D14	R/W	PSYSEREN	0	**PCI signaled system error enable** 0 Interrupt disabled 1 Interrupt enabled Bit 14 of PCI Status register

Table 272: PCI Bridge Interrupt Enable register

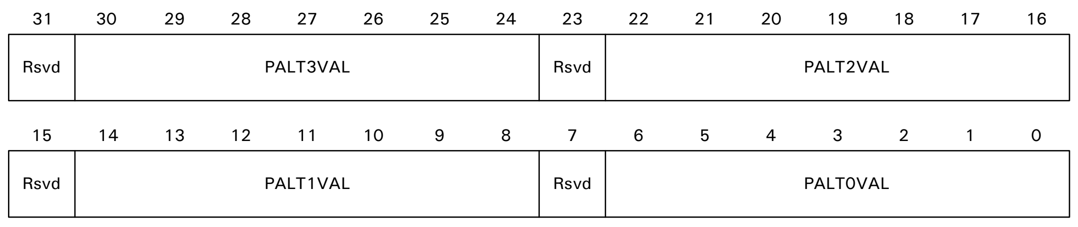
Bits	Access	Mnemonic	Reset	Description
D13	R/W	PRXMAEN	0	**PCI received master abort enable** 0 Interrupt disabled 1 Interrupt enabled Bit 13 of PCI Status register
D12	R/W	PRXTARN	0	**PCI received target abort enable** 0 Interrupt disabled 1 Interrupt enabled Bit 12 of PCI Status register
D11	R/W	PSIGTAEN	0	**PCI signaled target abort enable** 0 Interrupt disabled 1 Interrupt enabled Bit 11 of PCI Status register
D10:09	Hardwired to 0	Reserved	N/A	N/A
D08	R/W	PMPERREN	0	**PCI master data parity error enable** 0 Interrupt disabled 1 Interrupt enabled Bit 8 of PCI Status register
D07:01	Hardwired to 0	Reserved	N/A	N/A
D00	R/W	AHBERREN	0	**AHB bus error enable** 0 Interrupt disabled 1 Interrupt enabled Bit 0 of PCI Bridge Interrupt Status register

Table 272: PCI Bridge Interrupt Enable register

PCI Bridge AHB to PCI Memory Address Translate 0 register

Address: A030 0034

The PCI Bridge AHB-to-PCI Memory Address Translate 0 register translates the AHB addresses sent to the PCI-to-AHB bridge to the appropriate PCI memory addresses.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	Hardwired to 0	Reserved	N/A	N/A
D30:24	R/W	PALT3VAL	0x00	Bits [31:25] of PCI address when AHB address [27:25] = 011.
D23	Hardwired to 0	Reserved	N/A	N/A
D22:16	R/W	PALT2VAL	0x00	Bits [31:25] of PCI address when AHB address [27:25] = 010.
D15	Hardwired to 0	Reserved	N/A	N/A
D14:08	R/W	PALT1VAL	0x00	Bits [31:25] of PCI address when AHB address [27:25] = 001.
D07	Hardwired to 0	Reserved	N/A	N/A
D06:00	R/W	PALT0VAL	0x00	Bits [31:25] of PCI address when AHB address [27:25] - 000.

Table 273: PCI Bridge AHB-to-PCI Memory Address Translate 0 register

PCI Bridge AHB to PCI Memory Address Translate 1 register

Address: A030 0038

The PCI Bridge AHB-to-PCI Memory Address Translate 1 register translates the AHB addresses sent to the PCI-to-AHB bridge to the appropriate PCI memory addresses.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Rsvd	PALT7VAL							Rsvd	PALT6VAL						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd	PALT5VAL							Rsvd	PALT4VAL						

Register bit assignment

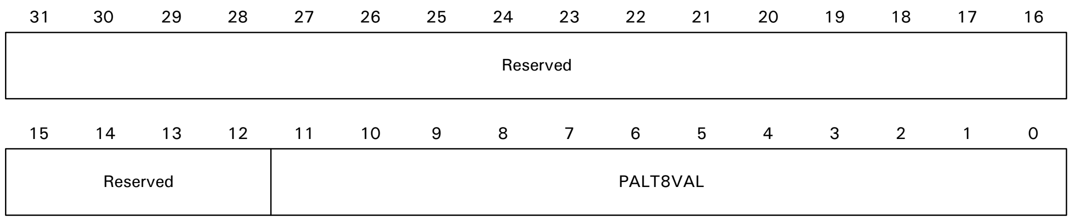
Bits	Access	Mnemonic	Reset	Description
D31	Hardwired to 0	Reserved	N/A	N/A
D30:24	R/W	PALT7VAL	0x00	Bits [31:25] of PCI address when AHB address [27:25] = 111.
D23	Hardwired to 0	Reserved	N/A	N/A
D22:16	R/W	PALT6VAL	0x00	Bits [31:25] of PCI address when AHB address [27:25] = 110.
D15	Hardwired to 0	Reserved	N/A	N/A
D14:08	R/W	PALT5VAL	0x00	Bits [31:25] of PCI address when AHB address [27:25] = 101.
D07	Hardwired to 0	Reserved	N/A	N/A
D06:00	R/W	PALT4VAL	0x00	Bits [31:25] of PCI address when AHB address [27:25] = 100.

Table 274: PCI Bridge AHB-to-PCI Memory Address Translate 1 register

PCI Bridge AHB-to-PCI IO Address Translate register

Address: A030 003C

The PCI Bridge AHB-to-PCI IO Address Translate register translates the AHB addresses sent to the PCI-to-AHB bridge to the appropriate PCI IO addresses.



Register bit assignment

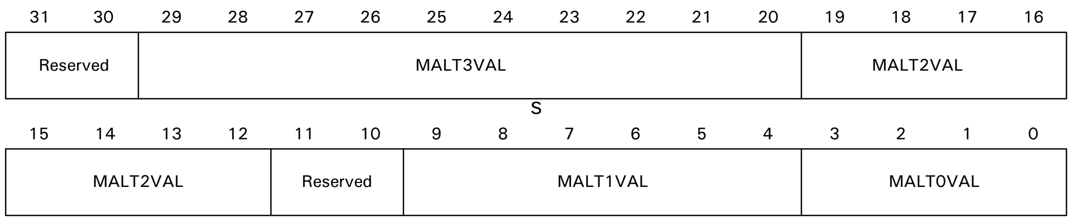
Bits	Access	Mnemonic	Reset	Description
D31:12	Hardwired to 0	Reserved	N/A	N/A
D11:00	R/W	PALT8VAL	0x000	Bits [31:20] of PCI IO address when AHB addresses PCI IO space.

Table 275: PCI Bridge AHB-to-PCI IO Address Translate register

PCI Bridge PCI to AHB Memory Address Translate 0

Address: A030 0040

The PCI Bridge PCI-to-AHB Memory Address Translate 0 register translates the PCI memory addresses to the appropriate AHB memory addresses.



Register bit assignment

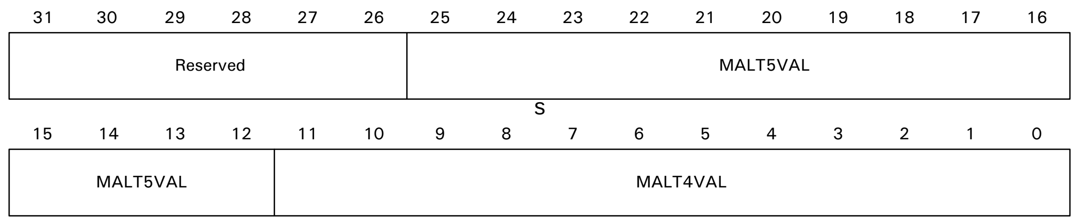
Bits	Access	Mnemonic	Reset	Description
D31:30	Hardwired to 0	Reserved	N/A	N/A
D29:20	R/W	MALT3VAL	0x000	Bits [31:22] of AHB address if PCI address matches BAR3.
D19:12	R/W	MALT2VAL	0x00	Bits [31:24] of AHB address if PCI address matches BAR2.
D11:10	Hardwired to 0	Reserved	N/A	N/A
D09:04	R/W	MALT1VAL	0x00	Bits [31:26] of AHB address if PCI address matches on BAR1.
D03:00	R/W	MALT0VAL	0x0	Bits [31:28] of AHB address if PCI address matches BAR0.

Table 276: PCI Bridge PCI-to-AHB Memory Address Translate 0 register

PCI Bridge PCI to AHB Memory Address Translate 1

Address: A030 0044

The PCI Bridge PCI-to-AHB Memory Translate 1 register translates the PCI memory addresses to the appropriate AHB memory addresses.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:26	Hardwired to 0	Reserved	N/A	N/A

Table 277: PCI Bridge PCI-to-AHB Memory Address Translate 1 register

Bits	Access	Mnemonic	Reset	Description
D25:12	R/W	MALT5VAL	0x0000	Bits [31:18] of AHB address if PCI address matches BAR5.
D11:00	R/W	MALT4VAL	0x000	Bits [31:20] of AHB address if PCI address matches BAR4.

Table 277: PCI Bridge PCI-to-AHB Memory Address Translate 1 register

PCI Bridge Address Translation Control register

Address: A030 0048

The PCI Bridge Address Translation Control register controls the address translation process in both direction (AHB-to-PCI and PCI-to-AHB).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														MALT_EN	PALT_EN

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:02	Hardwired to 0	Reserved	N/A	N/A
D01	R/W	MALT_EN	0	**Enable PCI to AHB address translation** 0 Do not translate PCI addresses. The same addresses are used for both PCI and AHB. 1 Translate PCI addresses per the MALTxxVAL fields in the PCI Bridge PCI-to-AHB Memory Address Translate registers (see page 439 and page 440).

Table 278: PCI Bridge Address Translation Control register

Bits	Access	Mnemonic	Reset	Description
D00	R/W	PALT_EN	0	**Enable AHB-to-PCI address translation for both PCI memory and IO space** 0 Do not translate AHB addresses. The same addresses are used for both PCI and AHB. 1 Translate AHB addresses per the PALTxxVAL fields in the PCI Bridge AHB-to-PCI Memory Address Translate registers (see page 437 and page 438) and the PCI Bridge AHB-to-PCI IO Address Translate register (see page 439).

Table 278: PCI Bridge Address Translation Control register

CardBus Miscellaneous Support register

Address: A030 004C

The CardBus Miscellaneous Support register is used for CardBus applications only, and provides NS9750-specific CardBus control and status. (See the *NS9750 Sample Driver Configurations* for examples of how this register is used.)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMS_ YV_ SKT	CMS_ XV_ SKT	CMS_ V3 SKT	CMS_ V5 SKT	Rsvd	CMS_ YV_ CARD	CMS_ XV CARD	CMS_ V3_ CARD	CMS_ V5 CARD	CMS_ BAD_ VCC	CMS_ DATA_ LOST	CMS_ NOTA_ CARD	CMS_ CB_ CARD	CMS_ CARD_ 16	CMS_ PWR_ CYC	CMS_ CCD2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMS_ CCD1	Reserved				REQ_ INTGT_ EN	REQ_ INT GATE	INTER- RO- GATE	Rsvd	CCLK RUN_ EN	CVS2	CVS1	CCLK_ STOP_ NACK	CCLK_ STOP_ ACK	CCD2	CCD1

Register Bit Assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	CMS_YV_SKT	0	Allows software to control the YV_SKT bit in the CardBus Socket Present State register. When set, indicates that VCC=Y.Y volts can be supplied to the socket.
D30	R/W	CMS_XV_SKT	0	Allows software to control the XV_SKT bit in the CardBus Socket Present State register. When set, indicates that VCC=X.X volts can be supplied to the socket.

Table 279: CardBus Miscellaneous Support register

Bits	Access	Mnemonic	Reset	Description
D29	R/W	CMS_V3_SKT	0	Allows software to control the V3_SKT bit in the CardBus Socket Present State register. When set, indicates that VCC=3.3 volts can be supplied to the socket.
D28	R/W	CMS_V5_SKT	0	Allows software to control the V5_SKT bit in the CardBus Socket Present State register. When set, indicates that VCC=5 volts can be supplied to the socket.
D27	Hard-wired to 0	Reserved	N/A	N/A
D26	R/W	CMS_YV_CARD	0	Allows software to control the YV_CARD bit in the CardBus Socket Present State register. When set, indicates that the card inserted into the socket supports VCC=Y.Y volts.
D25	R/W	CMS_XV_CARD	0	Allows software to control the XV_CARD bit in the CardBus Socket Present State register. When set, indicates that the card inserted into the socket supports VCC=X.X volts.
D24	R/W	CMS_V3_CARD	0	Allows the software to control the V3-CARD bit in the CardBus Socket Present State register. When set, indicates that the card inserted into the socket supports VCC=3.3 volts.
D23	R/W	CMS_V5_CARD	0	Allows software to control the V5_CARD bit in the CardBus Socket Present State register. When set, indicates that the card inserted into the socket supports VCC=5 volts.
D22	R/W	CMS_BAD_VCC_REQ	0	Allows software to control the BAD_VCC_REQ bit in the CardBus Socket Present State register. When set, indicates that there was an attempt to apply an unsupported or incorrect voltage to the socket.

Table 279: CardBus Miscellaneous Support register

Bits	Access	Mnemonic	Reset	Description
D21	R/W	CMS_DATA_LOST	0	Allows software to control the DATA_LOST bit in the CardBus Socket Present State register. When set, indicates that the external card was removed from the socket while the interface was active, and data may have been lost.
D20	R/W	CMS_NOTA_CARD	0	Allows the software to control the NOTA_CARD bit in the CardBus Socket Present State register. When set, indicates that an unsupported card is inserted in the socket.
D19	R/W	CMS_CB_CARD	0	Allows software to control the CB_CARD bit in the CardBus Socket Present State register. When set, indicates that a CardBus card is inserted in the socket.
D18	R/W	CMS_CARD_16	0	Allows software to control the CARD_16 bit in the CardBus Socket Present State register. When set, indicates that a 16-bit PC card is inserted in the socket.
D17	R/W	CMS_PWR_CYC	0	Allows software to control the PWR_CYC bit in the CardBus Socket Present State register. When set, indicates that the socket is powered up. When cleared, indicates that the socket is powered down.
D16	R/W	CMS_CCD2	0	Allows software to control the CCD2 bit in the CardBus Socket Present State register. Reflects the current state of the CardBus CCD#2 pin: 0 A card is inserted in the socket. 1 No card is in the socket. Because CCD2 can be shorted to either CVS2 or CVS1, the value here applies when CVS[2:1] are both 0.

Table 279: CardBus Miscellaneous Support register

Bits	Access	Mnemonic	Reset	Description
D15	R/W	CMS_CCD1	0	Allows the software to control the CCD1 bit in the Cardbus Socket Present State register. Reflects the current state of the CardBus CCD#1 pin: 0 A card is inserted in the socket. 1 No card is in the socket. Because CCD#1 can be shorted to either CVS2 or CVS1, the value here applies when CVS[2:1] are both 0.
D14:11	Hard-wired to 0	Reserved	N/A	N/A
D10	R/W	REQ_INTGT_EN	0	**Enable for REQ_INTGATE interrupt** 0 Disable interrupt (default) 1 Enable interrupt
D09	R/C	REQ_INTGATE	0	**CardBus interrogate socket request** Set to 1 when a 1 is written to the CV_TEST bit in the CardBus Force Event register. This bit causes an interrupt to the CPU when the REQ_INTGT_EN bit (D10 in this register) is set.
D08	R/W	INTERROGATE	0	**Socket interrogation** 0 Socket interrogation not in process 1 Socket interrogation on process Set to 1 during socket interrogation, to prevent changes in CCD#1, CCD#2, and CSTSCHG# from affecting the values in the CardBus Socket Event register.
D07	Hard-wired to 0	Reserved	N/A	N/A
D06	R/W	CCLKRUN_EN	0	**CardBus CCLKRUN# enable** 0 Attempt to negate CardBus CCLKRUN# using CCLKRUN# protocol 1 Assert CardBus CCLKRUN#
D05	R/W	CVS2	0	Value driven out on CVS2 pin during socket interrogation.

Table 279: CardBus Miscellaneous Support register

Bits	Access	Mnemonic	Reset	Description
D04	R/W	CVS1	0	Value driven out on CVS1 pin during socket interrogation.
D03	R	CCLK_STOP_NACK	0	**Response to request to negate CardBus CCLKRUN# using CCLKRUN# protocol** 0 CCLKRUN# request not refused yet. Check CCLK_STOP_ACK to determine whether CCLKRUN# is negated. 1 CCLKRUN# not negated because the external CardBus device will not allow it.
D02	R	CCLK_STOP_ACK	0	**Response to request to negate CardBus CCLKRUN# using CCLKRUN# protocol** 0 CCLKRUN# not negated yet. Check CCLK_STOP_NACK to determine whether CCLKRUN# is not negated because the external CardBus device will not allow it. 1 CCLKRUN# successfully negated
D01	R	CCD2	0	Current state of CCD2 pin.
D00	R	CCD1	0	Current state of CCD1 pin.

Table 279: CardBus Miscellaneous Support register

CardBus Socket Event register

Address: A030 1000

The CardBus Socket Event register is used for CardBus applications only, and indicates a change in socket status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PWR_ CHG	CCD2_ CHG	CCD1_ CHG	CSTS CHG_ CHG

Register bit assignment

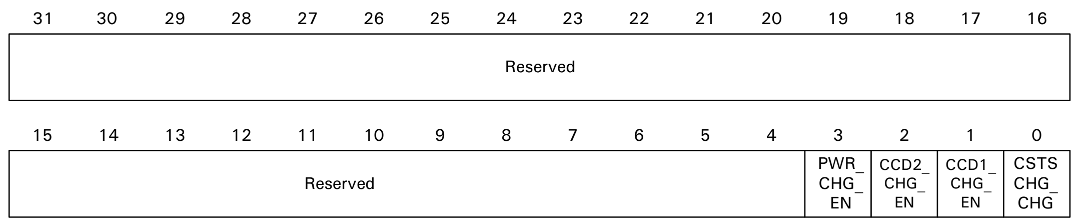
Bits	Access	Mnemonic	Reset	Description
D31:04	Hardwired to 0	Reserved	N/A	N/A
D03	R/C	PWR_CHG	0	Set when the PWR_CYC bit in the CardBus Socket Present State register changes. This bit can also be set by writing a 1 to the FPWR_CHG bit in the CardBus Socket Force Event register.
D02	R/C	CCD2_CHG	0	Set when the CCD#2 signal changes. Changes during card interrogation (when the INTERROGATE bit is set to 1 in the CardBus Miscellaneous Support register) are ignored. This bit can also be set by writing a 1 to the FCCD2_CHG bit in the CardBus Socket Force Event register.
D01	R/C	CCD1_CHG	0	Set when the CCD#1 signal changes. Changes during card interrogation (when the INTERROGATE bit is set to 1 in the CardBus Miscellaneous Support register) are ignored. This bit can also be set by writing a 1 to the FCCD1_CHG bit in the CardBus Socket Force Event register.
D00	R/C	CSTSCHG_CHG	0	Set when the CSTSCHG signal changes from low to high. Changes during card interrogation (when the INTERROGATE bit is set to 1 in the CardBus Miscellaneous Support register) are ignored. This bit can also be set by writing a 1 to the FCSTSCHG_CHG bit in the CardBus Socket Force Event register.

Table 280: CardBus Socket Event register

CardBus Socket Mask register

Address: A030 1004

The CardBus Socket Mask register is used for CardBus applications only, and contains the interrupt enable bits for each of the bits in the CardBus Socket Event register.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	Hardwired to 0	Reserved	N/A	N/A
D03	R/W	PWR_CHG_EN	0	**Power cycle interrupt enable** 0 Interrupt disabled 1 Interrupt enabled
D02	R/W	CCD2_CHG_EN	0	**CCD2 change interrupt enable** 0 Interrupt disabled 1 Interrupt enabled
D01	R/W	CCD1_CHG_EN	0	**CCD1 change interrupt enable** 0 Interrupt disabled 1 Interrupt enabled
D00	R/W	CSTSCHG_CHG_EN	0	**CSTSCHG change interrupt enable** 0 Interrupt disabled 1 Interrupt enabled

Table 281: CardBus Socket Mask register

CardBus Socket Present State register

Address: A030 1008

The CardBus Socket Present State register is used for CardBus applications only, and contains status information about the CardBus socket.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
YV_ SKT	XV_ SKT	V3_ SKT	V5_ SKT	ZV_ SUPP	Reserved										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		YV_ CARD	XV_ CARD	V3_ CARD	V5_ CARD	BAD_ VCC_ REQ	DATA_ LOST	NOTA_ CARD	CINT	CB_ CARD	CARD_ 16	PWR_ CYC	CCD2	CCD1	CSTS CHG

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R	YV_SKT	0	When set, indicates that VCC=Y.Y volts can be supplied to the socket.
D30	R	XV_SKT	0	When set, indicates that VCC=X.X volts can be supplied to the socket.
D29	R	V3_SKT	0	When set, indicates that VCC=3.3 volts can be supplied to the socket.
D28	R	V5_SKT	0	When set, this bit indicates that VCC=5 volts can be supplied to the socket.
D27	Hard-wired to 0	ZV_SUPPORT	0	**Zoomed video support** Always set to 0, as NS9750 does not support this bit.
D26:14	Hard-wired to 0	Reserved	N/A	N/A
D13	R	YV_CARD	0	When set, indicates that the card inserted into the socket supports VCC=Y.Y volts. This bit can also be set by writing a 1 to the FYV_CARD bit in the CardBus Socket Force Event register.
D12	R	XV_CARD	0	When set, indicates that the card inserted into the socket supports VCC=X.X volts. This bit can also be set by writing a 1 to the FXV_CARD in the CardBus Socket Fore Event register.

Table 282: Cardbus Socket Present State register

Bits	Access	Mnemonic	Reset	Description
D11	R	V3_CARD	0	When set, indicates that the card inserted into the socket supports VCC=3.3 volts. This bit can also be set by writing a 1 to the FV3_CARD bit in the CardBus Socket Force Event register.
D10	R	V5_CARD	0	When set, indicates that the card inserted into the socket supports VCC=5 volts. This bit can also be set by writing a 1 to the FV5_CARD bit in the CardBus Socket Force Event register.
D09	R	BAD_VCC_REQ	0	When set, indicates that the software tried to apply an unsupported or incorrect voltage to the socket. This bit can also be set by writing a 1 to the FBAD_VCC_REQ bit in the CardBus Socket Force Event register.
D08	R	DATA_LOST	0	When set, indicates that the external card was removed from the socket while the interface was active, and data may have been lost. This bit can also be set by writing a 1 to the FDATA_LOST bit in the CardBus Socket Force Event register.
D07	R	NOTA_CARD	0	When set, indicates that an unsupported card is inserted in the socket. This bit can also be set by writing a 1 to the FNOTA_CARD bit in the CardBus Socket Force Event register when a card is not in the socket.
D06	R	CINT	0	**Inverted current state of the CardBus CINT# pin** 0 CINT# negated 1 CINT# asserted
D05	R	CB_CARD	0	When set, indicates that a CardBus card is inserted in the socket. This bit can also be set by writing a 1 to the FCB_CARD bit in the CardBus Socket Force Event register when a card is not inserted in the socket.

Table 282: Cardbus Socket Present State register

Bits	Access	Mnemonic	Reset	Description
D04	R	CARD_16	0	When set, indicates that a 16-bit PC card is inserted in the socket. This bit can also be set by writing a 1 to the FCARD_16 bit in the CardBus Socket Force Event register when a card is not inserted in the socket.
D03	R	PWR_CYC	0	When set, indicates that the socket is powered up. When cleared, this bit indicates that the socket is powered down.
D02	R	CCD2	0	**Current state of the CardBus CCD#2 pin** 0 A card is inserted in the socket. 1 No card is in the socket. Because CCD2 can be shorted to either CVS2 or CVS1, the value here applies when CVS[2:1] are both 0.
D01	R	CCD1	0	**Current state of the CardBus CCD#1 pin** 0 A card is inserted in the socket. 1 No card is in the socket. Because CCD1 can be shorted to either CVS2 or CVS1, the value here applies when CVS[2:1] are both 0.
D00	R	CSTSCHG	0	**Current state of the CardBus CSTSCHG pin** 0 CSTSCHG negated 1 CSTSCHG asserted

Table 282: Cardbus Socket Present State register

CardBus Socket Force Event register

Address: A030 100C

The CardBus Socket Force Event register is used for CardBus applications only. This register is implemented only as an address that is written to force various status and event bits in the CardBus host bridge through software. Writing a 1 to a bit in this register sets a corresponding bit in the CardBus Socket Event register or CardBus Socket Present State register.

Note: The CardBus Socket Force Event register sets selected bits that are in the CardBus Socket Present State register. Clear these bits by clearing the corresponding bits in the CardBus Miscellaneous Support register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd	CV_ TEST	FYV_ CARD	FXV_ CARD	FV3_ CARD	FV5_ CARD	FBAD_ VCC_ REQ	F DATA_ LOST	**F NOTA_ CARD**	Rsvd	FCB_ CARD	F CARD_ 16	FPWR_ CHG	F CCD2_ CHG	F CCD1_ CHG	FCTS SCHG_ CHG

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:15	N/A	Reserved	N/A	N/A
D14	W	CV_TEST	N/A	■ Requests that the card interrogation procedure be run again. ■ Sets the REQ_INGATE bit in the CardBus Miscellaneous Support register to 1. ■ If the REQ_INGATE_EN bit in the CardBus Miscellaneous Support register is set, writing a 1 to this bit causes an interrupt back to the CPU.
D13	W	FYV_CARD	N/A	Sets the YV_CARD bit in the CardBus Socket Present State register.
D12	W	FXV_CARD	N/A	Sets the XV_CARD bit in the CardBus Socket Present State register.
D11	W	FV3_CARD	N/A	Sets the V3_CARD bit in the CardBus Socket Present State register.
D10	W	FV5_CARD	N/A	Sets the V5_CARD bit in the CardBus Socket Present State register.
D09	W	FBAD_VCC_REQ	N/A	Sets the BAD_VCC_REQ bit in the CardBus Socket Present State register.
D08	W	FDATA_LOST	N/A	Sets the DATA_LOST bit in the CardBus Socket Present State register.
D07	W	FNOTA_CARD	N/A	Sets the NOTA_CARD bit in the CardBus Socket Present State register. If a card is in the socket (that is, CMISC_CCD[1:0]=00 in the CardBus Miscellaneous Support register), writes to this bit are ignored.

Table 283: CardBus Socket Force Event register

Bits	Access	Mnemonic	Reset	Description
D06	N/A	Reserved	N/A	N/A
D05	W	FCB_CARD	N/A	Sets the CB_CARD bit in the CardBus Socket Present State register. If a card is in the socket (that is, CMISC_CCD[1:0]=00 in the CardBus Miscellaneous Support register), writes to this bit are ignored.
D04	W	FCARD_16	N/A	Sets the CARD_16 bit in the CardBus Socket Present State register. If a card is in the socket (that is, CMISC_CCD[1:0]=00 in the CardBus Miscellaneous Support register), writes to this bit are ignored.
D03	W	FPWR_CHG	N/A	Sets the PWR_CHG bit in the CardBus Socket Event register. This does not affect the PWR_CYC bit in the CardBus Socket Present State register.
D02	W	FCCD2_CHG	N/A	Sets the CCD2_CHG bit in the CardBus Socket Event register. This does not affect the CCD2 bit in the CardBus Socket Present State register.
D01	W	FCCD1_CHG	N/A	Sets the CCD1_CHG bit in the CardBus Socket Event register. This does not affect the CCD1 bit in the CardBus Socket Present State register.
D00	W	FCSTSCHG_CHG	N/A	Sets the CSTSCHG_CHG bit in the CardBus Socket Event register, This does not affect the CSTSCHG bit in the CardBus Socket Present State register.

Table 283: CardBus Socket Force Event register

CardBus Socket Control register

Address: A030 1010

The CardBus Socket Control register is used only for CardBus applications.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ZV_ ACTI-VITY	STDZ VREG	ZVEN	Rsvd	STOP_ CLK	VCC_CTL			Rsvd	VPP_CTL		

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:12	Hardwired to 0	Reserved	N/A	N/A
D11	Hardwired to 0	ZV_ACTIVITY	0	**Zoomed video activity bit** Hardwired to 0.
D10	Hardwired to 0	STANDARDZVREG	0	**Standardized zoomed video register model support** Hardwired to 0.
D09	R/W	ZVEN	0	**Zoomed video enable** Defaults to 0
D08	Hardwired to 0	Reserved	N/A	N/A
D07	R/W	STOP_CLK	0	**Stop CardBus clock** Defaults to 0.
D06:04	R/W	VCC_CTL	000	**Socket VCC control** 000 0 V 001 Reserved 010 5 V 011 3.3 V 1xx Reserved
D03	Hardwired to 0	Reserved	N/A	N/A

Table 284: CardBus Socket Control register

Bits	Access	Mnemonic	Reset	Description
D02:00	R/W	VPP_CTL	000	**Socket VPP/Core control** 000 0 V 001 12 V 010 5 V 011 3.3 V 100 Reserved 101 Reserved 110 1.8 V 111 Reserved

Table 284: CardBus Socket Control register

PCI system configurations

NS9750 can be connected to the PCI bus using an embedded (internal) or external PCI bus arbiter. Figure 72 shows how NS9750 is connected to the PCI bus for a typical system application using the embedded PCI bus arbiter, and where the NS9750 provides the central resource function (see "PCI central resource functions" on page 458). Up to three external masters are supported.

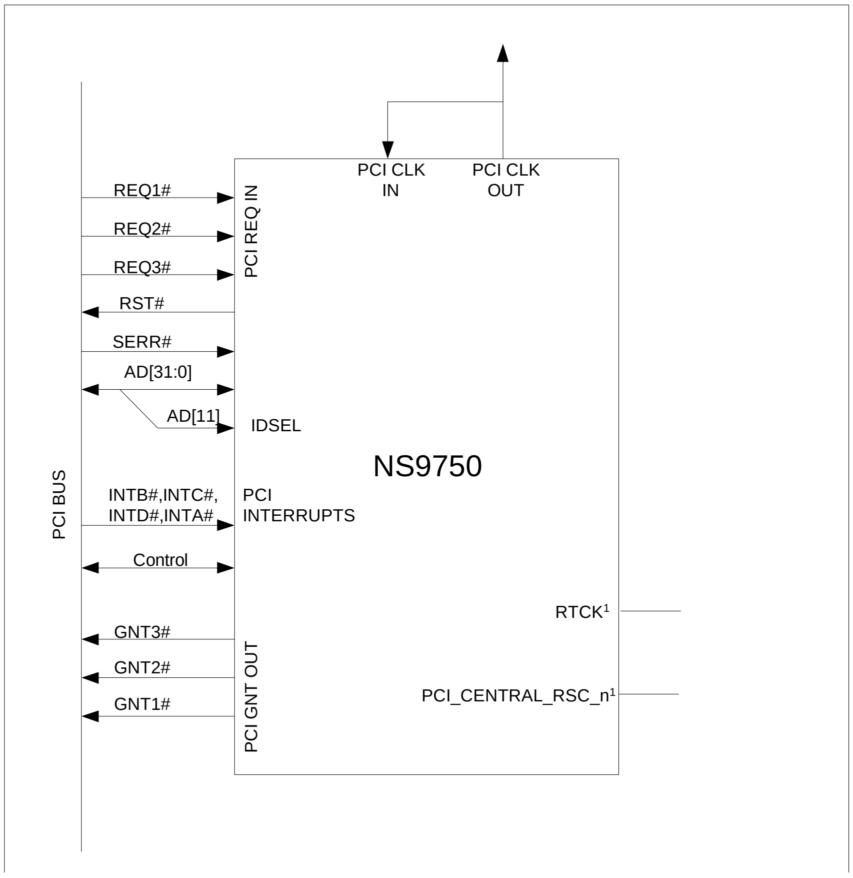


Figure 72: System connections to NS9750 — Internal arbiter and central resources

Note:

1 These pins are not connected because internal resistors tie these pins to the appropriate state.

The internal PCI arbiter is selected when the RTCK pin is set to 1 during powerup. Because the RTCK pad has a weak internal pullup, no external components are required to select the internal PCI arbiter.

Figure 73 shows how NS9750 is connected to the PCI bus for a typical application using an external PCI bus arbiter, and where the NS9750 does not provide the PCI central resource functions (see "PCI central resource functions" on page 458). RTCK is pulled low by an external resistor to configure NS9750 to use an external arbiter. The REQ0#/GNT0# pair for the external PCI arbiter is multiplexed out on the GNT1#/ REQ1# pins — GNT1# to REQ0# and REQ1# to GNT0#.

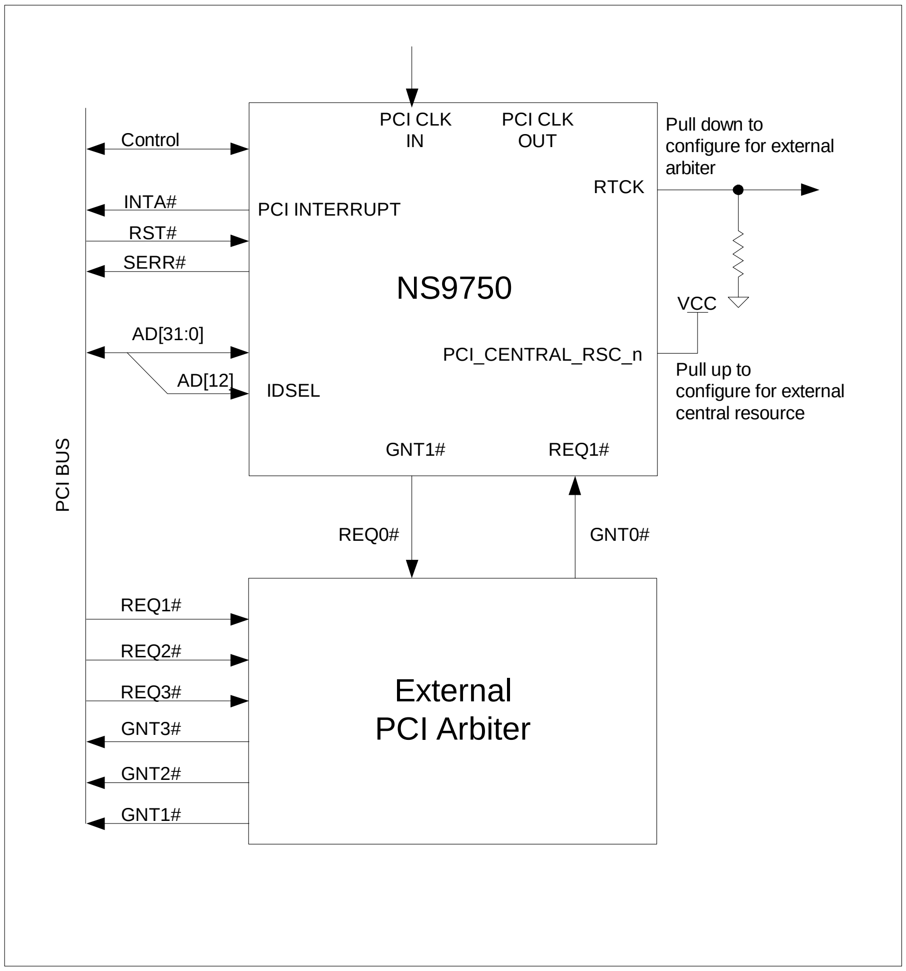


Figure 73: System connections to NS9750 — External arbiter and central resources

Device selection for configuration

The NS9750 IDSEL pin is used as a chip select during PCI configuration transactions. If the bridge's configuration registers are being programmed using the AHB bus, NS9750 must be set as Device 0 (see Figure 72, "System connections to NS9750 — Internal arbiter and central resources," on page 456, which shows IDSEL connected to AD[11] and which configures NS9750 as PCI Device 0).

If the bridge's configuration registers are programmed using an external PCI device, NS9750 can be configured as any PCI device number (see Figure 73, "System connections to NS9750 — External arbiter and central resources," on page 457, which shows IDSEL connected to AD[12] and which configured NS9750 as Device 1). Any accesses from the AHB bus to the bridge's configuration registers, however, must be done as Device 0.

PCI interrupts

NS9750 can serve as the interrupt controller for all four PCI interrupts. These interrupts go directly to the interrupt controller in the System Control module. Because these interrupts are open-drain type signals, the transition from low to high is relatively slow once the source of the interrupt is cleared. SERR# input can also cause an interrupt through PCISERR (in the PCI Arbiter Interrupt Status register).

Important: The system software must provide adequate time for the interrupt signals and SERR# to rise before re-enabling them.

If an external interrupt controller is used, NS9750 can drive INTA# using INTA2PCI (in the PCI Miscellaneous Support register).

PCI central resource functions

NS9750 provides several PCI central resource functions when NS9750's PCI_CENTRAL_RSC_n pin is pulled low (see Figure 72):

- RST# to the PCI system is driven through NS9750. RST# is asserted asynchronously and negated synchronously to the PCI clock. RST# is driven from the system reset to NS9750 and the PCI bit in the Reset and Sleep Control register in the System Control Module.
- SERR# is configured as an input.

- AD[31:0], C/BE[3:0], and PAR are driven low when RST# is asserted, to keep the signals from floating.

When the PCI_CENTRAL_RSC_n pin is pulled high (see Figure 73), these functions operate differently:

- RST# is configured as an input, and must be supplied by the system. In this situation, RST# is used as another system reset to NS9750; that is, either reset_n or RST# can reset NS9750, and both must be negated to take NS9750 out of reset.
- SERR# is configured as output.
- AD[31:0], C/BE[3:0], and PAR are tri-stated when RST# is asserted.

The system must provide pullup resistors on the following signals, regardless of the state of PCI_CENTRAL_RSC_n:

- FRAME#
- TRDY#
- IRDY#
- DEVSEL#
- STOP#
- SERR#
- PERR#
- LOCK#

  Note: The NS9750 does not have a LOCK# pin associated with it. If any PCI device in the system uses the LOCK# signal, the signal must have a pullup resistor.
- INTA#
- INTB#
- INTC#
- INTD#
- All REQ# inputs to NS9750
- All GNT# outputs from NS9750 that are connected to other PCI devices in the system (because they are tri-stated during RST#)

The PCI clock can be either generated from NS9750 (see Figure 72, "System connections to NS9750 — Internal arbiter and central resources," on page 456) or provided by an external source (see Figure 73, "System connections to NS9750 — External arbiter and central resources," on page 457). The PCI CLK input pad has a weak internal pullup.

Important: Note that in cases where NS9750 provides the PCI clock, the PCI clock connection to the NS9750 must still be made external to the NS9750, as shown in Figure 73 (that is, connect PCI_clk_out to PCI_clk_in). This is done to minimize the clock skew between the NS9750 and external PCI devices.

CardBus Support

NS9750 can support 32-bit CardBus applications using the existing PCI port and existing PCI-to-AHB bridge IP.

Figure 74 shows how NS9750 is configured for CardBus applications. The CardBus model has only one external card connected to a host adapter or PCI-to-CardBus bridge through a socket. All connections are point-to-point, and both the host adapter and the external card can be masters on the bus.

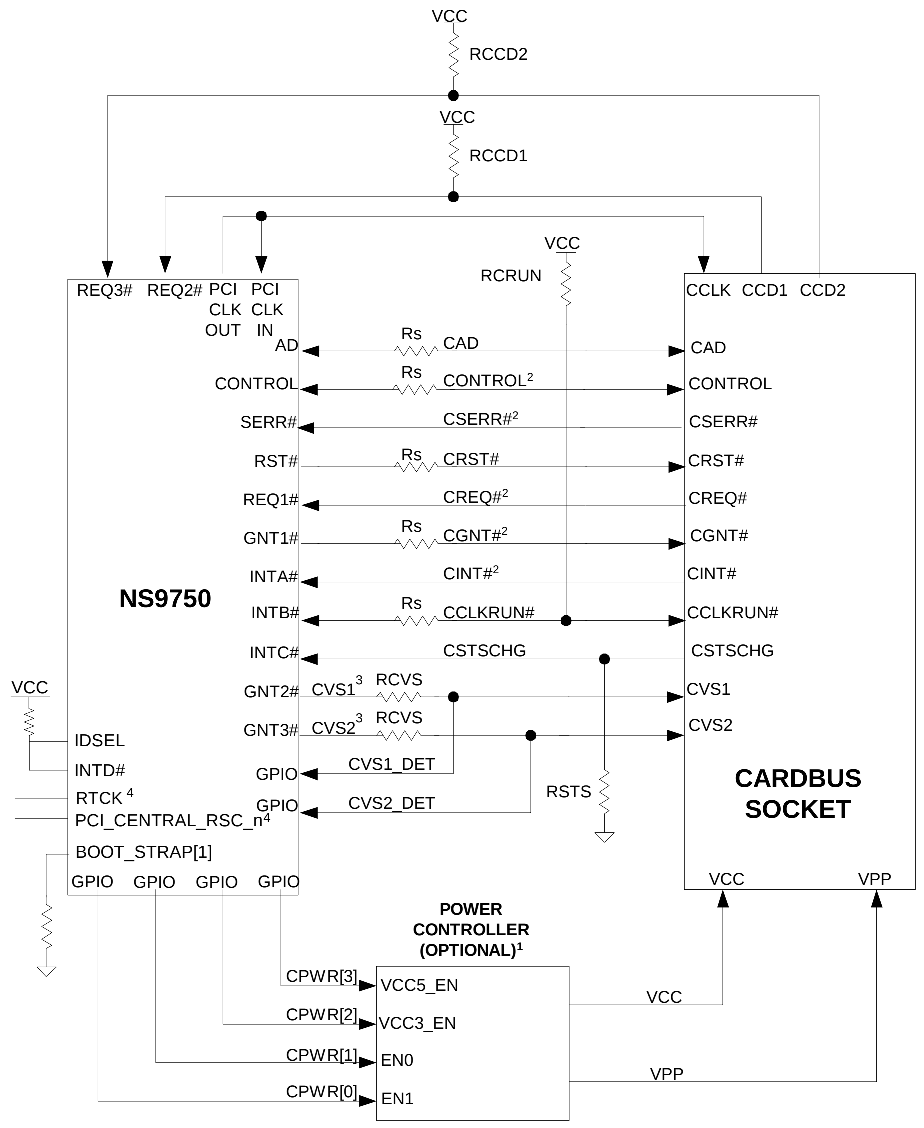


Figure 74: CardBus system connections to NS9750

Notes:

1 The power controller is required only for applications that support hot-insertion and hot-removal of the CardBus card. This requires additional components to isolate NS9750 from CardBus.

2 The system must provide external pullup per PCI specification. CAD, C/BE, and PAR do not require pullups.

3 Voltage detection signal optional for embedded system.

4 Pins not connected because internal resistors tie these to the appropriate state.

Configuring NS9750 for CardBus support

Although many CardBus signals are the same as those for the PCI bus, there are some unique signals. Table 285 lists the new signals and indicates the PCI signals with which they are multiplexed for NS9750.

PCI Signal	CardBus Signal	CardBus type	Comments
INTA#	CINT#	Input	Cardbus interrupt pin. INTA2PCI in the PCI Miscellaneous Support register must be 0.
INTB#	CCLKRUN#	Bidir	CardBus pin used to negotiate with the external CardBus device before stopping the clock. Also allows external CardBus device to request that the clock be restarted.
INTC#	CSTSCHG	Input	CardBus status change interrupt signal.
GNT1#	CGNT#	Output	Grant to external CardBus device from NS9750's internal arbiter.
GNT2#	CVS1	Output	Voltage sense pin. Normally driven low by NS9750, but toggled during interrogation of external CardBus device to detect voltage requirements.
GNT3#	CVS2	Output	Voltage sense pin. Normally driven low by NS9750 but toggled during interrogation of external CardBus device to detect voltage requirements.
REQ1#	CREQ#	Input	Request from external CardBus device to NS9750's internal arbiter.

Table 285: CardBus IO muxing

PCI Signal	CardBus Signal	CardBus type	Comments
REQ2#	CCD1	Input	Card detect pin. Pulled up by system when socket is empty and pulled low when the external CardBus device is present in the socket.
REQ3#	CCD2	Input	Card detect pin. Pulled up by system when socket is empty and pulled low when the external CardBus device is present in the socket.
N/A	CVS1_DET	Input	Voltage sense detect pin. Can be any GPIO input. Used to detect whether the external CardBus device shorts CVS1 to ground.
N/A	CVS2_DET	Input	Voltage sense detect pin. Can be any GPIO input. Used to detect whether the external device shorts CVS2 to ground.
N/A	CPWR[3:0]	Output	Controls to the external power controller that may be providing the power to the CardBus socket. Required only for hot-insertion and hot-removal. Can be any GPIO outputs.

Table 285: CardBus IO muxing

Notes:

- BOOTSTRAP[1] and PCI_CENTRAL_RSC_n are two strapping pins that must be pulled low to configure the IO and internal arbiter for CardBus. Because CardBus does not have an IDSEL signal, it is pulled high. As a result, NS9750 captures any configuration accesses from the external CardBus device since it is the only other device on the bus.
- The internal arbiter must be used for CardBus applications. Requests to the arbiter from the REQ2# and REQ3# pins, however, must be disabled by clearing the PCIEN_M2 and PCIEN_M3 bits in the PCI Arbiter Configuration register. The arbiter parks the bus only on the bridge — not on the last granted master, as is done for PC I.

CardBus adapter requirements

In a CardBus application, NS9750 is the *adapter*, or *bridge*. The adapter is required to have a set of socket registers that provide socket control and status. The following NS9750 registers support this requirement:

- CardBus Socket Event (see "CardBus Socket Event register" on page 446)
- CardBus Socket Mask (see "CardBus Socket Mask register" on page 447)
- CardBus Socket Present State (see "CardBus Socket Present State register" on page 448)
- CardBus Socket Force Event (see "CardBus Socket Force Event register" on page 451)
- CardBus Socket Control (see "CardBus Socket Control register" on page 454)

CardBus interrupts

The dedicated CINT# signal on the CardBus is connected directly to the interrupt controller in the System Control module, as Interrupt #10 (that is, the same as the INTA# signal for the PCI bus). Table 286 shows the CardBus-related maskable interrupt conditions that are reported to the System Control module's interrupt controller through the PCI arbiter's interrupt.

Bit field	Register
CSTSCHG_CHG	CardBus Socket Event
CCD1_CHG	CardBus Socket Event
CCD2_CHG	CardBus Socket Event
PWR_CHG	CardBus Socket Event
REQ_INTGATE	CardBus Miscellaneous Support
CCLKRUN	PCI Arbiter Interrupt Status

Table 286: CardBus interrupt sources

BBus Bridge

C H A P T E R 8

The NS9750 ASIC contains two busses that interconnect the peripherals. The high speed peripherals reside on the AMBA AHB bus. The low speed peripherals reside on the Digi proprietary BBus. The main function of the BBus bridge is to connect the main AHB bus to the proprietary Digi BBus. Both bus interfaces have a master and a slave interface.

BBus bridge functions

The Digi BBus is a low-speed secondary bus that operates at half the AHB clock frequency. The BBus interface houses the slower serial interfaces for USB, IEEE 1284, SPI, and UART, as well as dedicated BBus DMA control, to offload some of the bandwidth demands of the primary AHB bus. The BBus bridge controls the flow of data between the AHB and BBus interfaces.

The BBus bridge provides these functions:

- **BBus arbitration and multiplexing.** The USB and DMA peripheral, in addition to the BBus bridge, can be BBus masters. All BBus peripherals contain a slave interface.
- **Two-channel DMA controller.** The DMA controller performs memory-to-memory transfers across the AHB bus, allowing DMA transfers from an external peripheral to external memory or from external memory to an external peripheral.
- **System boot engine that fetches data from an external SPI-EEPROM and writes it to an external SDRAM.** The boot engine configures the memory controller accordingly, before fetching the contents of the EEPROM. While a serial boot operation takes place, the CPU is held in reset.

Figure 75 shows the four functions of the BBus bridge in relation to the AHB bus and the BBus.

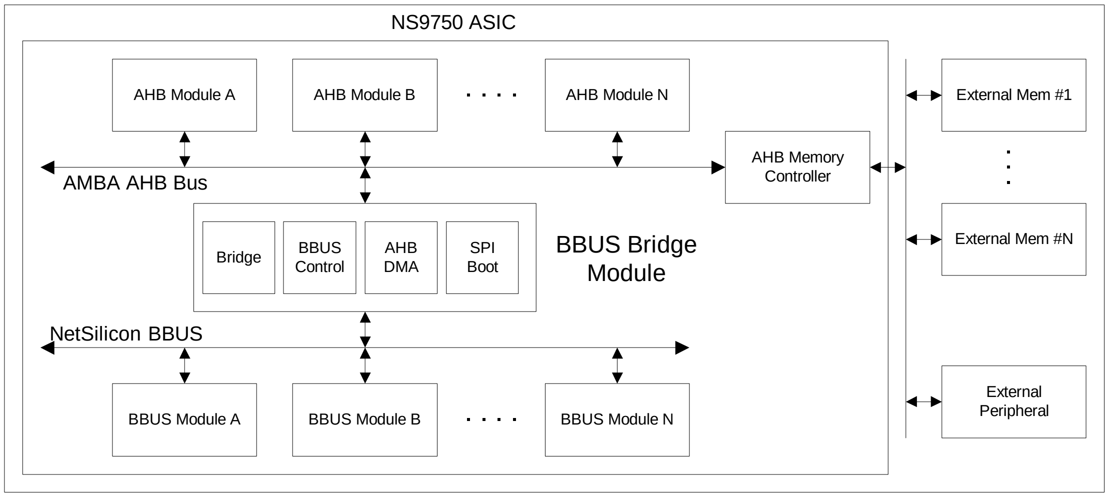


Figure 75: Basic block diagram

Bridge control logic

BBus bridge control logic translates the AHB bus protocol to the BBus protocol and vice versa. The AHB bus can operate at a maximum of 100MHz; the BBus operates at half the AHB clock frequency.

Figure 76 details BBus bridge control logic.

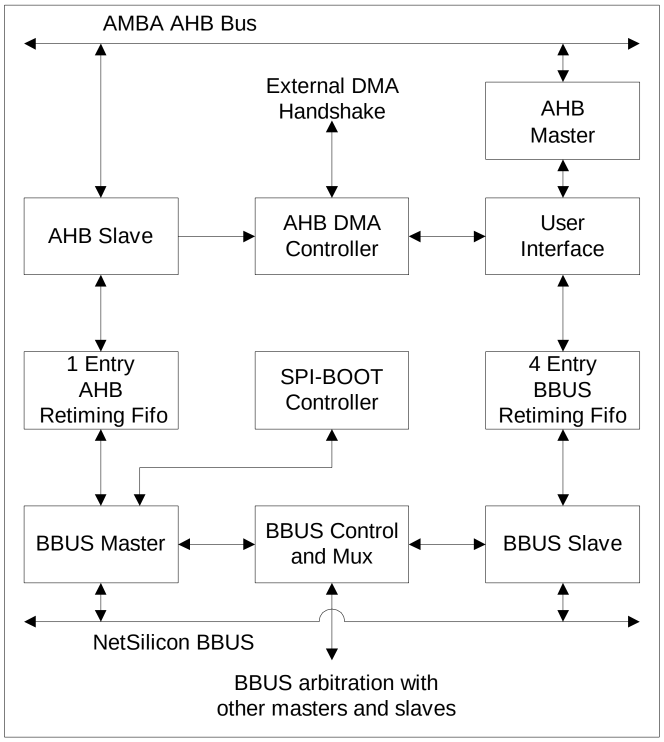


Figure 76: BBus bridge block diagram

Notes:

- The AHB bus and BBus clock domains are asynchronous to each other.
- A 4-entry bidirectional FIFO is implemented in the BBus-to-AHB data path to allow burst transfers.
- The FIFO size matches the maximum burst size supported by the BBus masters.
- The AHB-to-BBus data path does not support burst transfers, allowing only a single entry bidirectional FIFO to be implemented in the data path. FIFO size accommodates only one transaction at a time.

DMA accesses

There are two DMA controllers on the NS9750 BBus. One DMA controller services all BBus peripherals except the USB device; the other is dedicated to the USB device. Each DMA controller contains 16 channels that perform both DMA read and DMA write transactions.

Note: The USB host is a bus mastering BBus peripheral.

- **DMA memory-to-peripheral transfers (DMA read).** DMA read transactions begin with the DMA controller arbitrating for BBus control. When the bus has been granted, the read transaction is presented to the BBus slave interface within the BBus bridge. The command then is passed into the BBus command retiming FIFO, where the user interface picks it up and passes it to the AHB master interface. The AHB master arbitrates for the AHB bus, performs the specified AHB read transaction, and returns the data to the BBus retiming data FIFO. When the BBus slave detects the data in the retiming data FIFO, the BBus slave can respond to the read request from the BBus master.

  The AMBA AHB bus can indicate the burst size at the beginning of a new transfer; the AHB master sets the hburst[2:0] signals to the appropriate value. Because the BBus cannot indicate burst size, the user interface always issues a 4-transfer (4-word) request, which goes into the BBus retiming FIFO. When data is transferred to the BBus, as many words as are needed are moved. When the BBus read transaction completes, any words remaining in the retiming FIFO are flushed.

- **DMA peripheral-to-memory transfers (DMA write).** DMA write transactions begin with the DMA controller arbitrating for control of the BBus. Once the bus is granted, the write transaction is presented to the BBus slave interface within the BBus bridge. The BBus slave interface passes the command data to the BBus data retiming FIFO, but retains the command until the BBus transaction completes. At this point, the BBus slave knows the size of the burst (by counting the number of transfers); that information is passed with the command to the BBus retiming command FIFO. When the BBus detects the presence of the command, it passes the command to the AHB master. The AHB master arbitrates for the AHB bus and performs the AHB write transaction.

BBus control logic

BBus control logic consists of a round-robin arbiter to select a new master, the multiplexing logic to provide the new master's signals to the BBus slaves, and address decoding to select the target BBus slave.

BBus bridge masters and slaves

BBus bridge arbitration allows each bus master to control the bus in a round-robin manner. If a bus master does not require the bus resources when its turn comes around, that bus master is skipped until the next round-robin slot. Each potential bus master presents the bus with request and attribute signals. Once the bus grants mastership, the targeted device is selected.

Note: The CPU always is granted mastership when requested, because its transactions are time-sensitive and completed within 4 BBus clock cycles. When the CPU requests use of the bus, it must wait until the current transaction finishes. The CPU then takes mastership and performs its transaction, before the next BBus master with a pending request. When the CPU transaction is finished, the bus grants mastership to the appropriate requesting BBus master.

Table 287 shows the BBus bus master and slave modules.

Module	Master	Slave
BBus bridge	Y	Y
BBus DMA	Y	Y
SER		Y
I2C		Y
1284		Y
USB dev		Y
USB DMA	Y	Y
USB host	Y	Y

Table 287: BBus master and slave modules

Cycles and BBus arbitration

During a normal cycle, each bus master cycle is allowed only one read/write cycle if another bus master is waiting. There are two exceptions to this rule: burst transactions and read-modify-write transactions.

In a burst transaction, the master can perform more than one read or write cycle. In a read-modify-write transaction, the bus master performs one read and write cycle to the same location.

BBus peripheral address map (decoding)

The BBus address map is divided to allow access to the internal modules and external resources routed through the internal peripherals. The BBus configuration registers are located at base address 0xA040 0000 and are dedicated a 1 MB address space. The BBus peripherals are located at base address 0x9000 0000 and span a 256 MB address space. Each BBus peripheral, with the exception of the SER port controllers, resides in a separate 1 MB address space.

Table 288 specifies the address space given to each peripheral.

Base address	Peripheral
0x9000 0000	BBus DMA controller
0x9010 0000	USB controller
0x9020 0000	SER Port #B
0x9020 0040	SER Port #A
0x9030 0000	SER Port #C
0x9030 0040	SER Port #D
0x9040 0000	IEEE-1284 controller
0x9050 0000	I2C controller
0x9060 0000	BBus utility

Table 288: BBus peripheral address map

Two-channel AHB DMA controller (AHB bus)

Each DMA channel moves data from the source address to the destination address. Transfers can be specified as burst-oriented to maximize AHB bus efficiency. All transfers are executed in two steps:

1. Data is moved from the source address to an 8-entry buffer in the DMA control logic.
2. Data is moved from the 8-entry buffer to the destination address.

These steps are repeated until the DMA transfer is complete. Note that optimum performance is achieved when the source and destination addresses are word-aligned.

Initiating a DMA transfer

There are two ways to initiate a DMA transfer: processor-initiated and external-peripheral initiated.

When the processor initiates the DMA transfer, it performs these steps:

1. Sets up the required buffer descriptors.
2. Configures the appropriate DMA Channel 1/2 Control register (see "DMA Channel 1/2 Control register" on page 491).
3. Writes a 1 to the channel enable (CE) and channel go (CG) fields in the DMA Channel 1/2 Control register (see "DMA Channel 1/2 Control register" on page 491).

The external peripheral initiates a DMA transfer by asserting the appropriate REQ signal. Software must set up the required buffer descriptors and configure the DMA Channel 1/2 Control register (including setting the CE field to 1) before asserting the REQ signal.

DMA buffer descriptor

All DMA channels in NS9750 use a buffer descriptor. When a DMA channel is activated, it reads the DMA buffer descriptor pointed to by the Buffer Descriptor Pointer register (see "Buffer Descriptor Pointer register" on page 491). A DMA buffer descriptor is always fetched using an AHB INCR4 transaction, to maximize AHB bus bandwidth.

When the current descriptor is retired, the next descriptor is accessed from a circular buffer.

Each DMA buffer descriptor requires four 32-bit words to describe a transfer. Circular buffers of 1024 bytes contain multiple buffer descriptors. The first buffer descriptor address is provided by the DMA channel's Buffer Descriptor Pointer register. Subsequent buffer descriptors are found adjacent to the first descriptor. The final buffer descriptor is defined with its W bit set. When the DMA channel encounters the W bit, the channel wraps around to the first descriptor.

Each DMA channel can address a maximum of 64 buffer descriptors, each consisting of 16 bytes. Configuring the DMA channel for more than the maximum number of buffer descriptors results in unpredictable behavior.

Figure 77 shows the DMA buffer descriptor. Table 289 describes each section.

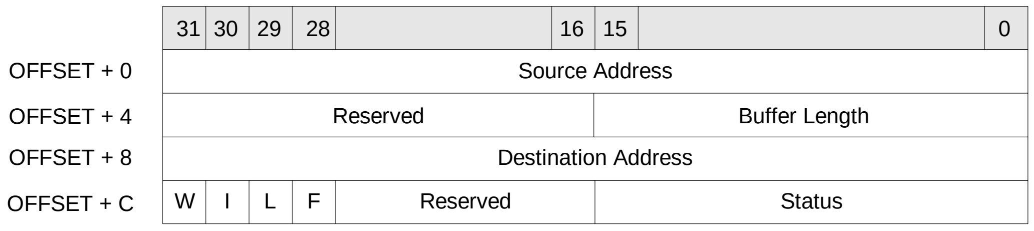


Figure 77: BBus bridge DMA buffer descriptor

Field/Section	Description
Source address	Identifies the starting location of the source data. The source address can be aligned to any byte boundary. Optimum performance results when the source address is aligned on a word boundary
Buffer length	Indicates the number of bytes to move between the source and the destination. After completing the transfer, the DMA controller updates this field with the actual number of bytes moved.
Destination address	Identifies the beginning of the location to which the source data will be moved. The destination address can be aligned to any byte boundary. Optimum performance results when the destination address is aligned on a word boundary.

Table 289: BBus bridge DMA buffer descriptor definition

Field/Section	Description
W	The **wrap** bit. When set, this bit tells the DMA controller that this is the last buffer descriptor within the continuous list of descriptors. The next buffer descriptor is found using the initial DMA channel buffer descriptor pointer. When the wrap bit is not set, the next buffer descriptor is found using an offset of 0x10 from the current buffer descriptor.
I	The **interrupt** bit. Tells the DMA controller to issue an interrupt to the CPU when the buffer is closed due to normal channel completion. The interrupt occurs no matter what the normal completion interrupt enable configuration is for the DMA channel.
L	The **last** bit. When set, this bit tells the DMA controller that this buffer descriptor is the last descriptor that completes an entire message frame. The DMA controller uses this bit to assert the normal channel completion status when the byte count reaches zero. If this bit is set, the DMA controller remains in the IDLE state after asserting the normal channel completion status. A write to the CE field in the DMA Channel 1/2 Control register re-enables the DMA channel.
F	The **full** bit. When set, this bit indicates that the buffer descriptor is valid and can be processed by the DMA channel. ■ The DMA channel clears this bit after completing the transfer(s). ■ The DMA channel doesn’t try a transfer with the F bit clear. ■ The DMA channel enters an IDLE state after fetching a buffer descriptor with the F bit cleared. ■ When the device driver modifies the F bit, it must also write a 1 to the CE bit in the DMA Channel 1/2 Control register to activate the idle channel.
Reserved	Write zero to this field.
Status	Not used. Read back 0x0000.

Table 289: BBus bridge DMA buffer descriptor definition

Descriptor list processing

When a DMA controller has completed the operation specified by the current buffer descriptor, the controller clears the F bit and fetches the next buffer descriptor. A DMA channel asserts the NRIP field in the DMA Status and Interrupt Enable register (see "DMA Status and Interrupt Enable register" on page 494) and returns to the idle state after fetching a buffer descriptor with the F bit in the incorrect state.

Peripheral DMA read access

Figure 78 and Figure 79 show how the DMA engine performs read accesses of an external peripheral. The CLK signal shown is for reference, and its frequency is equal to 1/2 the speed grade of the part. The rising edge of the READ_EN signal coincident with the assertion of the chip select signal must cause the peripheral to place the next quantum of data on the bus. The width of the READ_EN signal is always equal to one reference CLK period. The delay from the falling edge of CS# to the rising edge of ACK is always equal to one reference CLK period. The width of the CS# assertion is defined in the Static Memory Read Delay register (see "Static Memory Read Delay 0–3 registers" on page 236).

DMA read accesses from an external peripheral are treated as asynchronous operations by the NS9750. It is critical that the required width of the CS# assertion be computed correctly and programmed into the Static Memory Read Delay register. Total access time can be computed as shown:

$$\text{Total access time} = T_a + T_b + T_c + 10.0$$

The variables are defined as follows:

T_a =	Peripheral read access time
T_b =	Total board propagation including buffers
T_c =	One reference CLK cycle period

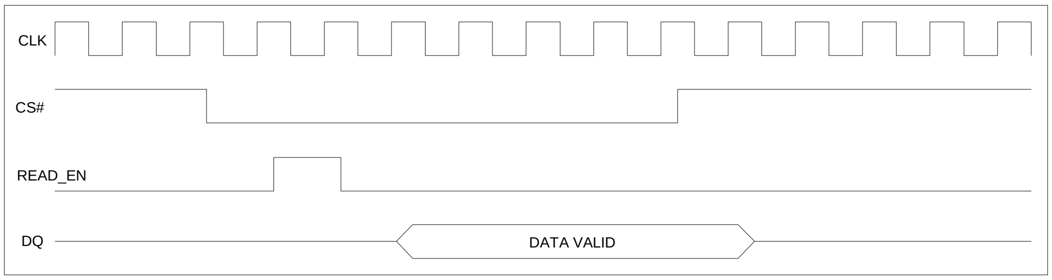


Figure 78: Peripheral DMA single read access

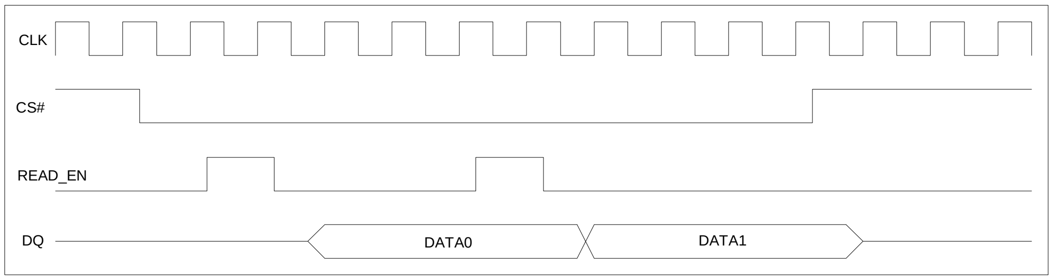


Figure 79: Peripheral DMA burst read access

Peripheral DMA write access

Figure 80 and Figure 81 show how the DMA engine performs write accesses of an external peripheral. The clock signal shown is for reference, and its clock frequency is equal to 1/2 the speed grade of the part. Data should be written on the rising edge of the WE# signal. Data and control signals are always held after the rising edge of WE# for one reference CLK cycle. The CS# signal is guaranteed to be deasserted for at least one CLK cycle between successive peripheral write accesses. The widths of the CS# assertion and the WE# assertion are defined using the Static Memory Write Delay register and the Static Memory Write Enable Delay register in the Memory Controller (see "Static Memory Read Delay 0–3 registers" on page 236 and "Static Memory Write Enable Delay 0–3 registers" on page 234).

Note that the ACK signal is not used during peripheral DMA write accesses.

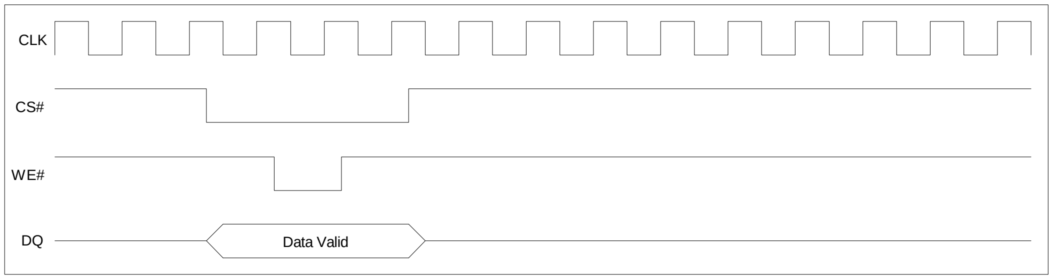


Figure 80: Peripheral DMA single write access

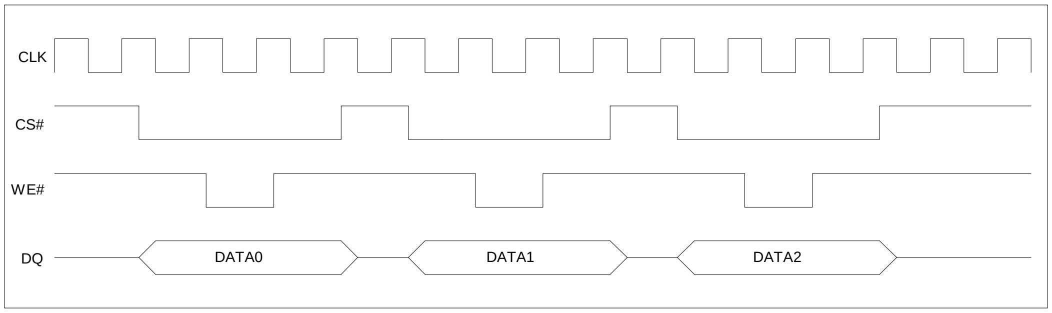


Figure 81: Peripheral DMA burst write access

Peripheral REQ signaling

An external peripheral indicates that it can accept or provide data by asserting its REQ signal. The AHB DMA controller fully processes one buffer descriptor for each assertion of the external peripheral's REQ signal.

The AHB DMA controller state machine executes these steps for each assertion of the REQ signal.

1. Fetch the next buffer descriptor in the list from system memory.
2. Read the number of bytes specified in the buffer length field from the address specified in the source address field. This data is placed in an on-chip temporary buffer.
3. Write the data from the on-chip temporary buffer to the address specified in the source address field.
4. Retire the buffer descriptor to system memory.
5. Assert any specified interrupts to the CPU.
6. Return to the idle condition and wait for the next assertion of the external peripheral's REQ signal.

For memory-to-memory DMA transfers that are initiated by software writing a 1 to the CG field in the DMA Channel 1/2 Control register, the peripheral signal REQ is ignored.

Design Limitations

The AHB DMA logic contains several design limitations. Carefully consider these limitations when making system level implementation decisions:

- The AHB DMA control logic is designed to operate on four-byte quantities, which limits the minimum number of accesses that the memory controller can perform on narrow external peripherals. Accesses to an 8-bit peripheral will always occur in multiples of four. Accesses to a 16-bit peripheral will always occur in multiples of two. Asserting the REQ signal when the peripheral is unable to meet the above conditions results in unpredictable system behavior.
- The length field in the buffer descriptor must be set to a value equal to the burst length multiplied by four. The burst length is specified in the SB/DB field in the DMA Channel 1/2 Control register.
- The peripheral can assert the REQ signal no more often than the AHB DMA response latency for the given system (see "Calculating AHB DMA response latency" on page 480).
- The REQ signal is an asynchronous input to the NS9750. For a REQ signal assertion to be found by the control logic, it must be asserted for no less than 4 AHB clock cycles and no more than 20 AHB clock cycles.
- The AHB DMA channels are allocated the unused BBus peripheral bandwidth, which limits the bandwidth available to the AHB DMA channels. Minimum bandwidth requirements can be met by allocating more AHB bus timeslots to the BBus master using the BRC registers in the System Control module.
- The AHB DMA channels provide no latency guarantee because they do not directly attach to the AHB bus. Allocating more system bandwidth reduces the worst case latency. In a fully loaded system, the response to the REQ signal assertion can be as long as 83us.

Calculating AHB DMA response latency

AHB DMA controller latency is defined as the time between the assertion of the peripheral's REQ signal and the AHB DMA channel being granted access to the AHB bus. Response latency is a function of the number of AHB timeslots given to the BBus and the number of BBus peripherals in use. Note that the BBus peripherals transferring data in non-DMA mode do not contribute to the calculation. The worst case AHB DMA response latency occurs when all of the BBus peripherals perform these operations within several microseconds of each other:

- Move the remaining data in or out of the data buffer.
- Close the buffer descriptor.
- Open a new buffer descriptor.
- Begin processing the new data buffer. This can be two steps for a transmitter.

Two AHB bandwidth calculations are defined here. The first scheme shows the worst case, where the BBus is given one out of ten AHB timeslots. The second scheme shows the best case, where the BBus is given one out of every four AHB timeslots.

Worst case:

AHB access = ((16 * 10 * 2) / 200 MHz) = 1.60us

This AHB access pattern looks like this:

Cpu, Erx, Cpu, Etx; Cpu, Lcd, Cpu, Pci; Cpu, *BBus*

Best case:

AHB access = ((16 * 4 * 2) / 200 MHz) = 0.64us

This AHB access pattern looks like this:

Cpu, Erx, Cpu, *BBus*; Cpu, Etx, Cpu, *BBus*; Cpu, Lcd, Cpu, *BBus*; Cpu, Pci; Cpu, *BBus*

Each receive channel contributes four AHB accesses to the calculation. Each transmit channel contributes five AHB accesses to the calculation. The USB device (or USB host) and IEEE 1284 are half-duplex-only peripherals, so only the transmit channel needs to be accounted for.

Also take into account adjustment for AHB DMA channel overhead: two if one DMA channel is in use and six if both DMA channels are in use. The worst case and best case equations for two DMA channels work out as shown:

Worst case latency = 1.60us * ((#Receive * 4) + (#Transmit * 5) + 6)

Best case latency = 0.64us * ((#Receive * 4) + (#Transmit * 5) + 6)

In a fully loaded system with four UARTs, IEEE 1284, and USB, the worst case latency is 83.2us and the best case latency is 33.28us.

Static RAM chip select configuration

The AHB DMA controller accesses an external peripheral using the external memory bus and one of the static RAM chip select signals (st_cs_n[N]). This table describes how to program the static RAM chip select control registers for access using the AHB DMA controller.

- Fields not explicitly listed should be left in the reset state.
- Fields listed but not defined must be defined by the user.

Register Name	Field	Value	Comment
Configuration	PB	1	System requirement.
	PM	user defined	Set to 1 if it is not necessary for the chip select signal to toggle for each access.
	MW	user defined	N/A
Read Delay	WTRD	user defined	Compute the total delay using the equation provided in "Peripheral DMA read access" on page 477. The total delay should be divided by the AHB clock period to produce this value. Round up any fractional result.
Page Read Delay	WTPG	user defined	For most applications, this value will be the same as the value for WTRD.
Output Enable Delay	WOEN	user defined	If the ACK signal is used to initiate a peripheral read, this field should be set to 0. If signal st_oe_n is used to initiate a peripheral read, this field should be set to (at least) 1.
Write Enable Delay	WWEN	user defined	For most applications, this field can be left in the default state.
Write Delay	WTWR	user defined	For most applications, this field can be left in the default state.
Turn Delay	WTTN	user defined	For most applications, this field can be left in the default state.

Table 290: Static RAM chip select configuration

Interrupt aggregation

All the peripherals on the BBus, as well as AHB DMA channels 1 and 2 in the BBus bridge, can interrupt the CPU when attention is required. These interrupts are aggregated in the BBus bridge, and a single interrupt is presented to the System Control Module on the bbus_int signal.

This function is performed in the BBus bridge because it allows the processor to quickly identify which BBus peripheral(s) is requesting attention. (See "BBus Bridge Interrupt Status register" on page 498 for more information.)

Note: The interrupt(s) must be serviced in the peripheral in which the interrupt(s) originated.

Bandwidth requirements

A single AHB timeslot is sufficient to support the ideal maximum bandwidth of the BBus peripherals plus overhead for DMA buffer descriptors. The maximum case occurs with four SPI masters, IEEE 1284, and either USB device or USB host.

- The SPI master interfaces support a maximum of 6.25 Mbps of full duplex traffic.
- The IEEE-1284 interface supports a maximum of 2 Mbps of full duplex traffic.
- The USB supports a maximum of 12 Mbps of full duplex traffic.

The total peripheral bandwidth is 64Mbps. Adding 8 Mbps for DMA buffer descriptors brings the total requirement to 9 MBps, less than one AHB timeslot with a full 16-slot rotation.

Important: Be aware that the AHB DMA channels and the BBus peripherals share an AHB timeslot. The BBus peripherals are always given a higher priority than the AHB DMA channels. The busier the BBus peripherals, then, the less available bandwidth for the AHB DMA channels.

SPI-EEPROM boot logic

SPI-EEPROM boot logic is enabled by strapping off the boot_cfg pins to the boot from SDRAM setting in the Miscellaneous System Configuration and Status register. Table 291 shows the related boot settings.

boot_cfg [1:0]	Description
00	Boot from 8-bit ROM or flash
01	Boot from 16-bit ROM or flash
10	Boot from 32-bit ROM or flash
11	Boot from SDRAM using SPI-EEPROM

Table 291: NS9750 boot configuration

When enabled, the boot logic copies the contents of an SPI-EEPROM to system memory, allowing you to boot from a low-cost serial memory. The boot logic works by interfacing to SER port B using the BBus — performing the transactions required to copy the boot code from SPI-EEPROM to external memory.

Important

- SPI-EEPROM must be connected to SER port B; the boot logic does not communicate with any other SER port.
- The endianness of the image in SPI must match the endianness of the system.
- In big endian mode, the boot image must be loaded as described in these steps:
  1. The entire boot image must be byte lane swapped before loading it into the SPI boot device. Given a word of data composed of DCBA, byte lane swapping transposes the bytes so the word looks like ABCD.
  2. The image must only be loaded into the SPI boot device using the BBus DMA controller. If the CPU directly loads the image into the SPI boot device, an incorrect image is stored in the device.

Calculation and example

This equation calculates the amount of time, in seconds, required to copy the contents of the SPI-EEPROM to external memory:

Time = (1 / freq) * $\text{EEPROM}_{\text{SIZE}}$

Example

SPI master clock frequency = 1.5 MHz

SPI-EEPROM = 256 Kb

Time for operation to complete = 175 ms

Serial Channel B configuration

When exiting the power-on reset state, serial channel B is in SPI master mode, which facilitates communication with the external SPI-EEPROM. When the copy operation is complete, serial channel B is returned to its default reset state. The next table shows which configuration fields are updated by hardware, allowing the SPI master interface to operate.

Register	Field	Value	Description
Control A	CE	0x1	Enable the channel
Control A	WLS	0x3	8 data bits per word
Control B	CSPOL	0x0	Chip select polarity to active low
Control B	MODE	0x2	SPI master mode
Control B	BITORDR	0x1	Bit order to MSB first
Bit rate	EBIT	0x1	Enable the bit rate generator
Bit rate	TMODE	0x1	Synchronous timing
Bit rate	CLKMUX	0x1	Select BBus clock as reference
Bit rate	TXCINV	0x1	Transmit clock inverted
Bit rate	N	0x00F	Create ~1.5 MHz SPI clock

Table 292: SPI master mode boot configuration

Memory Controller configuration

Note: See your ARM documentation for complete information about the memory controller.

The memory controller exits the reset state in non-operational mode. This requires the SPI-EEPROM boot logic to configure the memory controller as well as the external SDRAM before any memory access.

Important: The information required to configure the memory controller and the external SDRAM must be stored in a configuration header in the SPI-EEPROM in a contiguous block starting at address zero. Each entry in the header, with the exception of the pad entry, must be 4 bytes in length.

The size of the configuration header varies from 128 bytes to 130 bytes, due to the variable length nature of the SPI-EEPROM read command. Table 293 shows the order and contents of the configuration header.

EEPROM entry	Description
Pad entry	Variable length entry that ranges from 0 bytes to 2 bytes in length. The field length is computed by subtracting the length of the read command (including the address field) from 4. **Example** A 256 Kb EEPROM requires a 1-byte read command followed by a 2-byte address, resulting in a pad entry length of 1: 4-(1+2) = 1
Num words	Total number of words to fetch from the SPI-EEPROM. The total must include the 32-word header plus the initial discarded word. # words = $((S_1 + S_2) / 4) + 1)$ S_1 = Code image size in bytes S_2 = header = 128 bytes

Table 293: ARM boot configuration

EEPROM entry	Description
SDRAM config	All SDRAM components contain a Mode register, which has control information required to successfully access the component. The fields (available in any SDRAM specification) are defined as follows: ■ Burst length: 4 for 32-bit data bus, 8 for 16-bit data bus ■ Burst type: Sequential ■ CAS latency: Component-specific; 2 or 3 ■ OpMode: Standard ■ Write burst mode: Programmed burst length This value must be left-shifted by the number of row bits in the selected components. For example, 4Mx16 components can be combined to create a 32-bit bus. These parts require 12 row address bits. Assuming a CAS2 access, the Mode register contents would be 0x22. This value is shifted 12 places to the left (0x00022000) to form the value in the SDRAM config field.
Config register	See the Memory Controller chapter.
DynamicRefresh	See the Memory Controller chapter. For example, the value of this entry is 0x00000030 given a 100 MHz AHB clock and a 7.8125µs refresh period.
DynamicReadConfig	See the Memory Controller chapter.
DynamictRP	See the Memory Controller chapter.
DynamictRAS	See the Memory Controller chapter.
DynamictSREX	See the Memory Controller chapter.
DynamictAPR	See the Memory Controller chapter.
DynamictDAL	See the Memory Controller chapter.
DynamictWR	See the Memory Controller chapter.
DynamictRC	See the Memory Controller chapter.
DynamictRFC	See the Memory Controller chapter.
DynamictXSR	See the Memory Controller chapter.
DynamictRRD	See the Memory Controller chapter.
DynamictMRD	See the Memory Controller chapter.

Table 293: ARM boot configuration

EEPROM entry	Description
DynamicConfig0	Field B (buffer enable, in the DynamicConfig0 register) should be set to 0 (buffers disabled). The buffers will be enabled by hardware as part of the boot process. See your ARM documentation.
DynamicRasCas0	See the Memory Controller chapter.
Reserved	The remaining bytes are undefined. The final byte address of header is one of the following, depending on the pad entry length: ■ 0x7F ■ 0x80 ■ 0x81
Boot code	Must immediately follow the configuration header. The first byte address of the boot code is one of the following, depending on the pad entry length: ■ 0x80 ■ 0x81 ■ 0x82

Table 293: ARM boot configuration

SDRAM boot algorithm

Note: The SDRAM boot logic communicates only with serial channel B.

These steps describe the SDRAM boot algorithm:

1 Pins boot_cfg[1:0] are both strapped high.

Power-on reset is deasserted.

The CPU is held in reset by the SPI-EEPROM boot module.

Serial channel B comes out of reset in SPI master mode.

2 A SPI-EEPROM read command at address zero is written to the Fifo Data register. This is followed by seven more NOP entries.

3 The RXFDB and RRDY fields are monitored in Status Register A. Only complete words are read from the Fifo Data register. This process is repeated until four words have been received. An internal word counter tracks how many words have been taken from the SPI-EEPROM (see Table 294, "Boot algorithm actions," on page 489 for information about any actions taken).

4 The state machine enters a loop where four NOP words are written to the Fifo Data register and four words are read from the Fifo Data register. The RXFDB and RRDY fields are continuously monitored in Status Register A. The Fifo Data register is read only when a valid word is present.

5 The CPU is taken out of reset and serial channel B is placed into reset. Normal operation begins with the ARM fetching an instruction from system memory address 0x00000000.

Internal word counter	Action(s) taken
0x01	This word is discarded. The word is composed of the bytes shifted in while the read command and address are being shifted out, as well as the pad entry in the header.
0x02	Num words entry. This entry is saved locally.
0x03	SDRAM config entry. This entry is saved locally.
0x04 – 0x14	Memory controller entries. These entries are written to the appropriate memory controller register.
0x015	Field **I** in the memory controller Dynamic Control register is set to *PALL*, which allows several refresh operations to occur while the next 12 words are shifted in from the SPI-EEPROM.
0x016 – 0x20	No action taken. The word is discarded.
0x021	■ Field **I** in the memory controller Dynamic Control register is set to *MODE*. A system memory read operation is performed, to the address specified by the SDRAM config entry. This configures the external SDRAM devices. ■ Field **I** in the memory controller Dynamic Control register is set to *NORMAL*. Dynamic Configuration Register 0 is read and field **B** is set as required by the memory controller for normal operation.
0x022 – *End*	Each word is written to system memory starting at address 0x00000000. *End* is defined by the internal word counter matching the num words entry (0x02)

Table 294: Boot algorithm actions

BBus Bridge Control and Status registers

The BBus configuration registers are located at base address 0xA040.0000. All configuration registers are accessed with zero wait states. Table 295 lists the configuration and status registers in the BBus Bridge module. All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Address	Register
A040 0000	DMA Channel 1 Buffer Descriptor Pointer
A040 0004	DMA Channel 1 Control register
A040 0008	DMA Channel 1 Status and Interrupt Enable
A040 000C	DMA Channel 1 Peripheral Chip Select
A040 0020	DMA Channel 2 Buffer Descriptor Pointer
A040 0024	DMA Channel 2 Control register
A040 0028	DMA Channel 2 Status and Interrupt Enable
A040 002C	DMA Channel 2 Peripheral Chip Select
A040 0100	BBus Bridge Interrupt Status
A040 1004	BBus Bridge Interrupt Enable

Table 295: BBus Bridge module registers

Buffer Descriptor Pointer register

Address: A040 0000 / 0020

This register contains a 32-bit pointer to the first buffer descriptor in a contiguous list of buffer descriptors. The BBus bridge contains a Buffer Descriptor Pointer register for each DMA channel; each register is 16 bytes in length.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BuffDesc															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BuffDesc															

Register bit assignment

Bit(s)	Access	Mnemonic	Reset	Description
D31:00	RW	BuffDesc	0x00000000	**Buffer descriptor** 32-bit pointer to a buffer descriptor.

Table 296: Buffer Descriptor Pointer register bit definition

DMA Channel 1/2 Control register

Address: A040 0004 / 0024

This register contains required DMA transfer control information. The BBus bridge contains a DMA Channel Control register for each channel.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CE	CA	CG	SW		DW		SB		DB		SINC_N	DINC_N	Not used		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						INDEX									

Register bit assignment

Bit(s)	Access	Mnemonic	Reset	Description
D31	R/W	CE	0	**Channel enable** Enables and disables DMA operations, as wanted. Write a 1 to this field after a DMA channel has entered the IDLE state for any reason, to initiate additional DMA transfers.
D30	R/W	CA	0	**Channel abort** When set, causes the current DMA operation to complete, then closes the buffer.
D29	R/W	CG	0	**Channel go** When set, causes the DMA channel to exit the IDLE status and begin a DMA transfer. **Note:** The CE field must also be set. This allows software to initiate a memory-to-memory DMA transfer. External peripheral signal REQ is not used during memory-to-memory DMA transfers.
D28:27	R/W	SW	0	**Source width** Defines the size of the source data bus. Used only for peripheral to memory transfers. 00 8 bits 01 16 bits 10 32 bits 11 Undefined
D26:25	R/W	DW	0	**Destination width** Defines the size of the destination data bus. Used only for memory to peripheral transfers. 00 8 bits 01 16 bits 10 32 bits 11 Undefined

Table 297: DMA Channel 1/2 Control register bit definition

Bit(s)	Access	Mnemonic	Reset	Description
D24:23	R/W	SB	0	**Source burst** 00 1 01 2 (Recommended for 8-bit devices) 10 4 (Recommended for 16-bit devices) 11 8 (Recommended for 32-bit devices) Defines the AHB maximum burst size allowed when reading from the source.
D22:21	R/W	DB	0	**Destination burst** 00 1 01 2 (Recommended for 8-bit devices) 10 4 (Recommended for 16-bit devices) 11 8 (Recommended for 32-bit devices) Defines the AHB maximum burst size when writing to the destination. This field must be set to the same value as the source burst field.
D20	R/W	SINC_N	0	**Source address increment** 0 Increment source address pointer 1 Do not increment source address pointer Controls whether the source address pointers are incremented after each DMA transfer.
D19	R/W	DINC_N	0	**Destination address increment** 0 Increment destination address pointer 1 Do not increment destination address pointer Controls whether the destination address pointers are incremented after each DMA transfer.
D18:16	R/W	POL	0	Always set this field to 0.
D17	R/W	MODE	0	**Fly-by mode** 0 Defines a peripheral to memory fly-by write DMA transfer 1 Defines a memory-to-peripheral fly-by read DMA transfer Defines the direction of data movement for fly-by DMA transfers. This field is not used for memory-to-memory DMA transfers initiated by writing the CG field in the DMA Channel 1/2 Control register.

Table 297: DMA Channel 1/2 Control register bit definition

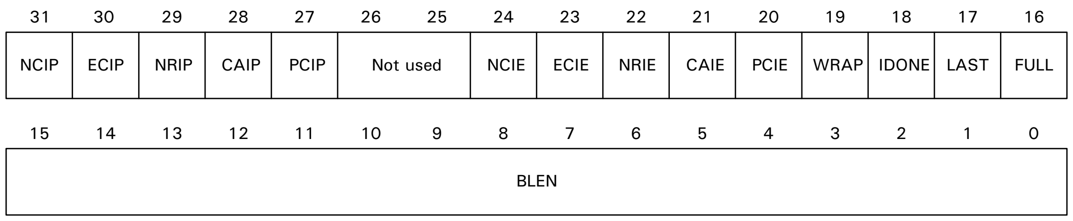
Bit(s)	Access	Mnemonic	Reset	Description
D16	R/W	RST	0	**Reset** Forces a reset of the DMA channel. Writing a 1 to this field forces all fields in the DMA Channel 1/2 Control register, *except the INDEX field,* to the reset state. The INDEX field is written with a value specified on signals ahb_wdat[9:0]. This field always reads back a 0. Writing a 1 to this field while the DMA channel is operational results in unpredictable behavior.
D15:10	N/A	Reserved	N/A	N/A
D09:00	R	INDEX	0	**Index value** Identifies the current byte offset pointer relative to the buffer descriptor pointer. This field can be written only when the RST field is being written to a 1.

Table 297: DMA Channel 1/2 Control register bit definition

DMA Status and Interrupt Enable register

Address: A040 0008 / 0028

The DMA Status and Interrupt Enable register contains the DMA transfer status and control information used for generating AHB DMA interrupt signals. The BBus bridge contains a DMA Status and Interrupt Enable register for each DMA channel.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	RW1TC	NCIP	0	**Normal completion interrupt pending** Set when a buffer descriptor has been closed. A normal DMA completion occurs when the BLEN count expires to 0 and the L bit in the Buffer descriptor is set, or when the peripheral device signals completion.
D30	RW1TC	ECIP	0	**Error completion interrupt pending** Set when the DMA channel finds either a bad buffer descriptor pointer or a bad data buffer pointer. When ECIP is set, the DMA channel stops until firmware clears the ECIP bit. The DMA channel does not advance to the next buffer descriptor. When firmware clears ECIP, the buffer descriptor is tried again from where it left off. You can use the CA bit in the DMA Channel 1/2 Control register to abort the current buffer descriptor and go to the next buffer descriptor.
D29	RW1TC	NRIP	0	**Buffer not ready interrupt pending** Set when the DMA channel finds a buffer descriptor whose F bit is in the incorrect state. The F bit must be set in order for the fetched buffer descriptor to be considered valid. If the bit is not set, the descriptor is considered invalid and the NRIP bit is set. When NRIP is set, the DMA channel stops until firmware clears the bit. The DMA channel does not advance to the next buffer descriptor.
D28	RW1TC	CAIP	0	**Channel abort interrupt pending** Set when the DMA channel finds the CA bit set in the DMA Channel 1/2 Control register. When CAIP is set, the DMA channel stops until firmware clears the bit. When CAIP is cleared, the DMA channel automatically advances to the next buffer descriptor. The CA bit must be cleared, through firmware, before CAIP is cleared. Failure to reset the CA bit causes the subsequent buffer descriptor to abort.

Table 298: DMA Status and Interrupt Enable register bit definition

Bits	Access	Mnemonic	Reset	Description
D27	RW1TC	PCIP	0	**Premature complete interrupt pending** Set when a DMA transfer is terminated by assertion of the dma_done signal. NCIP is set when PCIP is set, for backward compatibility.
D26:25	R/W	Not used	0	Always set this field to 0.
D24	R/W	NCIE	0x0	**Enable NCIP interrupt generation**
D23	R/W	ECIE	0x0	**Enable ECIP interrupt generation** Always enable during normal operation.
D22	R/W	NRIE	0x0	**Enable NRIP interrupt generation**
D21	R/W	CAIE	0x0	**Enable CAIP interrupt generation** Always enable during normal operation.
D20	R/W	PCIE	0x0	**Enable PCIP interrupt generation**
D19	R	WRAP	0x0	Debug field, indicating the last descriptor in the buffer descriptor list.
D18	R	IDONE	0x0	Debug field, indicating an interrupt on done occurrence.
D17	R	LAST	0x0	Debug field, indicating the last buffer descriptor in the current data frame.
D16	R	FULL	0x0	Debug field, indicating the status of the F bit from the current DMA buffer descriptor.
D15:00	R	BLEN	0x0000	Debug field, indicating the remaining byte transfer count.

Table 298: DMA Status and Interrupt Enable register bit definition

DMA Peripheral Chip Select register

Address: A040 000C / 002C

The DMA Peripheral Chip Select register contains the DMA peripheral chip select definition. The BBus bridge contains a DMA Peripheral Chip Select register for each channel.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used												POL	Not used	SEL	

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	R/W	Not used	0	Always set to 0.
D03	R/W	POL	0	**Chip select polarity** Defines the polarity of the memory interface chip select signal (stcsout[n]_n) connected to the external peripheral. 0 Defines an active high signal 1 Defines an active low signal
D02	R/W	Not used	0	Always set to 0.
D01:00	R/W	SEL	0	**Chip select selection** Defines which of the four memory interface chip select signals (stcsout[n]_n) is connected to the external peripheral. **Value** **Chip select** 0 CS[0] 1 CS[1] 2 CS[2] 3 CS[3] **Note:** This field is not used for memory-to-memory transfers.

Table 299: DMA Peripheral Chip Select register

BBus Bridge Interrupt Status register

Address: A040 1000

This register contains the interrupt status of the BBus peripherals. All interrupts must be serviced in the originating module.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used						ADMA2	ADMA1	Not used							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used				1284	I2C	S4TX	S4RX	S3TX	S3RX	S2TX	S2RX	S1TX	S1RX	USB	BBDMA

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:26	R	Not used	00	Always set this field to 0.
D25	R	ADMA2	0	AHB DMA channel #2 has asserted its interrupt.
D24	R	ADMA1	0	AHB DMA channel #1 has asserted its interrupt.
D23:13	R	Not used	0x000	Always set this field to 0.
D12	R	Not used	0	Always write to 0.
D11	R	1284	0	IEEE-1284 module has asserted its interrupt.
D10	R	I2C	0	I2C module has asserted its interrupt.
D09	R	SDTX	0	SER transmit module D has asserted its interrupt.
D08	R	SDRX	0	SER receive module D has asserted its interrupt.
D07	R	SCTX	0	SER transmit module C has asserted its interrupt.
D06	R	SCRX	0	SER receive module C has asserted its interrupt.
D05	R	SATX	0	SER transmit module A has asserted its interrupt.
D04	R	SARX	0	SER receive module A has asserted its interrupt.
D03	R	SBTX	0	SER transmit module B has asserted its interrupt.
D02	R	SBRX	0	SER receive module B has asserted its interrupt.

Table 300: BBus Bridge Interrupt Status register

Bits	Access	Mnemonic	Reset	Description
D01	R	USB	0	USB module has asserted its interrupt.
D00	R	BBDMA	0	BBus DMA module has asserted its interrupt.

Table 300: BBus Bridge Interrupt Status register

BBus Bridge Interrupt Enable register

Address: A040 1004

The BBus Bridge Interrupt Enable register allows you to enable or disable BBus interrupts on an individual basis as well as a global basis. Writing a 1 to a bit enables the interrupt, allowing it to be included in the aggregate signal that is sent to the vector interrupt controller in the System Control module. These fields affect only the generation of the signal sent to the vector interrupt controller (VIC); they do not affect the originating modules.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GLBL	Not used					DMA2	DMA1	Not used							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used				1284E	I2CE	S4TXE	S4RXE	S3TXE	S3RXE	S2TXE	S2RXE	S1TXE	S1RXE	USBE	DMAE

Register bit assignment

Note: Enable = Set to 1.

Bits	Access	Mnemonic	Reset	Description
D31	R/W	GLBL	0	Enable the aggregate interrupt signal to propagate to the VIC in the System Control module.
D30:26	R/W	Not used	0	Always set this field to 0.
D25	R/W	DMA2	0	Enable interrupt from AHB DMA Channel #2.
D24	R/W	DMA1	0	Enable interrupt from AHB DMA Channel #1.

Table 301: BBus Bridge Interrupt Enable register bit definition

Bits	Access	Mnemonic	Reset	Description
D23:13	R/W	Not used	0x000	Always set this field to 0.
D12	R/W	Not used	0	**Always write to 0.**
D11	R/W	1284E	0	Enable interrupt from IEEE-1284 module.
D10	R/W	I2CE	0	Enable interrupt from I2C module.
D09	R/W	SDTXE	0	Enable interrupt from SER transmit module D.
D08	R/W	SDRXE	0	Enable interrupt from SER receive module D.
D07	R/W	SCTXE	0	Enable interrupt from SER transmit module C.
D06	R/W	SCRXE	0	Enable interrupt from SER receive module C.
D05	R/W	SATXE	0	Enable interrupt from SER transmit module A.
D04	R/W	SARXE	0	Enable interrupt from SER receive module A.
D03	R/W	SBTXE	0	Enable interrupt from SER transmit module B.
D02	R/W	SBRXE	0	Enable interrupt from SER receive module B.
D01	R/W	USBE	0	Enable interrupt from USB module.
D00	R/W	DMAE	0	Enable aggregate interrupt from BBus DMA module. These interrupts can be controlled on a per-DMA channel basis in the BBus utility module.

Table 301: BBus Bridge Interrupt Enable register bit definition

BBus DMA Controller

C H A P T E R 9

The NS9750 ASIC BBus *subsystem* contains two DMA controllers, each with 16 channels.

Note: These DMA controllers are different than the AHB DMA controllers discussed in the BBus Bridge chapter.

About the BBus DMA controllers

There are two BBus DMA controllers. One DMA controller supports all BBus peripherals except the USB device; the other DMA controller is dedicated to the USB device interface (see the USB Controller Module chapter for more information). Each DMA controller contains 16 channels, and each DMA channel moves data between external memory and internal peripherals in fly-by mode, minimizing CPU intervention.

Figure 82 shows the data flow for fly-by DMA transfers.

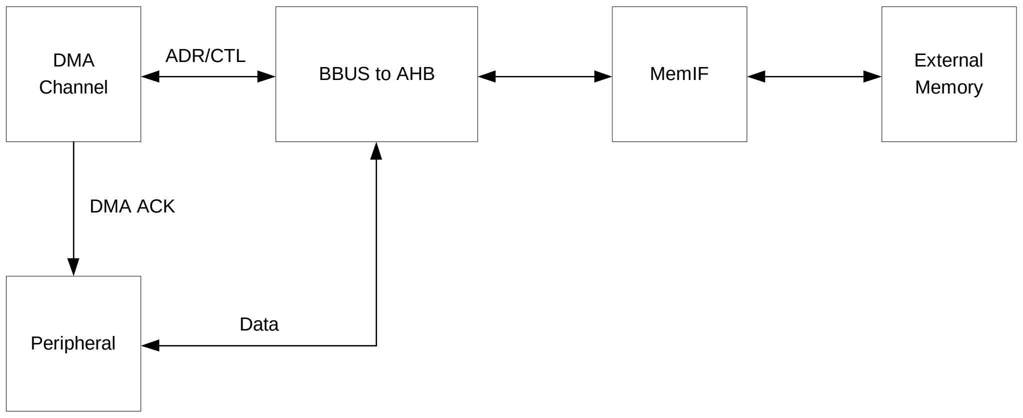


Figure 82: DMA fly-by transfers

Note: Neither memory-to-memory transfers nor DMA transfers to external peripherals are supported.

Each DMA controller has a state machine and a block of static RAM, referred to as *context* RAM.

- The context RAM contains the current state of each DMA channel.
- The single state machine supports all DMA channels in parallel, by context-switching from channel to channel.

Figure 83 shows the BBus DMA controller block.

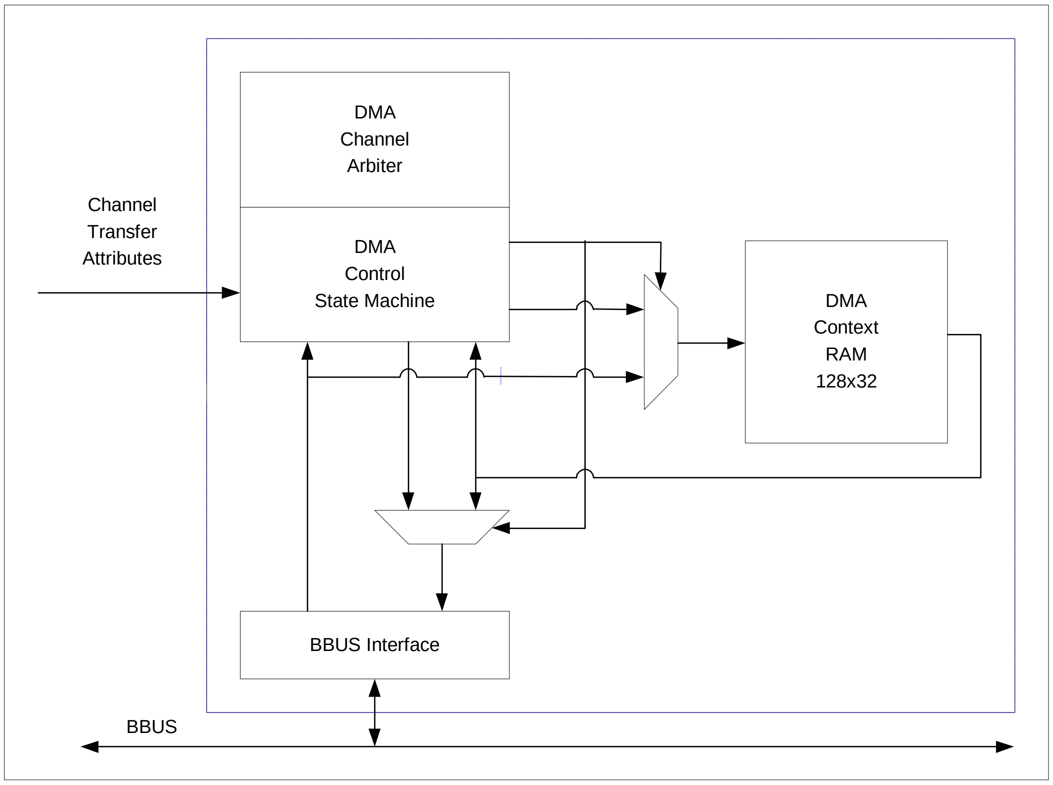


Figure 83: DMA controller block

Each DMA controller arbiter determines in which channel the state machine currently is operating.

DMA context memory

Each DMA controller maintains state for all 16 channels using an on-chip SRAM known as the *context memory*. One 128x32 single port SRAM macrocell comprises this memory. Table 302 defines the entries that describe the state of each DMA channel.

Offset	Description
0x00	Buffer descriptor pointer
0x01	Control register
0x02	Status register
0x03	Unused
0x04	Source Address register
0x05	Buffer Length register
0x06	Destination Address register
0x07	Control flags and transfer status

Table 302: DMA context memory entry

DMA buffer descriptor

All DMA channels operate using a buffer descriptor. Each DMA channel remains idle until enabled through the DMA Channel Control register. When a DMA channel is activated, it reads the DMA buffer descriptor pointed to by the Buffer Descriptor Pointer register. When the current descriptor is retired, the next descriptor is accessed from a circular buffer.

Each DMA buffer descriptor is four 32-bit words in length. Multiple buffer descriptors are located in circular buffers of 1024 bytes, with a maximum of 64 buffer descriptors. The DMA channel's buffer descriptor pointer provides the first buffer descriptor address. Subsequent buffer descriptors are found adjacent to the first descriptor. The final buffer descriptor is defined with its *W* bit set. When the DMA channel encounters the W bit, the channel wraps around to the first descriptor.

Note: Configuring a DMA channel for more than the maximum number of buffer descriptors results in unpredictable behavior.

Figure 84 shows the DMA buffer descriptor. Table 303 explains each buffer descriptor component.

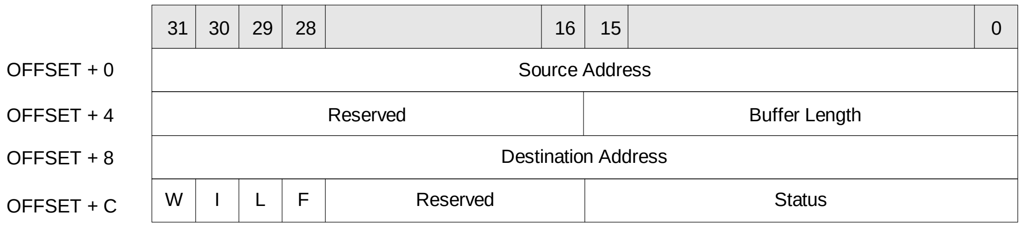


Figure 84: DMA buffer descriptor

Field	Description
Source address	Identifies the starting location of the source data buffer. ■ For **transmit** buffers. The source address can start on any byte boundary. ■ For **receive** buffers. The source address must be word-aligned. Be sure the source address field points to an existing memory location.
Buffer length	■ Indicates, in *fly-by peripheral-to-memory operations*, the maximum number of bytes available in the receive buffer pointed to by the source buffer pointer. After filling a receive buffer with peripheral data, the DMA controller updates this field with the initial buffer length less the actual receive data byte count. **Note:** The buffer length must be a multiple of four bytes. ■ Indicates, in *fly-by memory-to-peripheral operations*, the number of bytes to move from the source address pointer to the peripheral device. After completing a transmit buffer descriptor, the DMA controller updates this field with the initial buffer length less the actual transmit data byte count (useful for error conditions). In either mode, this field is limited to 16 bits, which supports a maximum transfer size of 65535 bytes.
Destination address	**This field is not used in BBus DMA transfers.** The field is updated with all zeroes when the descriptor is retired.
W	The **wrap** bit. When set, this bit informs the DMA controller that this is the last buffer descriptor within the continuous list of descriptors for the channel. The next buffer descriptor is found using the initial DMA channel buffer descriptor pointer. When the WRAP bit is not set, the next buffer descriptor is found using a 16 byte offset from the current buffer descriptor.

Table 303: DMA buffer descriptor definition

Field	Description
I	The **interrupt** bit. When set, this bit tells the DMA controller to issue an interrupt to the CPU when the buffer is closed due to a normal channel completion. The interrupt occurs no matter what the normal completion interrupt enable configuration is for the DMA channel.
L	The **last** bit. This bit indicates end-of-packet status. ■ In fly-by peripheral-to-memory operations, this bit indicates that the buffer was closed due to an end-of-packet status signal from the peripheral to the DMA controller. ■ In fly-by memory-to-peripheral operations, this bit indicates to the DMA controller that this buffer descriptor marks the end of the packet. **Note:** For USB-IN transactions (DMA read), this bit must always be set to 1.
F	The **full** bit. When set, this bit indicates that the buffer is full. A DMA channel sets this bit after filling a buffer. A DMA channel clears this bit after emptying a buffer. A DMA channel does not try to empty a buffer with the F bit clear. Similarly, a DMA channel does not try to fill a buffer with the F bit set. When firmware modifies the F bit, the firmware also must write a 1 to the CE bit in the DMA Channel Control register to activate the idle channel.
Reserved	You must write a 0 to this field.
Status	16-bit status field. The USB and serial controllers use this field to store transmit and receive status words that result from a completed transmit or receive data frame. **Be advised:** The BBus DMA buffer descriptor status field may not reflect the occurrence of a receive overrun in the serial module.

Table 303: DMA buffer descriptor definition

DMA transfer status

The DMA buffer descriptor status field is updated when the buffer descriptor is retired. Tables 304 through 309 provide a brief description of the 16-bit status fields for each peripheral. See the appropriate chapters in this manual for more information about each bit.

Bits	Mnemonic	Description
15	MATCH1	Receive character match #1
14	MATCH2	Receive character match #2
13	MATCH3	Receive character match #3
12	MATCH4	Receive character match #4
11	BGAP	Buffer gap timeout
10	CGAP	Character gap timeout
09:04	UNUSED	Not used — read back 0
03	RBRK	Receive line break
02	RFE	Receive frame error
01	RPE	Receive parity error
00	ROVER	Receive overrun error

Table 304: Peripheral bit fields: Serial controller — UART RX mode

Bits	Mnemonic	Description
15	MATCH1	Receive character match #1
14	MATCH2	Receive character match #2
13	MATCH3	Receive character match #3
12	MATCH4	Receive character match #4
11:00	UNUSED	Not used — read back 0

Table 305: Peripheral bit fields: Serial controller — SPI RX mode

Bits	Mnemonic	Description
15:00	UNUSED	Not used — read back 0

Table 306: Peripheral bit fields: Serial controller — UART TX mode

Bits	Mnemonic	Description
15:00	UNUSED	Not used — read back 0

Table 307: Peripheral bit fields: Serial controller — SPI TX mode

Bits	Mnemonic	Description
15:14	STATE	00 Undefined 01 Data phase transaction 10 Status phase transaction 11 No-data status phase transaction Defines the state of the endpoint after the most recent communication with the USB device module. This field is used primarily for debugging.
13	M31	See the USB Controller module chapter.
12	M30	See the USB Controller module chapter.
11:00	CIA	**If field M30 equals 1,** this field contains the least significant 12 bits of the Setup command address. Because all Setup command addresses are required to be in the format ???, where ??? is not equal to 000, the most significant nibble is zero. **If the M30 field equals 0,** this field contains the configuration, interface, and alternate information for the specified endpoint.

Table 308: Peripheral bit fields: USB device controller

Bits	Mnemonic	Description
15:00	UNUSED	Not used — read back 0

Table 309: Peripheral bit fields: IEEE 1284 controller

DMA channel assignments

Each BBus DMA controller contains 16 DMA channels. Controller DMA1 is dedicated to the BBus peripherals. Controller DMA2 is dedicated to the USB device endpoints. Any given DMA channel is hard-wired to a peripheral.

Table 310 indicates which peripherals are hard-wired to which DMA channels, and the DMA mode (direction) required for each. These are the DMA modes:

- FBR — Fly-by memory-to-peripheral
- FBW — Fly-by peripheral-to-memory
- FBRW — Fly-by programmable for either direction

DMA	Channel	DMA channel peripheral	Fly-by direction
DMA1	1	SER channel B receiver	FBW
DMA1	2	SER channel B transmitter	FBR
DMA1	3	SER channel A receiver	FBW
DMA1	4	SER channel Atransmitter	FBR
DMA1	5	SER channel C receiver	FBW
DMA1	6	SER channel C transmitter	FBR
DMA1	7	SER channel D receiver	FBW
DMA1	8	SER channel D transmitter	FBR
DMA1	9	1284 command receiver	FBW
DMA1	10	Unused	N/A
DMA1	11	1284 data receiver	FBW
DMA1	12	1284 data transmitter	FBR
DMA1	13	Unused	N/A
DMA1	14	Unused	N/A
DMA1	15	Unused	N/A
DMA1	16	Unused	N/A

Table 310: DMA channel assignments

DMA	Channel	DMA channel peripheral	Fly-by direction
DMA2	1	USB device control-OUT endpoint #0	FBW
DMA2	2	USB device control-IN endpoint #0	FBR
DMA2	3	USB device endpoint#1	FBRW
DMA2	4	USB device endpoint#2	FBRW
DMA2	5	USB device endpoint#3	FBRW
DMA2	6	USB device endpoint#4	FBRW
DMA2	7	USB device endpoint#5	FBRW
DMA2	8	USB device endpoint#6	FBRW
DMA2	9	USB device endpoint#7	FBRW
DMA2	10	USB device endpoint#8	FBRW
DMA2	11	USB device endpoint#9	FBRW
DMA2	12	USB device endpoint#10	FBRW
DMA2	13	USB device endpoint#11	FBRW
DMA2	14	Unused	N/A
DMA2	15	Unused	N/A
DMA2	16	Unused	N/A

Table 310: DMA channel assignments

DMA Control and Status registers

The configuration registers for DMA1 are located at 0x9000 0000. The configuration registers for DMA2 are located at 0x9011 0000. All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Table 311 is a single DMA controller address map.

Important: Be aware that the registers listed in this table are not discrete registers; they are combined with other information and stored in the context SRAM within each DMA module. The offsets allow address bits [08:05] to encode the DMA channel number.

Offset	Description
9000 0000 / 9011 0000	DMA Channel 1 Buffer Descriptor Pointer
9000 0020 / 9011 0020	DMA Channel 2 Buffer Descriptor Pointer
9000 0040 / 9011 0040	DMA Channel 3 Buffer Descriptor Pointer
9000 0060 / 9011 0060	DMA Channel 4 Buffer Descriptor Pointer
9000 0080 / 9011 0080	DMA Channel 5 Buffer Descriptor Pointer
9000 00A0 / 9011 00A0	DMA Channel 6 Buffer Descriptor Pointer
9000 00C0 / 9011 00C0	DMA Channel 7 Buffer Descriptor Pointer
9000 00E0 / 9011 00E0	DMA Channel 8 Buffer Descriptor Pointer
9000 0100 / 9011 0100	DMA Channel 9 Buffer Descriptor Pointer
9000 0120 / 9011 0120	DMA Channel 10 Buffer Descriptor Pointer
9000 0140 / 9011 0140	DMA Channel 11 Buffer Descriptor Pointer
9000 0160 / 9011 0160	DMA Channel 12 Buffer Descriptor Pointer
9000 0180 / 9011 0180	DMA Channel 13 Buffer Descriptor Pointer
9000 01A0 / 9011 01A0	DMA Channel 14 Buffer Descriptor Pointer
9000 01C0 / 9011 01C0	DMA Channel 15 Buffer Descriptor Pointer
9000 01E0 / 9011 01E0	DMA Channel 16 Buffer Descriptor Pointer
9000 0010 / 9011 0010	DMA Channel 1 Control register
9000 0030 / 9011 0030	DMA Channel 2 Control register
9000 0050 / 9011 0050	DMA Channel 3 Control register
9000 0070 / 9011 0070	DMA Channel 4 Control register
9000 0090 / 9011 0090	DMA Channel 5 Control register
9000 00B0 / 9011 00B0	DMA Channel 6 Control register
9000 00D0 / 9011 00D0	DMA Channel 7 Control register
9000 00F0 / 9011 00F0	DMA Channel 8 Control register
9000 0110 / 9011 0110	DMA Channel 9 Control register

Table 311: DMA Control and Status register address map

Offset	Description
9000 0130 / 9011 0130	DMA Channel 10 Control register
9000 0150 / 9011 0150	DMA Channel 11 Control register
9000 0170 / 9011 0170	DMA Channel 12 Control register
9000 0190 / 9011 0190	DMA Channel 13 Control register
9000 01B0 / 9011 01B0	DMA Channel 14 Control register
9000 01D0 / 9011 01D0	DMA Channel 15 Control register
9000 01F0 / 9011 01F0	DMA Channel 16 Control register
9000 0014 / 9011 0014	DMA Channel 1 Status/Interrupt Enable register
9000 0034 / 9011 0034	DMA Channel 2 Status/Interrupt Enable register
9000 0054 / 9011 0054	DMA Channel 3 Status/Interrupt Enable register
9000 0074 / 9011 0074	DMA Channel 4 Status/Interrupt Enable register
9000 0094 / 9011 0094	DMA Channel 5 Status/Interrupt Enable register
9000 00B4 / 9011 00B4	DMA Channel 6 Status/Interrupt Enable register
9000 00D4 / 9011 00D4	DMA Channel 7 Status/Interrupt Enable register
9000 00F4 / 9011 00F4	DMA Channel 8 Status/Interrupt Enable register
9000 0114 / 9011 0114	DMA Channel 9 Status/Interrupt Enable register
9000 0134 / 9011 0134	DMA Channel 10 Status/Interrupt Enable register
9000 0154 / 9011 0154	DMA Channel 11 Status/Interrupt Enable register
9000 0174 / 9011 0174	DMA Channel 12 Status/Interrupt Enable register
9000 0194 / 9011 0194	DMA Channel 13 Status/Interrupt Enable register
9000 01B4 / 9011 01B4	DMA Channel 14 Status/Interrupt Enable register
9000 01D4 / 9011 01D4	DMA Channel 15 Status/Interrupt Enable register
9000 01F4 / 9011 01F4	DMA Channel 16 Status/Interrupt Enable register

Table 311: DMA Control and Status register address map

DMA Buffer Descriptor Pointer

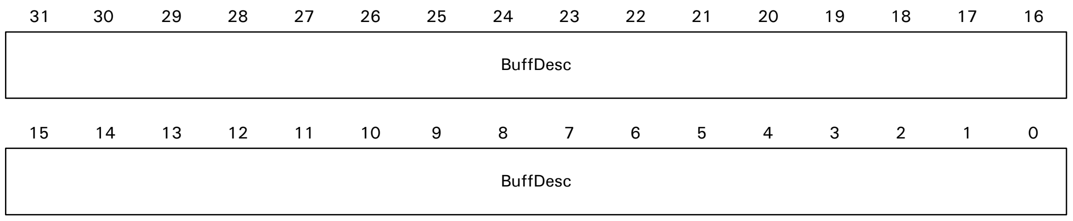
Address: DMA1

9000 0000 / 0020 / 0040 / 0060 / 0080 / 00A0 / 00C0 / 00E0 / 0100 / 0120 / 0140 / 0160 / 0180 / 01A0 / 01C0 / 01E0

Address: DMA2

9011 0000 / 0020 / 0040 / 0060 / 0080 / 00A0 / 00C0 / 00E0 / 0100 / 0120 / 0140 / 0160 / 0180 / 01A0 / 01C0 / 01E0

The DMA Buffer Descriptor Pointer register contains a 32-bit pointer to the first buffer descriptor in a contiguous list of buffer descriptors. There is one Buffer Descriptor Pointer for each channel within each DMA controller module. Each buffer descriptor is 16 bytes in length.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	BuffDesc	0x00000000	**Buffer descriptor** 32-bit pointer to a buffer descriptor.

Table 312: BBus DMA Buffer Descriptor Pointer register bit definition

DMA Control register

Address: DMA1

9000 0010 / 0030 / 0050 / 0070 / 0090 / 00B0 / 00D0 / 00F0 / 0110 / 0130 / 0150 / 0170 / 0190 / 01B0 / 01D0 / 01F0

Address: DMA2

9011 0010 / 0030 / 0050 / 0070 / 0090 / 00B0 / 00D0 / 00F0 / 0110 / 0130 / 0150 / 0170 / 0190 / 01B0 / 01D0 / 01F0

The DMA Control register contains required transfer control information. There is a DMA Control register for each channel within each DMA controller module.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CE	CA	BB		MODE		BTE		REQ	BDR	SINC_N	Not used			SIZE	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STATE						INDEX									

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	CE	0	**Channel enable** 0 Disables DMA operations 1 Enables DMA operations Enables and disables DMA operations, as wanted.
D30	R/W	CA	0	**Channel abort** When set, causes the current DMA operation to complete and closes the buffer.
D29:28	R/W	BB	0	**Bus bandwidth** Always set to 0.

Table 313: BBus DMA Control register bit definition

Bits	Access	Mnemonic	Reset	Description
D27:26	R/W	MODE	0	**Fly-by mode** 00 Fly-by write (peripheral-to-memory) 01 Fly-by read (memory-to-peripheral) 10 Undefined 11 Undefined Defines the fly-by transfer mode.
D25:24	R/W	BTE	0	**Burst transfer enable** 00 1 operand 01 2 operands 10 4 operands (**Recommended**) 11 Reserved Determines whether the DMA channel can use burst transfers through the bus. This configuration applies to both buffer descriptor and peripheral data access.
D23	R/W	REQ	0	Always set to 0.
D22	R/W	BDR	0	**Buffer descriptor refetch** Causes the DMA controller to refetch the current buffer descriptor before proceeding. This is necessary to retransmit erroneous packets sent from the USB device to the USB host. Hardware automatically clears this field after refetching the buffer descriptor.
D21	R/W	SINC_N	0	**Source address increment field** 0 Increment source address pointer 1 Do not increment source address pointer Controls whether the source address pointers are incremented after each DMA transfer. The DMA controller uses this field in all modes whenever referring to a memory address.
D20:18	R/W	Not used	0	N/A
D17:16	R/W	SIZE	0	**Size field** Must always be set to 0. The datapath between the BBus and AHB bus is 32 bits. If the system memory bus is less than 32 bits, the translation is handled in the memory controller.

Table 313: BBus DMA Control register bit definition

Bits	Access	Mnemonic	Reset	Description
D15:10	R	STATE	0	**State field** 0x00 Idle 0x20 Transfer in progress 0x18 Update buffer descriptor Describes the current state of the DMA controller state machine.
D09:00	R	INDEX	0	**Index value** Identifies the current byte offset pointer relative to the buffer descriptor pointer.

Table 313: BBus DMA Control register bit definition

DMA Status/Interrupt Enable register

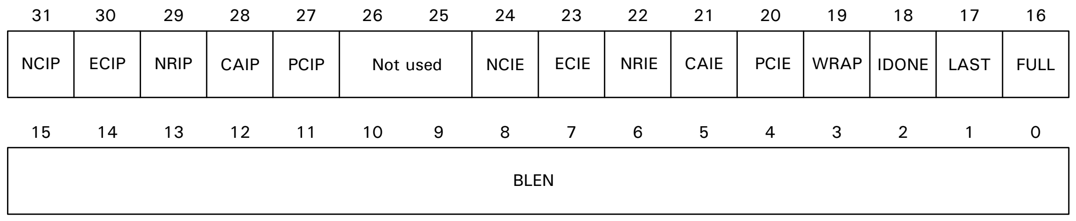
Address: DMA1

9000 0014 / 0034 / 0054 / 0074 / 0094 / 00B4 / 00D4 / 00F4 / 0114 / 0134 / 0154 / 0174 / 0194 / 01B4 / 01D4 / 01F4

Address: DMA2

9011 0014 / 0034 / 0054 / 0074 / 0094 / 00B4 / 00D4 / 00F4 / 0114 / 0134 / 0154 / 0174 / 0194 / 01B4 / 01D4 / 01F4

The DMA Status/Interrupt Enable register contains DMA transfer status as well as control information for generating interrupt signals. There is a DMA Status/Interrupt Enable register for each channel within each DMA controller module.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	RW1TC	NCIP	0	**Normal completion interrupt pending** Set when a buffer descriptor is closed (for normal conditions). An interrupt is generated when either the NCIE (D24) bit is set or the IDONE (D18) bit is found active in the current buffer descriptor. A normal DMA channel completion occurs when the BLEN count (15:00) expires to 0 or when a peripheral device signals completion.
D30	RW1TC	ECIP	0	**Error completion interrupt pending** Set when the DMA channel encounters either a bad buffer descriptor pointer or a bad data buffer pointer. An interrupt is generated if the ECIE (D23) bit is set. The DMA channel stops until the CE bit (in the DMA Channel Control register) is written to a 1 by firmware. The DMA channel does not advance to the next buffer descriptor. When ECIP Is cleared by firmware, the buffer descriptor is tried again from where it left off. The CA bit in the appropriate DMA Channel Control register can be used to abort the current buffer descriptor and advance to the next buffer.
D29	RW1TC	NRIP	0	**Buffer not ready interrupt pending** Set when the DMA channel finds a buffer descriptor whose F bit is in the incorrect state. An interrupt is generated if the NRIE (D22) bit is set. When NRIP is set, the DMA channel stops until firmware writes a 1 to the CE field (in the DMA Channel Control register). The DMA channel does not advance to the next buffer descriptor.

Table 314: DMA Status/Interrupt Enable register bit definition

Bits	Access	Mnemonic	Reset	Description
D28	RW1TC	CAIP	0	**Channel abort interrupt pending** Set when the DMA channel finds the CA bit set in the DMA Channel Control register. An interrupt is generated when the CAIE (D21) bit is set. When CAIP Is set, the DMA channel retires the current buffer descriptor and stops until firmware writes a 1 to the CE bit (in the appropriate DMA Channel Control register). **Note:** The CA bit must be cleared, using firmware, before the CE field is written. Failure to reset the CA bit causes the subsequent buffer descriptor to abort.
D27	R/W	PCIP	0	**Premature complete interrupt pending** Set when the DMA channel, configured for fly-by write mode, receives an end-of-transfer indicator from the peripheral while processing a DMA buffer descriptor. An interrupt is generated if the PCIE (D20) bit is set. The DMA channel continues processing buffer descriptors. NCIP is set when PCIP is set, for backward compatibility.
D26:25	R/W	Unused	0	Always set to 0.
D24	R/W	NCIE	0	Enable NCIP interrupt generation.
D23	R/W	ECIE	0	Enable ECIP interrupt generation. This bit always should be enabled during normal operation.
D22	R/W	NRIE	0	Enable NRIP interrupt generation.
D21	R/W	CAIE	0	Enable CAIP interrupt generation. This bit always should be enabled during normal operation.
D20	R/W	PCIE	0	Enable PCIP interrupt generation.
D19	R	WRAP	0	Debug field, indicating the last descriptor in the descriptor list.
D18	R	IDONE	0	Debug field, indicating interrupt on done.
D17	R	LAST	0	Debug field, indicating the last buffer descriptor in the current data frame.
D16	R	FULL	0	Debug field, indicating the buffer is full.
D15:00	R	BLEN	0x000	Debug field, indicating the remaining byte transfer count.

Table 314: DMA Status/Interrupt Enable register bit definition

BBus Utility

C H A P T E R 1 0

The BBus utility provides chip-level support for the low speed peripherals in the NS9750 ASIC that reside on the Digi proprietary BBus. The BBus utility handles functions such as bus monitors, GPIO control, and peripheral reset.

BBus Utility Control and Status registers

The BBus Utility configuration registers are located at base address 0x9060 0000. Table 315 lists the control and status registers in the BBus Utility.

All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Address	Description
9060 0000	Master Reset register
9060 0004	Reserved (Do not write to this address)
9060 0010	GPIO Configuration Register #1
9060 0014	GPIO Configuration Register #2
9060 0018	GPIO Configuration Register #3
9060 001C	GPIO Configuration Register #4
9060 0020	GPIO Configuration Register #5
9060 0024	GPIO Configuration Register #6
9060 0028	GPIO Configuration Register #7
9060 0030	GPIO Control Register #1
9060 0034	GPIO Control Register #2
9060 0040	GPIO Status Register #1
9060 0044	GPIO Status Register #2
9060 0050	BBus Monitor register
9060 0060	BBus DMA Interrupt Status register
9060 0064	BBus DMA Interrupt Enable register
9060 0070	USB Configuration register
9060 0080	Endian Configuration register
9060 0090	ARM Wake-up register

Table 315: BBus Utility configuration and status register address map

Master Reset register

Address: 9060 0000

The Master Reset register contains the reset control signals for all BBus peripherals. All BBus peripherals, except the bridge, are held in reset after power-on reset is deasserted. All reset bits in this register are active high.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used			Reserved					I2C	1284	SerD	SerC	SerA	SerB	USB	DMA

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:13	R	Not used	0x0	Always read as 0x0.
D12:08	N/A	Reserved	N/A	N/A
D07	R/W	I^2C	1	I^2C Controller reset
D06	R/W	1284	1	IEEE 1284 Controller reset
D05	R/W	SerD	1	Serial Controller port D reset
D04	R/W	SerC	1	Serial Controller port C reset
D03	R/W	SerA	1	Serial Controller port A reset
D02	R/W	SerB	1	Serial Controller port B reset
D01	R/W	USB	1	USB Controller reset
D00	R/W	DMA	1	BBus DMA reset

Table 316: Master Reset register

GPIO Configuration registers

GPIO Configuration registers #1 – #7 contain the configuration information for each of the 50 GPIO pins in the NS9750. Each GPIO pin is defined to have up to four functions. Configure each pin for the appropriate function and direction, as shown in Table 324: "GPIO Configuration register options" on page 528.

GPIO Configuration Register #7

Address: 9060 0028

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								gpio49				gpio48			

Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A
D07:04	R/W	gpio49	0x3	gpio[49] configuration
D03:00	R/W	gpio48	0x3	gpio[48] configuration

Table 317: GPIO Configuration Register #7

GPIO Configuration Register #6

Address: 9060 0024

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
gpio47				gpio46				gpio45				gpio44			

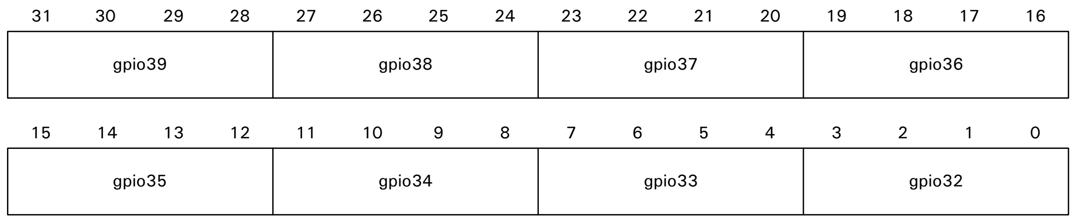
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio43				gpio42				gpio41				gpio40			

Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio47	0x3	gpio[47] configuration
D27:24	R/W	gpio46	0x3	gpio[46] configuration
D23:20	R/W	gpio45	0x3	gpio[45] configuration
D19:16	R/W	gpio44	0x3	gpio[44] configuration
D15:12	R/W	gpio43	0x3	gpio[43] configuration
D11:08	R/W	gpio42	0x3	gpio[42] configuration
D07:04	R/W	gpio41	0x3	gpio[41] configuration
D03:00	R/W	gpio40	0x3	gpio[40] configuration

Table 318: GPIO Configuration Register #6

GPIO Configuration Register #5

Address: 9060 0020



Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio39	0x3	gpio[39] configuration
D27:24	R/W	gpio38	0x3	gpio[38] configuration
D23:20	R/W	gpio37	0x3	gpio[37] configuration
D19:16	R/W	gpio36	0x3	gpio[36] configuration
D15:12	R/W	gpio35	0x3	gpio[35] configuration
D11:08	R/W	gpio34	0x3	gpio[34] configuration
D07:04	R/W	gpio33	0x3	gpio[33] configuration
D03:00	R/W	gpio32	0x3	gpio[32] configuration

Table 319: GPIO Configuration Register #5

GPIO Configuration Register #4

Address: 9060 001C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
gpio31				gpio30				gpio29				gpio28			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio27				gpio26				gpio25				gpio24			

Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio31	0x3	gpio[31] configuration
D27:24	R/W	gpio30	0x3	gpio[30] configuration
D23:20	R/W	gpio29	0x3	gpio[29] configuration
D19:16	R/W	gpio28	0x3	gpio[28] configuration
D15:12	R/W	gpio27	0x3	gpio[27] configuration
D11:08	R/W	gpio26	0x3	gpio[26] configuration
D07:04	R/W	gpio25	0x3	gpio[25] configuration
D03:00	R/W	gpio24	0x3	gpio[24] configuration

Table 320: GPIO Configuration Register #4

GPIO Configuration Register #3

Address: 9060 0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
gpio23				gpio22				gpio21				gpio20			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio19				gpio18				gpio17				gpio16			

Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio23	0x3	gpio[23] configuration
D27:24	R/W	gpio22	0x3	gpio[22] configuration
D23:20	R/W	gpio21	0x3	gpio[21] configuration
D19:16	R/W	gpio20	0x3	gpio[20] configuration
D15:12	R/W	gpio19	0x3	gpio[19] configuration
D11:08	R/W	gpio18	0x3	gpio[18] configuration
D07:04	R/W	gpio17	0x3	gpio[17] configuration
D03:00	R/W	gpio16	0x3	gpio[16] configuration

Table 321: GPIO Configuration register #3

GPIO Configuration Register #2

Address: 9060 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
gpio15				gpio14				gpio13				gpio12			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio11				gpio10				gpio9				gpio8			

Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio15	0x3	gpio[15] configuration
D27:24	R/W	gpio14	0x3	gpio[14] configuration
D23:20	R/W	gpio13	0x3	gpio[13] configuration
D19:16	R/W	gpio12	0x3	gpio[12] configuration
D15:12	R/W	gpio11	0x3	gpio[11] configuration
D11:08	R/W	gpio10	0x3	gpio[10] configuration
D07:04	R/W	gpio9	0x3	gpio[9] configuration
D03:00	R/W	gpio8	0x3	gpio[8] configuration

Table 322: GPIO Configuration Register #2

GPIO Configuration Register #1

Address: 9060 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
gpio7				gpio6				gpio5				gpio4			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio3				gpio2				gpio1				gpio0			

Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio7	0x3	gpio[7] configuration
D27:24	R/W	gpio6	0x3	gpio[6] configuration
D23:20	R/W	gpio5	0x3	gpio[5] configuration
D19:16	R/W	gpio4	0x3	gpio[4] configuration
D15:12	R/W	gpio3	0x3	gpio[3] configuration
D11:08	R/W	gpio2	0x3	gpio[2] configuration
D07:04	R/W	gpio1	0x3	gpio[1] configuration
D03:00	R/W	gpio0	0x3	gpio[0] configuration

Table 323: GPIO Configuration Register #1

GPIO Configuration register options

Bits	Access	Mnemonic	Description
D03	R/W	PINd	0 Input 1 Output Controls the direction of the GPIO pin. All GPIO pins reset to the input state. In certain modes, the GPIO pin is bidirectional and controlled by the selected peripheral.
D02	N/A	Not used	Must write 0.

Table 324: GPIO Configuration register options

Bits	Access	Mnemonic	Description	
D01:00	R/W	PINn	00	Function #0
			01	Function #1
			10	Function #2
			11	Function #3
			Use these bits to select the function to use. See the discussion of GPIO MUX for details about the available pin functions.	

Table 324: GPIO Configuration register options

GPIO Control registers

GPIO Control Registers #1 and #2 contain the control information for each of the 50 GPIO pins in the NS9750, as shown in Table 325 and Table 326.

When a GPIO pin is configured as a GPIO output, the corresponding bit in GPIO Control Registers #1 and #2 is driven out the GPIO pin. In all configurations, the CPU has read/write access to the register.

GPIO Control Register#2

Address: 9060 0034

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														gpio 49	gpio 48

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio 47	gpio 46	gpio 45	gpio 44	gpio 43	gpio 42	gpio 41	gpio 40	gpio 39	gpio 38	gpio 37	gpio 36	gpio 35	gpio 34	gpio 33	gpio 32

Bits	Access	Mnemonic	Reset	Description
D31:18	N/A	Reserved	N/A	N/A
D17	R/W	gpio49	0	gpio[49] control bit
D16	R/W	gpio48	0	gpio[48] control bit

Table 325: GPIO Control Register #2

Bits	Access	Mnemonic	Reset	Description
D15	R/W	gpio47	0	gpio[47] control bit
D14	R/W	gpio46	0	gpio[46] control bit
D13	R/W	gpio45	0	gpio[45] control bit
D12	R/W	gpio44	0	gpio[44] control bit
D11	R/W	gpio43	0	gpio[43] control bit
D10	R/W	gpio42	0	gpio[42] control bit
D09	R/W	gpio41	0	gpio[41] control bit
D08	R/W	gpio40	0	gpio[40] control bit
D07	R/W	gpio39	0	gpio[39] control bit
D06	R/W	gpio38	0	gpio[38] control bit
D05	R/W	gpio37	0	gpio[37] control bit
D04	R/W	gpio36	0	gpio[36] control bit
D03	R/W	gpio35	0	gpio[35] control bit
D02	R/W	gpio34	0	gpio[34] control bit
D01	R/W	gpio33	0	gpio[33] control bit
D00	R/W	gpio32	0	gpio[32] control bit

Table 325: GPIO Control Register #2

GPIO Control Register #1

Address: 9060 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
gpio 31	gpio 30	gpio 29	gpio 28	gpio 27	gpio 26	gpio 25	gpio 24	gpio 23	gpio 22	gpio 21	gpio 20	gpio 19	gpio 18	gpio 17	gpio 16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio 15	gpio 14	gpio 13	gpio 12	gpio 11	gpio 10	gpio 9	gpio 8	gpio 7	gpio 6	gpio 5	gpio 4	gpio 3	gpio 2	gpio 1	gpio 0

Bits	Access	Mnemonic	Reset	Description
D31	R/W	gpio31	0	gpio[31] control bit
D30	R/W	gpio30	0	gpio[30] control bit
D29	R/W	gpio29	0	gpio[29] control bit
D28	R/W	gpio28	0	gpio[28] control bit
D27	R/W	gpio27	0	gpio[27] control bit
D26	R/W	gpio26	0	gpio[26] control bit
D25	R/W	gpio25	0	gpio[25] control bit
D24	R/W	gpio24	0	gpio[24] control bit
D23	R/W	gpio23	0	gpio[23] control bit
D22	R/W	gpio22	0	gpio[22] control bit
D21	R/W	gpio21	0	gpio[21] control bit
D20	R/W	gpio20	0	gpio[20] control bit
D19	R/W	gpio19	0	gpio[19] control bit
D18	R/W	gpio18	0	gpio[18] control bit
D17	R/W	gpio17	0	gpio[17] control bit
D16	R/W	gpio16	0	gpio[16] control bit
D15	R/W	gpio15	0	gpio[15] control bit
D14	R/W	gpio14	0	gpio[14] control bit
D13	R/W	gpio13	0	gpio[13] control bit
D12	R/W	gpio12	0	gpio[12] control bit
D11	R/W	gpio11	0	gpio[11] control bit
D10	R/W	gpio10	0	gpio[10] control bit
D09	R/W	gpio9	0	gpio[9] control bit
D08	R/W	gpio8	0	gpio[8] control bit
D07	R/W	gpio7	0	gpio[7] control bit
D06	R/W	gpio6	0	gpio[6] control bit

Table 326: GPIO Control Register #1

Bits	Access	Mnemonic	Reset	Description
D05	R/W	gpio5	0	gpio[5] control bit
D04	R/W	gpio4	0	gpio[4] control bit
D03	R/W	gpio3	0	gpio[3] control bit
D02	R/W	gpio2	0	gpio[2] control bit
D01	R/W	gpio1	0	gpio[1] control bit
D00	R/W	gpio0	0	gpio[0] control bit

Table 326: GPIO Control Register #1

GPIO Status registers

GPIO Status Registers #1 and #2 contain the status information for each of the 50 GPIO pins in the NS9750, as shown in Table 327 and Table 328. In all configurations, the value on the GPIO input pin is brought to the Status register and the CPU has *read-only* access to the register.

GPIO Status Register #2

Address: 9060 0044

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														gpio 49	gpio 48

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio 47	gpio 46	gpio 45	gpio 44	gpio 43	gpio 42	gpio 41	gpio 40	gpio 39	gpio 38	gpio 37	gpio 36	gpio 35	gpio 34	gpio 33	gpio 32

Note: The reset values for all of the status bits are undefined because they depend on the state of the GPIO pins to NS9750.

Bits	Access	Mnemonic	Reset	Description
D31:18	R	Not used	0x0	Always read as 0x0
D17	R	gpio49	undefined	gpio[49] status bit

Table 327: GPIO Status Register #2

Bits	Access	Mnemonic	Reset	Description
D16	R	gpio48	undefined	gpio[48] status bit
D15	R	gpio47	undefined	gpio[47] status bit
D14	R	gpio46	undefined	gpio[46] status bit
D13	R	gpio45	undefined	gpio[45] status bit
D12	R	gpio44	undefined	gpio[44] status bit
D11	R	gpio43	undefined	gpio[43] status bit
D10	R	gpio42	undefined	gpio[42] status bit
D09	R	gpio41	undefined	gpio[41] status bit
D08	R	gpio40	undefined	gpio[40] status bit
D07	R	gpio39	undefined	gpio[39] status bit
D06	R	gpio38	undefined	gpio[38] status bit
D05	R	gpio37	undefined	gpio[37] status bit
D04	R	gpio36	undefined	gpio[36] status bit
D03	R	gpio35	undefined	gpio[35] status bit
D02	R	gpio34	undefined	gpio[34] status bit
D01	R	gpio33	undefined	gpio[33] status bit
D00	R	gpio32	undefined	gpio[32] status bit

Table 327: GPIO Status Register #2

GPIO Status Register #1

Address: 9060 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
gpio 31	gpio 30	gpio 29	gpio 28	gpio 27	gpio 26	gpio 25	gpio 24	gpio 23	gpio 22	gpio 21	gpio 20	gpio 19	gpio 18	gpio 17	gpio 16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio 15	gpio 14	gpio 13	gpio 12	gpio 11	gpio 10	gpio 9	gpio 8	gpio 7	gpio 6	gpio 5	gpio 4	gpio 3	gpio 2	gpio 1	gpio 0

Note: The reset values for all of the status bits are undefined because they depend on the state of the GPIO pins to NS9750.

Bits	Access	Mnemonic	Reset	Description
D31	R	gpio31	undefined	gpio[31] status bit
D30	R	gpio30	undefined	gpio[30] status bit
D29	R	gpio29	undefined	gpio[29] status bit
D28	R	gpio28	undefined	gpio[28] status bit
D27	R	gpio27	undefined	gpio[27] status bit
D26	R	gpio26	undefined	gpio[26] status bit
D25	R	gpio25	undefined	gpio[25] status bit
D24	R	gpio24	undefined	gpio[24] status bit
D23	R	gpio23	undefined	gpio[23] status bit
D22	R	gpio22	undefined	gpio[22] status bit
D21	R	gpio21	undefined	gpio[21] status bit
D20	R	gpio20	undefined	gpio[20] status bit
D19	R	gpio19	undefined	gpio[19] status bit
D18	R	gpio18	undefined	gpio[18] status bit
D17	R	gpio17	undefined	gpio[17] status bit
D16	R	gpio16	undefined	gpio[16] status bit
D15	R	gpio15	undefined	gpio[15] status bit
D14	R	gpio14	undefined	gpio[14] status bit
D13	R	gpio13	undefined	gpio[13] status bit
D12	R	gpio12	undefined	gpio[12] status bit
D11	R	gpio11	undefined	gpio[11] status bit
D10	R	gpio10	undefined	gpio[10] status bit
D09	R	gpio9	undefined	gpio[9] status bit
D08	R	gpio8	undefined	gpio[8] status bit
D07	R	gpio7	undefined	gpio[7] status bit

Table 328: GPIO Status Register #1

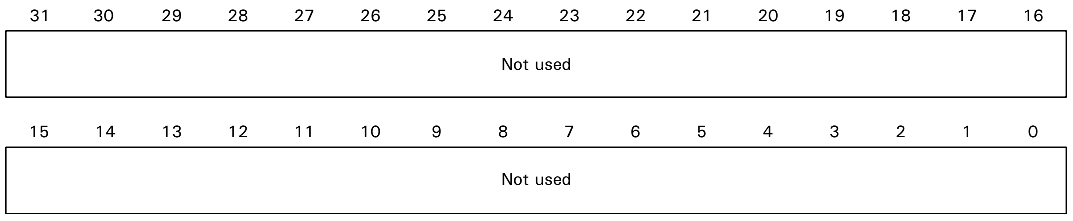
Bits	Access	Mnemonic	Reset	Description
D06	R	gpio6	undefined	gpio[6] status bit
D05	R	gpio5	undefined	gpio[5] status bit
D04	R	gpio4	undefined	gpio[4] status bit
D03	R	gpio3	undefined	gpio[3] status bit
D02	R	gpio2	undefined	gpio[2] status bit
D01	R	gpio1	undefined	gpio[1] status bit
D00	R	gpio0	undefined	gpio[0] status bit

Table 328: GPIO Status Register #1

BBus Monitor register

Address: 9060 0050

Write 0 to this register.



BBus DMA Interrupt Status register

Address: 9060 0060

The BBus DMA Interrupt Status register contains the interrupt status bits for the BBus DMA Controller. The interrupt bits are active high. Service these interrupts in the BBus DMA controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BINT 16	BINT 15	BINT 14	BINT 13	BINT 12	BINT 11	BINT 10	BINT 9	BINT 8	BINT 7	BINT 6	BINT 5	BINT 4	BINT 3	BINT 2	BINT 1

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	R	Not used	0x0	Always read as 0x0
D15	R	BINT16	0	BBus DMA channel #16 interrupt status
D14	R	BINT15	0	BBus DMA channel #15 interrupt status
D13	R	BINT14	0	BBus DMA channel #14 interrupt status
D12	R	BINT13	0	BBus DMA channel #13 interrupt status
D11	R	BINT12	0	BBus DMA channel #12 interrupt status
D10	R	BINT11	0	BBus DMA channel #11 interrupt status
D09	R	BINT10	0	BBus DMA channel #10 interrupt status
D08	R	BINT9	0	BBus DMA channel #9 interrupt status
D07	R	BINT8	0	BBus DMA channel #8 interrupt status
D06	R	BINT7	0	BBus DMA channel #7 interrupt status
D05	R	BINT6	0	BBus DMA channel #6 interrupt status
D04	R	BINT5	0	BBus DMA channel #5 interrupt status
D03	R	BINT4	0	BBus DMA channel #4 interrupt status

Table 329: BBus DMA Interrupt Status register

Bits	Access	Mnemonic	Reset	Description
D02	R	BINT3	0	BBus DMA channel #3 interrupt status
D01	R	BINT2	0	BBus DMA channel #2 interrupt status
D00	R	BINT1	0	BBus DMA channel #1 interrupt status

Table 329: BBus DMA Interrupt Status register

BBus DMA Interrupt Enable register

Address: 9060 0064

The BBus DMA Interrupt Enable register allows you to enable or disable the BBus DMA interrupts on an individual basis. Writing a 1 enables the interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BINT_EN16	BINT_EN15	BINT_EN14	BINT_EN13	BINT_EN12	BINT_EN11	BINT_EN10	BINT_EN9	BINT_EN8	BINT_EN7	BINT_EN6	BINT_EN5	BINT_EN4	BINT_EN3	BINT_EN2	BINT_EN1

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	R	Not used	0x0	Always read as 0x0
D15	R/W	BINT_EN16	0	BBus DMA channel #16 interrupt enable
D14	R/W	BINT_EN15	0	BBus DMA channel #15 interrupt enable
D13	R/W	BINT_EN14	0	BBus DMA channel #14 interrupt enable
D12	R/W	BINT_EN13	0	BBus DMA channel #13 interrupt enable
D11	R/W	BINT_EN12	0	BBus DMA channel #12 interrupt enable
D10	R/W	BINT_EN11	0	BBus DMA channel #11 interrupt enable
D09	R/W	BINT_EN10	0	BBus DMA channel #10 interrupt enable
D08	R/W	BINT_EN9	0	BBus DMA channel #9 interrupt enable

Table 330: BBus DMA Interrupt Enable register

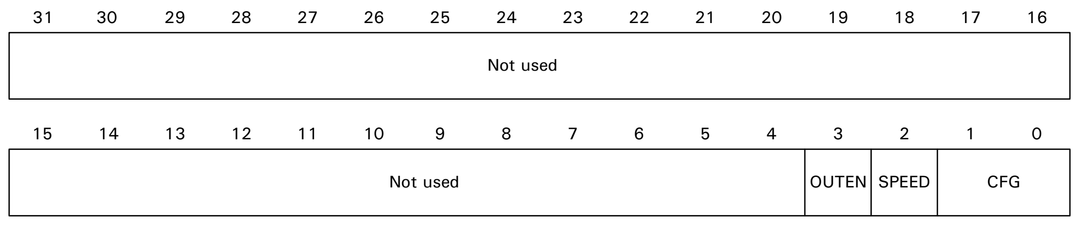
Bits	Access	Mnemonic	Reset	Description
D07	R/W	BINT_EN8	0	BBus DMA channel #8 interrupt enable
D06	R/W	BINT_EN7	0	BBus DMA channel #7 interrupt enable
D05	R/W	BINT_EN6	0	BBus DMA channel #6 interrupt enable
D04	R/W	BINT_EN5	0	BBus DMA channel #5 interrupt enable
D03	R/W	BINT_EN4	0	BBus DMA channel #4 interrupt enable
D02	R/W	BINT_EN3	0	BBus DMA channel #3 interrupt enable
D01	R/W	BINT_EN2	0	BBus DMA channel #2 interrupt enable
D00	R/W	BINT_EN1	0	BBus DMA channel #1 interrupt enable

Table 330: BBus DMA Interrupt Enable register

USB Configuration register

Address: 9060 0070

The USB Configuration register contains power-on USB configuration information. Write to this register only when the USB module is in reset, as indicated by the USB field in the Master Reset register (see page 523).



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	R	Not used	0x0	Always read as 0x0

Table 331: USB Configuration register

Bits	Access	Mnemonic	Reset	Description
D03	R/W	OUTEN	0	Enables the USB output driver during USB loopback testing. The output driver is enabled only when either the host or device indicates that it is driving the USB pins. Writing a 1 enables this feature.
D02	R/W	SPEED	1	0 Low speed (1.5 Mbps) 1 Full speed (12 Mbps) Defines the operational speed of the *USB device block*.
D01:00	R/W	CFG	11	**Configuration** 00 USB disabled 01 USB device mode; no software control 10 USB host mode; no software control 11 USB device mode; software control enabled Defines the operational mode of the USB module. This field can be modified only when the USB field in the Master Reset register is asserted (see page 523). For normal operation, this field should not need to be modified.

Table 331: USB Configuration register

Endian Configuration register

Address: 9060 0080

The Endian Configuration register contains the endian control for the BBus peripherals and the AHB bus master. NS9750 can be configured such that some peripherals transfer data in direct mode and some peripherals transfer data in DMA mode. Those that are accessed in direct mode must have their endian configuration match the AHB. The endian configuration of the AHB master must always match the AHB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used			AHBM	Reserved				I2C	IEEE 1284	SerD	SerC	SerA	SerB	USB	DMA

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:13	R	Not used	0x0	Always read as 0x0
D12	R/W	AHBM	Reset to the value provided on strapping pin gpio[44]	**AHB bus master** 0 Little endian 1 Big endian
D11:08	N/A	Reserved	N/A	N/A
D07	R/W	I2C	0	**I^{2}C controller** 0 Little endian 1 Big endian
D06	R/W	IEEE1284	0	**IEEE 1284 controller** 0 Little endian 1 Big endian
D05	R/W	SerD	0	**Serial controller port D** 0 Little endian 1 Big endian
D04	R/W	SerC	0	**Serial controller port C** 0 Little endian 1 Big endian
D03	R/W	SerA	0	**Serial controller port A** 0 Little endian 1 Big endian

Table 332: Endian Configuration register

Bits	Access	Mnemonic	Reset	Description
D02	R/W	SerB	0	**Serial controller port B** 0 Little endian 1 Big endian
D01	R/W	USB	0	**USB** 0 Little endian 1 Big endian This does not affect the USB DMA controller.
D00	R/W	DMA	Reset to the value provided on strapping pin gpio[44]	**BBus DMA** 0 Little endian 1 Big endian This field controls both the general BBus DMA controller and the USB DMA controller.

Table 332: Endian Configuration register

ARM Wake-up register

Address: 9060 0090

The ARM Wake-up register contains the ARM wake-up word used only by Serial Controller Interface #1. This pattern, when found as the next entry in the receive FIFO, causes a wake-up signal to be asserted to the ARM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WAKE															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAKE															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	WAKE	0x0000_0000	Defines the byte-wise match in order for the Serial Controller to signal a wake-up to the ARM.

Table 333: ARM Wake-up register

I^2C Master/Slave Interface

CHAPTER 11

The I^2C master/slave interface provides an interface between the ARM CPU and the I^2C bus.

The I^2C master/slave interface basically is a parallel-to-serial and serial-to-parallel converter. The parallel data received from the ARM CPU has to be converted to an appropriate serial form to be transmitted to an external component using the I^2C bus. Similarly, the serial data received from the I^2C bus has to be converted to an appropriate parallel form for the ARM CPU. The I^2C master interface also manages the interface timing, data structure, and error handling.

Overview

The I^{2}C module is designed to be a master and slave. The slave is active *only* when the module is being addressed during an I^{2}C bus transfer; the master can arbitrate for and access the I^{2}C bus *only* when the bus is free (idle) — therefore, the master and slave are mutually exclusive.

Physical I^{2}C bus

The physical I^{2}C bus consists of two open-drain signal lines: serial data (SDA) and serial clock (SCL). Pullup resistors are required; see the standard I^{2}C bus specification for the correct value for the application. Each device connected to the bus is software-addressable by a unique 7- or 10-bit address, and a simple master/slave relationship exists at all times.

A master can operate as a master-transmitter (writes)) or a master-receiver (reads). The slaves respond to the received commands accordingly:

- In transmit mode (slave is read), the host interface receives character-based parallel data from the ARM. The module converts the parallel data to serial format and transmits the serial data to the I^{2}C bus.
- In receive mode (slave is written to), the I^{2}C bus interface receives 8-bit-based serial data from the I^{2}C bus. The module converts the serial data to parallel format and interrupts the host. The host's interrupt service routine reads the parallel data from the data register inside the I^{2}C module. The serial data stream synchronization and throttling are done by modulating the serial clock. Serial clock modulation can be controlled by both the transmitter and receiver, based in their hosts' service speed.

The I^{2}C is a true multi-master bus with collision detection and arbitration to prevent data corruption when two or more masters initiate transfer simultaneously. If a master loses arbitration during the addressing stage, it is possible that the winning master is trying to address the transfer. The losing master must therefore immediately switch over to its slave mode.

The on-chip filtering rejects spikes on the bus data line to preserve data integrity. The number of ICs that can be connected to the same bus is limited only by a maximum bus capacity of 400 pf.

I^{2}C external addresses

I^{2}C external [bus] addresses are allocated as two groups of eight addresses (0000XXX and 1111XXX), as shown in Table 334.

Slave address	R/W bit	Description
0000 000	0	General call address
0000 000	1	START byte (not supported in NS9750)
0000 001	X	CBUS address (not supported in NS9750)
0000 010	X	Reserved for different bus format
0000 011	X	Reserved
0000 1xx	X	hs-mode master code (not supported in NS9750)
1111 1xx	X	Reserved
1111 0xx	X	10-bit slave address

Table 334: Reserved slave addresses

The general call address is for addressing all devices connected to the I^{2}C bus. A device can ignore this address by not issuing an acknowledgement. The meaning of the general call address is always specified in the second byte.

I^{2}C command interface

The I^{2}C module converts parallel (8-bit) data to serial data and serial data to parallel data between the NS9750 and the I^{2}C bus, using a set of interface registers.

- The primary interface register for transmitting data is the CMD_TX_DATA_REG (write-only).
- The primary interface register for receiving data is the STATUS_RX_DATA_REG (read-only).

Locked interrupt driven mode

I^{2}C operates in a *locked interrupt driven mode*, which means that each command issued must wait for an interrupt response before the next command can be issued (illustrated in "Flow charts," beginning on page 556).

The first bit of the command — 0 or 1 — indicates to which module — master or slave, respectively — the command in the CMD field (of the CMD_TX_DATA_REG; see page 548) is sent. The master module can be sent a master command only; the slave module can be sent a slave command only (see "Master and slave module commands," beginning on page 546, for a list of commands). If a command is sent to the master module, that module is locked until a command acknowledgement is given. Similarly, if a command is sent to the slave module, the slave module is locked until it receives a command acknowledgement. With either module, the acknowledgement can be any interrupt associated with that module. When a module is locked, another command must not be sent to that module.

The command lock status can be checked in the STATUS_RX_DATA_REG.

Master module and slave module commands

The I^{2}C master recognizes four high-level commands, which are used in the CMD field of the Command register (see page 548); the I^{2}C slave recognizes two high-level commands:

Command	Name	Description
00_{hex}	M_NOP	No operation.
04_{hex}	M_READ	Start reading bytes from slave.
05_{hex}	M_WRITE	Start writing bytes to slave.
06_{hex}	M_STOP	Stop this transaction (give up the I^{2}C bus).
10_{hex}	S_NOP	No operation. This command is necessary for 16-bit mode, providing data in TX_DATA_REG without a command.
16_{hex}	S_STOP	Stop transaction by not acknowledging the byte received.

Table 335: Master and slave module commands

Bus arbitration

Any M_READ or M_WRITE command causes the I^{2}C module to participate in the bus arbitration process when the I^{2}C bus is free (idle). If the module becomes the new bus owner, the transaction goes through. If the module loses bus arbitration, an M_ARBIT_LOST interrupt is generated to the host processor and the command must be reissued.

I^{2}C registers

All registers have 8-bit definitions, but must be accessed in pairs. For example, TX_DATA_REG and CMD_REG are written simultaneously and RX_DATA_REG and STATUS_REG are read simultaneously.

Table 336 shows the register addresses.

Register	Description
9050 0000	Command Transmit Data register (CMD_TX_DATA_REG) Status Receive Data register (STATUS_RX_DATA_REG)
9050 0004	Master Address register
9050 0008	Slave Address register
9050 000C	Configuration register

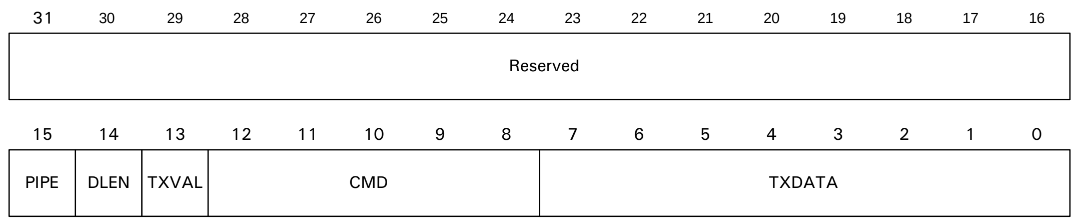
Table 336: I^{2}C register address map

After a reset, all registers are set to the initial value. If an unspecified register or bit is read, a zero is returned.

Command Transmit Data register

Address: 9050 0000

The Command Transmit Data (CMD_TX_DATA_REG) register is the primary interface register for transmission of data between the NS9750 BBus and I^{2}C bus. *This register is write only.*



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	**N/A**
D15	W	PIPE	00_{hex}	**Pipeline mode** Must be set to 0.
D14	W	DLEN	00_{hex}	**I^{2}C DLEN port** (iic_dlen) Must be set to 0.
D13	W	TXVAL	00_{hex}	**Provide new transmit data in** CMD_TX_DATA_REG (tx_data_val).
D12:08	W	CMD	00_{hex}	Command to be sent (see "Master and slave module commands," beginning on page 546)
D07:00	W	TXDATA	00_{hex}	Transmit data to I^{2}C bus.

Table 337: CMD_REG and TX_DATA_REG

Status Receive Data register

Address: 9050 0000

The Status Receive Data register (STATUS_RX_DATA_REG) is the primary interface register for receipt of data between the NS9750 BBus and I^2C bus. *This register is read only.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BSTS	RDE	SCMDL	MCMDL	IRQCD				RXDATA							

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	**N/A**
D15	R	BSTS	N/A	**Bus status (master only)** 0 Bus is free 1 Bus is occupied
D14	R	RDE	N/A	**Receive data enable (**rx_data_en**)** Received data is available.
D13	R	SCMDL	N/A	**Slave command lock** The Slave Command register is locked.
D12	R	MCMDL	N/A	**Master command lock** The Master Command register is locked.
D11:08	R	IRQCD	N/A	**Interrupt codes** (irq_code) The interrupt is cleared if this register is read. See "Interrupt Codes" on page 553 for more information.
D07:00	R	RXDATA	N/A	**Received data from I^2C bus** Together with a RX_DATA interrupt, this register provides a received byte (see Table 342: "Master/slave interrupt codes" on page 553).

Table 338: STATUS_REG and RX_DATA_REG

Master Address register

Address: 9050 0004

If using 7-bit addressing, the master device address field uses only bits D07:01; otherwise, all 10 bits are used.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					Master device address										Mstr addr mode

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D10:01	R/W	MDA	00_{hex}	**Master device address** Used for selecting a slave. ■ Represents bits 6:0 of the device address if using 7-bit address. D10:08 are not used. ■ Represents bits 9:0 of device address if using 10-bit address.
D00	R/W	MAM	00_{hex}	**Master addressing mode** 0 7 bit address mode 1 10 bit address mode

Table 339: Master Address register (7-bit and 10-bit)

Slave Address register

Address: 9050 0008

If using 7-bit addressing, the slave device address field uses only bits D07:01; otherwise, bits 10:01 are used.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				Gnrl call addr	Slave device address										Slave addr mode

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D11	R/W	GCA	00_{hex}	**General call address (**s_gca_irq_en**)** Enable the general call address. Default value is 1.
D10:01	R/W	SDA	00_{hex}	**Slave device address** Default value is $7F_{hex}$. ■ Represents bits 6:0 of device address if using 7-bit address; D10:08 are not used. ■ Represents bits 9:0 of device address if using 10-bit address.
D00	R/W	SAM	00_{hex}	**Slave addressing mode** 0 7 bit address mode 1 10 bit address mode

Table 340: Slave Address register (7-bit and 10-bit)

Configuration register

Address: 9050 000C

The Configuration register controls the timing on the I^2C bus. This register also controls the external interrupt indication, which can be disabled.

The I^2C bus clock timing is programmable by the scl_ref value (D08:00). The timing parameter for standard mode is as follows:

I^2C_bus_clock = clk / ((CLREF*2) + 4 + scl_delay)

clk = cpu_clk/4

Note: In noisy environments and fast-mode transmission, spike filtering can be applied to the received I^2C data and clock signal. The spike filter evaluates the incoming signal and suppresses spikes. The maximum length of the suppressed spikes can be specified in the spike filter width field of the Configuration register (see page 553).

The timing parameter for fast-mode is as follows:

I^2C_bus_clock = (4 / 3) x (clk / ((CLREF*2) + 4 + scl_delay))

scl_delay is influenced by the SCL rise time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

S

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQD	TMDE	VSCD	SFW				CLREF								

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A
D15	R/W	IRQD	0	**Mask the interrupt to the ARM CPU** (irq_dis) Must be set to 0.

Table 341: Configuration register

Bits	Access	Mnemonic	Reset	Description
D14	R/W	TMDE	1	**Timing characteristics of serial data and serial clock** 0 Standard mode 1 Fast mode
D13	R/W	VSCD	1	**Virtual system clock divider for master and slave** Must be set to 0.
D12:09	R/W	SFW	F_{hex}	**Spike filter width** A default value of 1 is recommended. Available values are 0–15.
D08:00	R/W	CLREF	00_{hex}	**clk_ref[9:1]** The I2C clock on port iic_scl_out is generated by the system clock divided by the 10-bit value of clk_ref. **Note:** The LSB of clk_ref cannot be programmed, and is set to 0 internally. The programmed value of clk_ref[9:1] must be greater than 3.

Table 341: Configuration register

Interrupt Codes

Interrupts are signaled in the irq_code field in the STATUS_REG, by providing the appropriate interrupt code (see Table 342: "Master/slave interrupt codes" on page 553). The ARM CPU waits for an interrupt by polling the STATUS_REG or checking the irq signal. An interrupt is cleared by reading the STATUS_REG, which also forces the irq signal down (minimum one cycle if another interrupt is stored).

Note: RX_DATA_REG contains only a received byte if it is accessed after a RX_DATA master or slave interrupt is signaled. At all other times, the internal master or slave shift register is accessed with RX_DATA_REG (see "Status Receive Data register" on page 549).

Code	Name	Master/slave	Description
0_{hex}	NO_IRQ	N/A	No interrupt active
1_{hex}	M_ARBIT_LOST	Master	Arbitration lost; the transfer has to be repeated

Table 342: Master/slave interrupt codes

Code	Name	Master/slave	Description
2_{hex}	M_NO_ACK	Master	No acknowledge by slave
3_{hex}	M_TX_DATA	Master	TX data required in register TX_DATA
4_{hex}	M_RX_DATA	Master	RX data available in register RX_DATA
5_{hex}	M_CMD_ACK	Master	Command acknowledge interrupt
6_{hex}	N/A	N/A	Reserved
7_{hex}	N/A	N/A	Reserved
8_{hex}	S_RX_ABORT	Slave	The transaction is aborted by the master before the slave performs a NO_ACK.
9_{hex}	S_CMD_REQ	Slave	Command request
A_{hex}	S_NO_ACK	Slave	No acknowledge by master (TX_DATA_REG is reset)
B_{hex}	S_TX_DATA_1ST	Slave	TX data required in register TX_DATA, first byte of transaction
C_{hex}	S_RX_DATA_1ST	Slave	RX data available in register RX_DATA, first byte of transaction
D_{hex}	S_TX_DATA	Slave	TX data required in register TX_DATA
E_{hex}	S_RX_DATA	Slave	RX data available in register RX_DATA
F_{hex}	S_GCA	Slave	General call address

Table 342: Master/slave interrupt codes

Software driver

The I^{2}C master software driver uses three commands only:

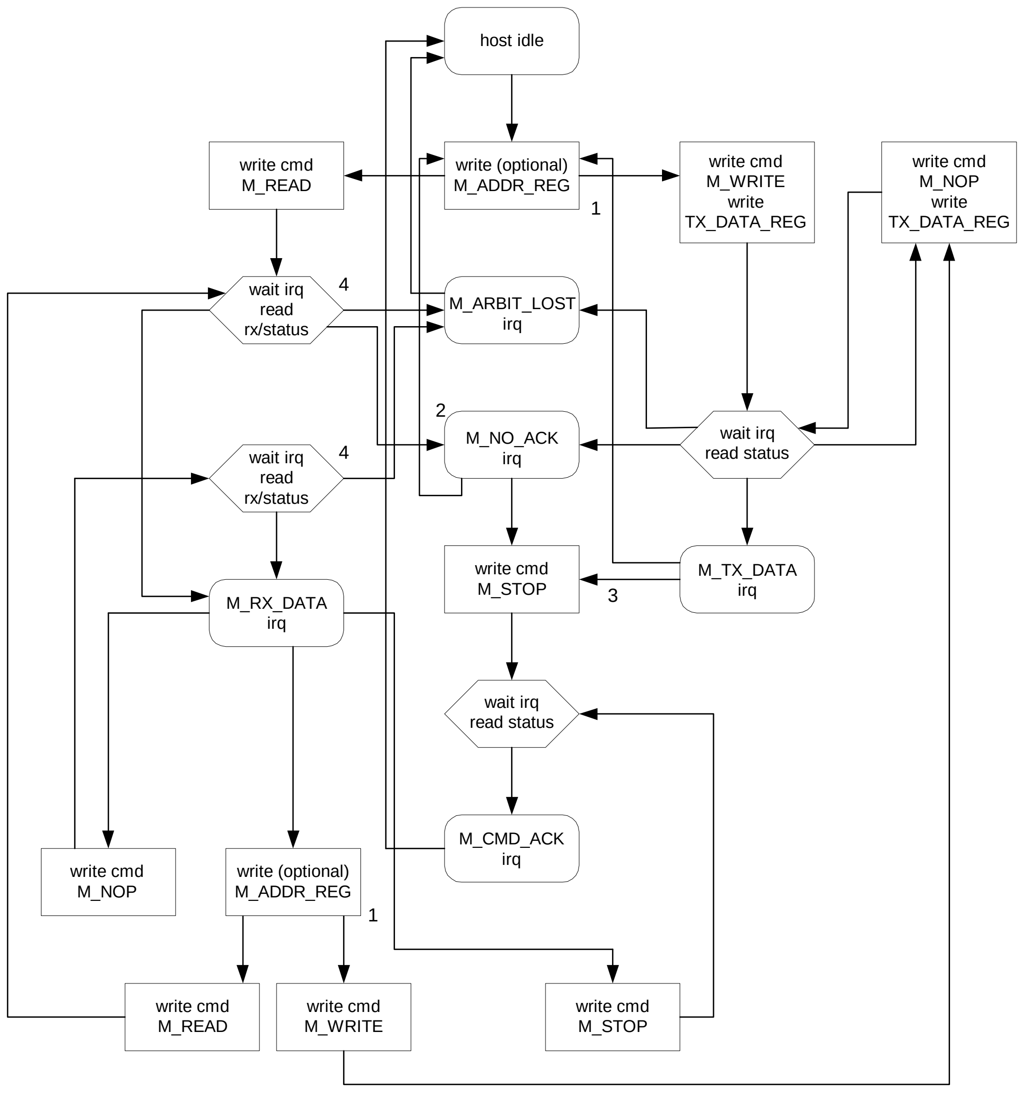
- M_READ to start a read sequence
- M_WRITE to start a write sequence
- M_STOP to give up the I^{2}C bus

If, during a read or write sequence, another M_READ or M_WRITE is requested by the ARM CPU, a restart is performed on the I^{2}C bus. This opens the opportunity to provide a new slave device address in the MAster Address register before the command request.

The I^{2}C slave high level driver identifies one command: S_STOP, to discontinue a transaction. After this command, the slave remains inactive until the next start condition on the I^{2}C bus. If a slave is accessed by a master, it generates S_RX_DATA and S_TX_DATA interrupts (see "Master/slave interrupt codes" on page 553). To distinguish the transactions from each other, special S_RX_DATA_1ST and S_TX_DATA_1ST interrupts are generated for the transmitted byte.

Flow charts

Master module (normal mode, 16-bit)



Notes:

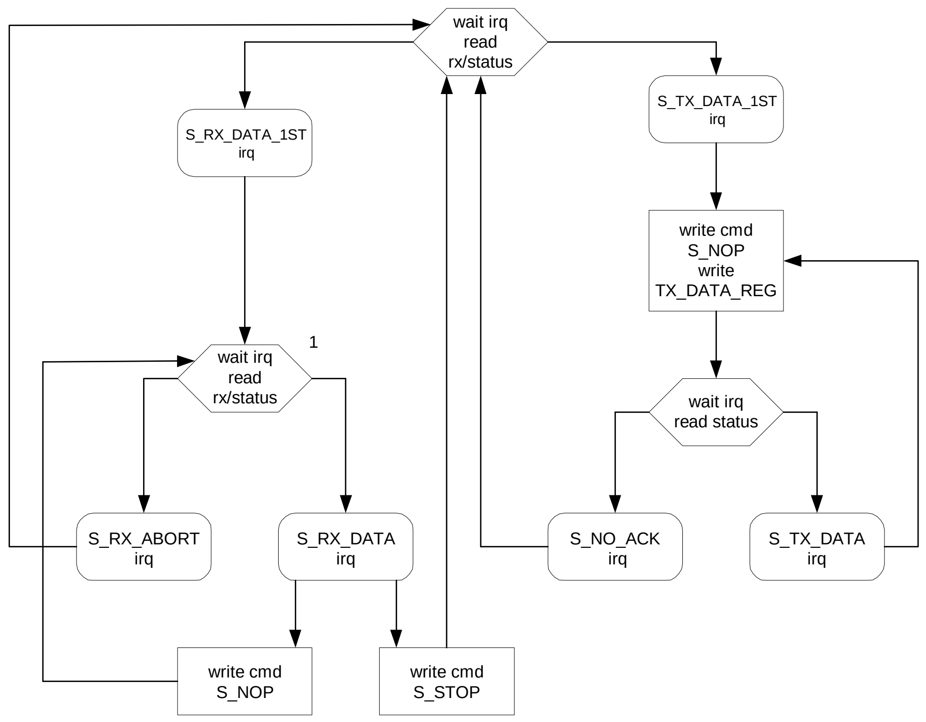
1 Writing M_ADDR_REQ is not required if the device address is not changed.

2 Read on a non-existing slave.

3 Do not wait for the slave to perform a NO_ACK.

4 STATUS_REG and RX_DATA_REG are read simultaneously.

Slave module (normal mode, 16-bit)



Note:

1 STATUS_REG and RX_DATA_REG are read simultaneously.

LCD Controller

C H A P T E R 1 2

The NS9750 LCD (Liquid Crystal Display) controller is a DMA master module that connects to the AHB bus. The LCD controller provides the signals required to interface directly to TFT and STN color and monochrome LCD panels.

LCD controller timing diagrams can be found in the Timing chapter.

LCD features

The NS9750 LCD controller provides these features:

- Dual 64-deep, 32-bit wide FIFOs, for buffering incoming display data
- Support for color and monochrome single- and dual-panel for Super Twisted Nematic (STN) displays with 4- or 8-bit interfaces
- Support for Thin Film Transistor (TFT) color displays
- Resolution programmable up to 1024 x 768
- 15 gray-level mono, 3375 color STN, and 64K color TFT support
  - Patented gray-scale algorithm
- 1, 2, or 4 bits-per-pixel (bpp) palettized displays for mono STN
- 1, 2, 4, or 8 bpp palettized color displays for color STN and TFT
- 16 bpp true-color non-palettized, for color STN and TFT
- 24 bpp true-color non-palettized for TFT
- Programmable timing for different display panels
- 256 entry, 16-bit palette RAM, arranged as a 128 x 32-bit RAM
- Frame, line, and pixel clock signals
- AC bias signal for STN, data enable signal for TFT panels
- Support for little and big endian, as well as Windows CE data formats

Programmable parameters

These key parameters are programmable:

- Horizontal front and back porch
- Horizontal synchronization pulse width
- Number of pixels per line
- Vertical front and back porch
- Vertical synchronization pulse width
- Number of lines per panel
- Number of panel clocks per line

- Signal polarity, active high or low
- AC panel bias
- Panel clock frequency
- Bits-per-pixel
- Display type, STN mono/color or TFT
- STN 4- or 8-bit interface mode
- STN dual- or single-panel mode
- Little endian, big endian, or WinCE mode
- Interrupt generation event

LCD panel resolution

The LCD can be programmed to support a wide range of panel resolutions, including but not limited to:

- 320 x 200, 320 x 240
- 640 x 200, 640 x 240, 640 x 480
- 800 x 600
- 1024 x 768

LCD panel support

The LCD controller supports these types of panels:

- Active matrix TFT panels with up to 24-bit bus interface
- Single-panel monochrome STN panels (4-bit and 8-bit bus interface)
- Dual-panel monochrome STN panels (4-bit and 8-bit bus interface per panel)
- Single-panel color STN panels, 8-bit bus interface
- Dual-panel color STN panels, 8-bit bus interface per panel

Number of colors

The number of colors supported differs per panel type.

TFT panels

TFT panels support one or more of these color modes:

- 1 bpp, palettized, 2 colors selected from available colors
- 2 bpp, palettized, 4 colors selected from available colors
- 4 bpp, palettized, 16 colors selected from available colors
- 8 bpp, palettized, 256 colors selected from available colors
- 16 bpp, direct 5:5:5 RGB, with one bpp usually not used. This pixel is still output, and can be used as a *bright* bit to connect to the least significant bit (lsb) of R, G, and B components of a 6:6:6 TFT panel.
- 24 bpp, direct 8:8:8 RGB, providing over 16 million colors

Each 16-bit palette entry is made up of five bpp (RGB) plus a common intensity bit, which provides better memory use and performance compared with a full six bpp structure. The total amount of colors supported can be doubled from 32K to 64K if the intensity bit is used and applied to all three color components simultaneously.

Color STN panels

Color STN panels support one or more of these color modes:

- 1 bpp, palettized, 2 colors selected from 3375
- 2 bpp, palettized, 4 colors selected from 3375
- 4 bpp, palettized, 16 colors selected from 3375
- 8 bpp, palettized, 256 colors selected from 3375
- 16 bpp, direct 4:4:4 RGB, with 4 bpp not used

Mono STN panels

Mono STN panels support one or more of these modes:

- 1 bpp, palettized, 2 grayscales selected from 15
- 2 bpp, palettized, 4 grayscales selected from 15
- 4 bpp, palettized, 15 grayscales selected from 15

LCD power up and power down sequence support

This procedure provides an example of how the LCD controller can be programmed to provide the powerup sequence to an LCD panel (see Figure 85, "Power up and power down sequences," on page 564):

1. V_{DD} is applied simultaneously to the NS9750 and panel display driver logic. The following signals are pulled up to V_{DD} until the LCD controller is configured: CLLP, CLCP, CLFP, CLAC, CLD[23:0], and CLLE.
2. After the LCD controller is configured, a 1 is written to the LcdEn bit in the LCDControl register. This enables the CLLP, CLCP, CLFP, CLAC, and CLLE signals, but the CLD[23:0] signals will be low.
3. When the signals in Step 2 have stabilized, the contrast voltage, V_{EE} (which is not controlled or supplied by the LCD controller), is applied where appropriate. If required, a software timer routine can be used to provide the minimum display specific delay time between application of V_{DD} and application of V_{EE}.
4. If required, a software timer routine can be used to provide the minimum display specific delay time between application of the control signals and power to the panel display. When the software timer routine completes, power is applied to the panel by writing a 1 to the LcdPwr bit in the LcdControl register, which, in turn, sets the CLPOWER signal high and enables the CLD[23:0] signals into their active state. The CLPOWER signal gates the power to the LCD panel.

The power down sequence is the reverse of the powerup procedure, with the respective register bits written to 0 rather than 1.

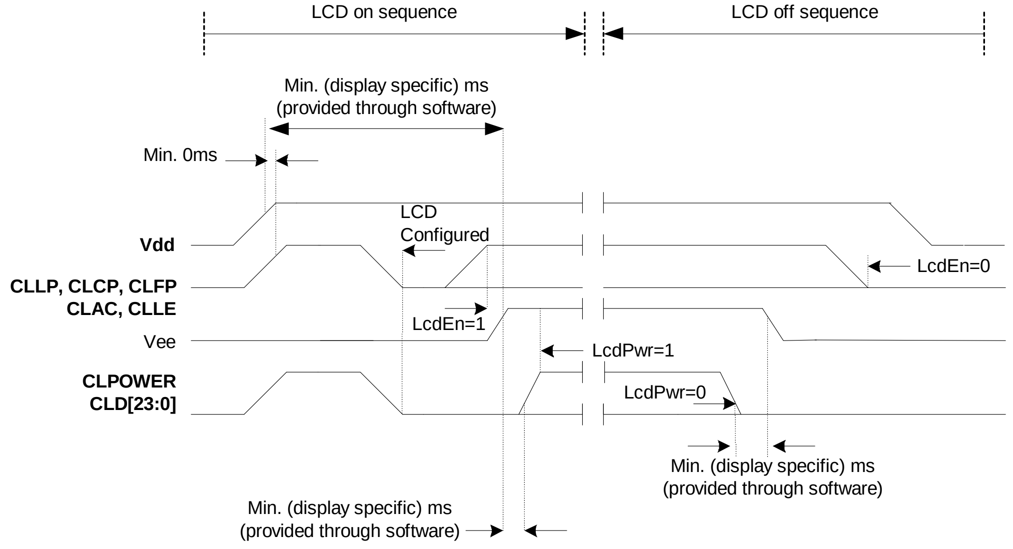

LCD on sequence
LCD off sequence
Min. (display specific) ms
(provided through software)
Min. 0ms
LCD
Configured
Vdd
CLLP, CLCP, CLFP
CLAC, CLLE
Vee
CLPOWER
CLD[23:0]
LcdEn=1
LcdPwr=1
LcdEn=0
LcdPwr=0
Min. (display specific) ms
(provided through software)
Min. (display specific) ms
(provided through software)

Figure 85: Power up and power down sequences

LCD controller functional overview

The LCD controller translates pixel-coded data into the required formats and timing to drive a variety of single and dual mono and color LCDs.

The controller supports passive STN and active TFT LCD display types.

- STN display panels require algorithmic pixel pattern generation to provide pseudo-grayscaling on mono displays or color creation on color displays.
- TFT display panels require the digital color value of each pixel to be applied to the display data units.

Packets of pixel-coded data are fed, through the AHB interface, to two independent, 64-deep, 32-bit wide DMA FIFOs, which act as input data flow buffers. The buffered pixel-coded data is then unpacked using a pixel serializer.

Depending on the LCD type and mode, the unpacked data can represent one of the following:

- An actual true display gray or color value
- An address to a 256 x 16 bit wide palette RAM gray or color value

With STN displays, either a value obtained from the addressed palette location or the true value is passed to the grayscaling generators. The hardware-coded grayscale algorithm logic sequences the addressed pixels activity over a programmed number of frames to provide the proper display appearance.

With TFT displays, either an addressed palette value or true color value is passed directly to the output display drivers, bypassing the grayscaling algorithm logic.

Clocks

The NS9750 LCD controller requires separate AHB (HCLK) and LCD (CLCDCLK) input clocks. The source of CLCDCLK is programmable using the LCD panel select field in the Clock Configuration register. Table 343 shows the clock selections.

LCD panel clock select	CLCDCLK
000	HCLK
001	HCLK/2
010	HCLK/4
011	HCLK/8
1xx	lcdclk/2 **Note:** lcdclk is an external clock input to NS9750. A *divided-by-2* version of this value is sent to the LCD controller.

Table 343: CLCDCLK selection

The LCD controller uses CLCDCLK internally. The clock sent to the LCD panel (CLCP) normally is derived from CLCDCLK using the PCD (panel clock divisor) value in the LCDTiming2 register (see page 586). The LCD controller also can bypass the internal clock divider controlled by PCD and use CLCDCLK as CLCP directly by setting the BCD (bypass pixel clock divider) bit to 1 in the LCDTiming2 register (see page 584).

Signals and interrupts

The LCD controller provides a set of programmable display control signals, and generates individual interrupts for different conditions.

Programmable control signals

- LCD power panel enable
- Pixel clock
- Horizontal and vertical synchronization pulses
- Display bias

Individual interrupts

- Base address update signification
- Vertical compare
- Bus error

There is also a single combined interrupt that is generated when any of the individual interrupts become active.

Figure 86 shows the LCD controller module.

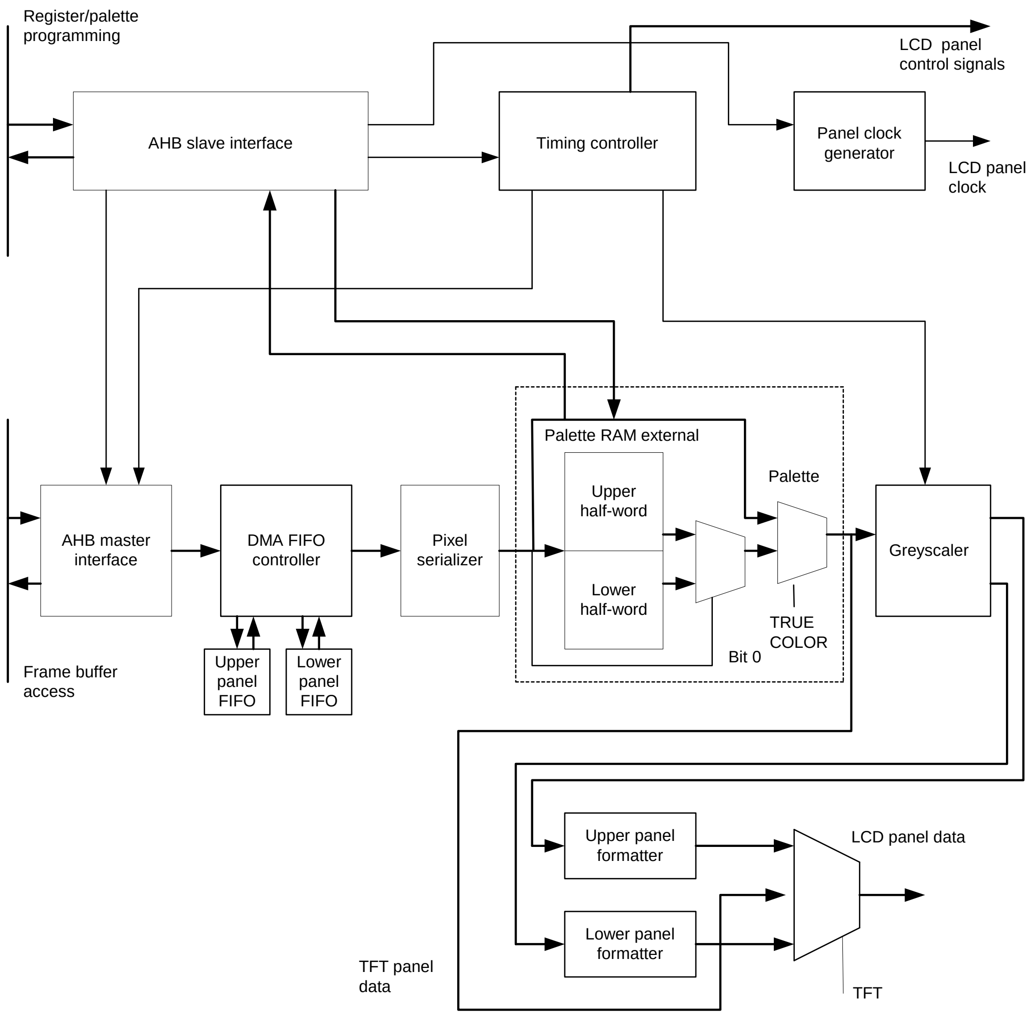


Figure 86: LCD controller block diagram

AHB interface

The AHB interface includes the AHB slave interface and the AHB master interface.

AHB master and slave interfaces

The AHB master interface transfers display data from memory to the LCD controller DMA FIFOs.

The AHB slave interface connects the LCD to the AHB bus and provides CPU accesses to the registers and palette RAM. The LCD controller AHB slave interface supports these features:

Dual DMA FIFOs and associated control logic

The pixel data accessed from memory is buffered by two DMA FIFOs, which can be controlled independently to cover single and dual panel LCD types. Each FIFO is 64 words deep by 32 bits wide, and can be cascaded to form a 128-word deep FIFO in single panel mode. The FIFO input ports are connected to the AHB interface and the output port feeds the pixel serializer.

Synchronization logic is used to transfer the pixel data from the AHB HCLK domain to the CLCDCLK clock domain. The DMA FIFOs are clocked by HCLK.

The water level marks within each FIFO are set such that each FIFO requests data when at least four locations become available.

Pixel serializer

The pixel serializer block reads the 32-bit wide LCD data from DMA FIFO output port, and extracts 24, 16, 8, 4, 2, or 1 bpp, depending on the current mode of operation. The LCD controller supports big endian, little endian, and WinCE data formats. In dual panel mode, data is read alternately from the upper and lower DMA FIFOs. The mode of operation determines whether the extracted data is used to point to a color/ grayscale value in the palette ram or is actually a true color value that can be applied directly to an LCD panel input.

The next six figures show the data structure in each DMA FIFO word corresponding to the Endianness and bpp combinations. For each of the three supported data formats, the required data for each panel display pixel must be extracted from the data word.

Figure 87 and Figure 88 show the data structure for little endian byte, little endian pixel — LBLP.

Figure 89 and Figure 90 show the data structure for big endian byte, big endian pixel — BBBP.

Figure 91 and Figure 92 show the data structure for little endian byte, big endian pixel — LBBP. (This is WinCE format.)

bpp	DMA FIFO OUTPUT BITS															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	p31	p30	p29	p28	p27	p26	p25	p24	p23	p22	p21	p20	p19	p18	p17	p16
2	p15		p14		p13		p12		p11		p10		p9		p8	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p7				p6				p5				p4			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p3								p2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
									23	22	21	20	19	18	17	16

Figure 87: LBLP, DMA FIFO output bits 31:16

bpp	DMA FIFO OUTPUT BITS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	p15	p14	p13	p12	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0
2	p7		p6		p5		p4		p3		p2		p1		p0	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p3				p2				p1				p0			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p1								p0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 88: LBLP, DMA FIFO output bits 15:0

bpp	DMA FIFO OUTPUT BITS															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15
2	p0		p1		p2		p3		p4		p5		p6		p7	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p0				p1				p2				p3			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p0								p1							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
									23	22	21	20	19	18	17	16

Figure 89: BBBP, DMA FIFO output bits 31:16

bpp	DMA FIFO OUTPUT BITS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	p16	p17	p18	p19	p29	p21	p22	p23	p24	p25	p26	p27	p28	p29	p30	p31
2	p8		p9		p10		p11		p12		p13		p14		p15	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p4				p5				p6				p7			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p2								p3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 90: BBBP, DMA FIFO output bits 15:0

bpp	DMA FIFO OUTPUT BITS															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	p24	p25	p26	p27	p28	p29	p30	p31	p16	p17	p18	p19	p20	p21	p22	p23
2	p12		p13		p14		p15		p8		p9		p10		p11	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p6				p7				p4				p5			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p3								p2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
									23	22	21	20	19	18	17	16

Figure 91: LBBP, DMA FIFO output bits 31:16

bpp	DMA FIFO OUTPUT BITS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	p8	p9	p10	p11	p12	p13	p14	p15	p0	p1	p2	p3	p4	p5	p6	p7
2	p4		p5		p6		p7		p0		p1		p2		p3	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p2				p3				p0				p1			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p1								p0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 92: LBBP, DMA FIFO output bits 15:0

RAM palette

The palette RAM is a 256 x 16 bit dual port RAM, physically structured as 128 x 32 bit. This allows two entries to be written into the palette from a single word write access. The least significant bit of the serialized pixel data selects between upper and lower halves of the palette RAM. The half selected depends on the byte-ordering mode. In little endian mode, setting the least significant bit selects the upper half of the palette; in big endian mode, setting the least significant bit selects the lower half. Because WinCE byte ordering is little endian, setting the least significant byte results in selection of the upper half of the palette.

Pixel data values can be written and verified using the slave interface.

The palette RAM has independent controls and addresses for each port.

- Port1 is used as a read/write port, and is connected to the AHB slave interface. The palette entries can be written and verified through this port.
- Port2 is used as a read-only port, and is connected to the unpacker and grayscaler.

Table 344 shows the bit representation of each word in the palette.

Bit	Name	Description
31	I	Intensity/unused
30:26	B[4:0]	Blue palette data
25:20	G[4:0]	Green palette data
19:16	R[4:0]	Red palette data
15	I	Intensity/unused
14:10	B[4:0]	Blue palette data
09:05	G[4:0]	Green palette data
04:00	R[4:0]	Red palette data

Table 344: Palette data storage

For mono STN, only the red palette field bits (4:1) are used. In STN color mode, however, the green and blue [4:1] are also used.

Red and blue pixel data can be swapped to support BGR data format using the appropriate control register bit.

In 16- and 24-bpp TFT mode, the palette is bypassed and the pixel serializer output is used as the TFT panel data.

Grayscaler

A unique grayscale algorithm drives mono and color STN panels.

- For mono displays, the algorithm provides 15 grayscales.
- For STN color displays, the three color components (red, green, and blue) are grayscaled simultaneously, resulting in 3375 (15 x 15 x 15) colors available. The grayscaler transforms each 4-bit gray value into a sequence of activity-per-pixel over several frames, relying somewhat on the display characteristics, to give representation of grayscales and color.

Upper and lower panel formatters

Each formatter consists of three 3-bit (red, green, and blue) *shift left* registers. Red, green, and blue pixel data bit values from the grayscaler are shifted concurrently into the respective registers. When enough data is available, a byte is constructed by multiplexing the registered data to the correct bit position to satisfy the RGB data pattern of the LCD panel. The byte is transferred to the 3-byte FIFO, which has enough space to store eight color pixels.

Panel clock generator

The panel clock generator block output is the panel clock. This is a divided down version of CLCDCLK, and can be programmed in the range CLCDCLK/2 to CLCDCLK/33 to match the bpp data rate of the LCD panel.

Timing controller

The timing controller block's primary function is to generate the horizontal and vertical timing panel signals. The timing controller also provides the panel bias/enable signal. Use the AHB slave interface to program these signals in the appropriate registers.

Generating interrupts

The LCD controller has three individually masked interrupts and a single combined interrupt. The single combined interrupt is asserted if any of the combined interrupts are asserted and unmasked.

External pad interface signals

The external pad interface signals are brought out through GPIO.

Signal name	Type	Description
CLPOWER	Output	LCD panel power enable
CLLP	Output	Line synchronization pulse (STN)/horizontal synchronization pulse (TFT)
CLCP	Output	LCD panel clock
CLFP	Output	Frame pulse (STN)/vertical synchronization pulse (TFT)
CLAC	Output	STN AC bias drive or TFT data enable output
CLD[23:0]	Output	LCD panel data
CLLE	Output	Line end signal

Table 345: External pad interface signals

LCD panel signal multiplexing details

The CLLP, CLAC, CLFP, and CLLE signals are common, but the CLD[23:0] bus has eight modes of operation:

- TFT 24-bit interface
- TFT 18-bit interface
- Color STN single panel
- Color STN dual panel
- 4-bit mono STN single panel
- 4-bit mono STN dual panel
- 8-bit mono STN single panel
- 8-bit mono STN dual panel

Table 346 shows which CLD[23:0] pins provide the pixel data to the STN panel for each mode of operation. The abbreviations used in the table are defined as follows:

- CUSTN = Color upper panel STN, dual and/or single panel
- CLSTN = Color lower panel STN, dual
- MUSTN = Mono upper panel STN, dual and/or single panel
- MLSTN = Mono lower panel STN, dual

Ext pin	GPIO pin & description	Color STN single panel	Color STN dual panel	4-bit mono STN single panel	4-bit mono STN dual panel	8-bit mono STN single panel	8-bit mono STN dual panel
CLD[23]	AA4=LCD data bit 23 (O2)	N/A	N/A	N/A	N/A	N/A	N/A
CLD[22]	AB3=LCD data bit 22 (O2)	N/A	N/A	N/A	N/A	N/A	N/A
CLD[21]	AE1=LCD data bit 21 (O2)	N/A	N/A	N/A	N/A	N/A	N/A
CLD[20]	AD2=LCD data bit 20 (O2)	N/A	N/A	N/A	N/A	N/A	N/A
CLD[19]	AF3=LCD data bit 19 (O2)	N/A	N/A	N/A	N/A	N/A	N/A
CLD[18]	AE4=LCD data bit 18 (O2)	N/A	N/A	N/A	N/A	N/A	N/A
CLD[17]	AD5=LCD data bit 17 (O2)	N/A	N/A	N/A	N/A	N/A	N/A
CLD[16]	AC6=LCD data bit16 (O2)	N/A	N/A	N/A	N/A	N/A	N/A
CLD[15]	AF4=LCD data bit 15 (O2)	N/A	CLSTN[0][1]	N/A	N/A	N/A	MLSTN[0][1]
CLD[14]	AE5=LCD data bit 14 (O2)	N/A	CLSTN[1]	N/A	N/A	N/A	MLSTN[1]
CLD[13]	AD6=LCD data bit 13 (O2)	N/A	CLSTN[2]	N/A	N/A	N/A	MLSTN[2]
CLD[12]	AF5=LCD data bit 12 (O2)	N/A	CLSTN[3]	N/A	N/A	N/A	MLSTN[3]
CLD[11]	AE6=LCD data bit 11 (O2) AE8=LCD data bit 11 (O2)	N/A	CLSTN[4]	N/A	MLSTN[0][1]	N/A	MLSTN[4]
CLD[10]	AD7=LCD data bit 10 (O2) AD9=LCD data bit 10 (O2)	N/A	CLSTN[5]	N/A	MLSTN[1]	N/A	MLSTN[5]
CLD[9]	AD8=LCD data bit 9 (O2) AF8=LCD data bit 9 (O2)	N/A	CLSTN[6]	N/A	MLSTN[2]	N/A	MLSTN[6]
CLD[8]	AF7=LCD data bit 8 (O2) AE9=LCD data bit 8 (O2)	N/A	CLSTN[7]	N/A	MLSTN[3]	N/A	MLSTN[7]

Table 346: LCD STN panel signal multiplexing

Ext pin	GPIO pin & description	Color STN single panel	Color STN dual panel	4-bit mono STN single panel	4-bit mono STN dual panel	8-bit mono STN single panel	8-bit mono STN dual panel
CLD[7]	AE8=LCD data bit 7 (O1)	CUSTN[0][1]	CUSTN[0][1]	N/A	N/A	MUSTN[0]	MUSTN[0][1]
CLD[6]	AD9=LCD data bit 6 (O1)	CUSTN[1]	CUSTN[1]	N/A	N/A	MUSTN[1]	MUSTN[1]
CLD[5]	AF8=LCD data bit 5 (O1)	CUSTN[2]	CUSTN[2]	N/A	N/A	MUSTN[2]	MUSTN[2]
CLD[4]	AE9=LCD data bit 4 (O1)	CUSTN[3]	CUSTN[3]	N/A	N/A	MUSTN[3]	MUSTN[3]
CLD[3]	AF9=LCD data bit 3 (O1)	CUSTN[4]	CUSTN[4]	MUSTN[0]	MUSTN[0][1]	MUSTN[4]	MUSTN[4]
CLD[2]	AD10=LCD data bit 2 (O1)	CUSTN[5]	CUSTN[5]	MUSTN[1]	MUSTN[1]	MUSTN[5]	MUSTN[5]
CLD[1]	AE10=LCD data bit 1 (O1)	CUSTN[6]	CUSTN[6]	MUSTN[2]	MUSTN[2]	MUSTN[6]	MUSTN[6]
CLD[0]	AF10=LCD data bit 0 (O1)	CUSTN[7]	CUSTN[7]	MUSTN[3]	MUSTN[3]	MUSTN[7]	MUSTN[7]

1 This data bit corresponds to the first “pixel position.” For example, for an 8-bit mono STN display, CUSTN[0] is the leftmost pixel on the panel and CUSTN[7] is the rightmost pixel within the 8-bit data. For a color STN display, bits [7, 6, 5] form the leftmost pixel.

Table 346: LCD STN panel signal multiplexing

Table 347 shows which CLD[23:0] pins are used to provide the pixel data to the TFT panel for each mode of operation.

External pin	TFT 24 bit	TFT 15 bit
CLD[23]	Blue[7]	Reserved
CLD[22]	Blue[6]	Reserved
CLD[21]	Blue[5]	Reserved
CLD[20]	Blue[4]	Reserved
CLD[19]	Blue[3]	Reserved
CLD[18]	Blue[2]	Reserved
CLD[17]	Blue[1]	Blue[4]
CLD[16]	Blue[0]	Blue[3]

Table 347: LCD TFT panel signal multiplexing

External pin	TFT 24 bit	TFT 15 bit
CLD[15]	Green[7]	Blue[2]
CLD[14]	Green[6]	Blue[1]
CLD[13]	Green[5]	Blue[0]
CLD[12]	Green[4]	Intensity bit
CLD[11]	Green[3]	Green[4]
CLD[10]	Green[2]	Green[3]
CLD[9]	Green[1]	Green[2]
CLD[8]	Green[0]	Green[1]
CLD[7]	Red[7]	Green[0]
CLD[6]	Red[6]	Intensity bit
CLD[5]	Red[5]	Red[4]
CLD[4]	Red[4]	Red[3]
CLD[3]	Red[3]	Red[2]
CLD[2]	Red[2]	Red[1]
CLD[1]	Red[1]	Red[0]
CLD[0]	Red[0]	Intensity bit

Table 347: LCD TFT panel signal multiplexing

This LCD TFT panel signal multiplexing table shows the RGB alignment to a 15-bit TFT with the intensity bit not used. The intensity bit, if used, should be connected to the LSB (that is, RED[0], GREEN[0], BLUE[0]) input of an 18-bit LCD TFT panel as shown in the next table.

RGB[x]:	4	3	2	1	0	Intensity
18-bit TFT	5	4	3	2	1	0
15-bit TFT	4	3	2	1	0	x
12-bit TFT	3	2	1	0	x	x
9-bit TFT	2	1	0	x	x	x

Table 348: RGB bit alignment according to TFT interface size (one color shown)

If you want reduced resolution, the least significant color bits can be dropped, starting with Red[0], Green[0], and Blue[0].

Registers

Table 349 lists the LCD controller registers. All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Address	Register	Description
A080 0000	LCDTiming0	Horizontal axis panel control
A080 0004	LCDTiming1	Vertical axis panel control
A080 0008	LCDTiming2	Clock and signal polarity control
A080 000C	LCDTiming3	Line end control
A080 0010	LCDUPBASE	Upper panel frame base address
A080 0014	LCDLPBASE	Lower panel frame base address
A080 0018	LCDINTRENABLE	Interrupt enable mask
A080 001C	LCDControl	LCD panel pixel parameters
A080 0020	LCDStatus	Raw interrupt status
A080 0024	LCDInterrupt	Final masked interrupts
A080 0028	LCDUPCURR	LCD upper panel current address value
A080 002C	LCDLPCURR	LCD lower panel current address value
A080 0030 – A080 01FC	Reserved	Reserved
A080 0200 – A080 03FC	LCDPalette	256 x 16-bit color palette

Table 349: LCD registers

LCDTiming0

Address: A080 0000

The LCDTiming0 register controls the horizontal axis panel, which includes:

- Horizontal synchronization pulse width (HSW)
- Horizontal front porch (HFP) period
- Horizontal back porch (HBP) period
- Pixels-per-line (PPL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HBP								HFP							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSW								PPL						Reserved	

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:24	R/W	HBP	0x00	**Horizontal back porch** Number of CLCP periods between the negation of CLLP and the start of active data. Program this field with value minus 1. HBP specifies the number of pixel clock periods inserted at the beginning of each line or row of pixels. HBP can generate a delay of 1 to 256 pixel clock cycles.
D23:16	R/W	HFP	0x00	**Horizontal front porch** Number of CLCP periods between the end of active data and the assertion of CLLP. Program this field with value minus 1. HFP sets the number of pixel clock periods at the end of each line or row of pixels, before CLLP is asserted. HFP can generate a period of 1 to 256 pixel clock cycles.

Table 350: LCDTiming0 register

Bits	Access	Mnemonic	Reset	Description
D15:08	R/W	HSW	0x00	**Horizontal synchronization pulse width** Width of the CLLP signal in CLCP periods. Program this field with value minus 1. HSW specifies the pulse width of the line clock in passive mode, or the horizontal synchronization pulse in active mode.
D07:02	R/W	PPL	0x00	**Pixels-per-line** Actual pixels-per-line = 16 * (PPL+1) The PPL field specifies the number of pixels in each line or row of the screen. PPL is a 6-bit value that represents between 16 and 1024 PPL. PPL counts the number of pixel clocks that occur before HFP is applied (program the value required divided by 16, minus 1).
D01:00	N/A	Reserved	N/A	N/A

Table 350: LCDTiming0 register

Horizontal timing restrictions

DMA requests new data at the beginning of a horizontal display line. Some time must be allowed for the DMA transfer and for the data to propagate down the FIFO path in the LCD interface. The data path latency forces some restrictions on the usable minimum values for horizontal porch width in STN mode. The minimum values are HSW = 2 and HBP = 2. Table 351 shows the recommended minimum values for STN displays:

Mode	HSW	HBP	HFP	Panel clock divisor (PCD)
Single panel STN mode	3	5	5	1 (CLCDCLK/3)
Dual panel STN mode	3	5	5	5 (CLCDCLK/7)

Table 351: Minimum recommended values for STN displays

If sufficient time is given at the beginning of the line (for example, HSW is set to 6 and HBP is set to 10), data will not become corrupted for PCD = 4 (minimum value for dual panel mode).

LCDTiming1

Address: A080 0004

The LCDTiming1 register controls the vertical axis panel, which includes:

- Number of lines-per-panel (LPP)
- Vertical synchronization pulse width (VSW)
- Vertical front porch (VFP) period
- Vertical back porch (VBP period)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VBP								VFP							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VSW						LPP									

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:24	R/W	VBP	0x00	**Vertical back porch** Number of inactive lines at the start of a frame, after vertical synchronization period. Program this field to zero on passive displays, to avoid reduced contrast. VBP specifies the number of line clocks inserted at the beginning of each frame. The VBP count starts just after the CLFP for the previous frame has been negated for active mode, or the extra horizontal synchronization lines have been asserted as specified by the VSW field in passive mode. After this occurs, the count value in VBP sets the number of horizontal synchronization lines inserted before the next frame. VBP generates from 0 –255 extra line clock cycles.

Table 352: LCDTiming1 register

Bits	Access	Mnemonic	Reset	Description
D23:16	R/W	VFP	0x00	**Vertical front porch** Number of inactive lines at the end of the frame, before vertical synchronization period. Program this field to zero on passive displays, to avoid reduced contrast. VFP specifies the number of blank lines to insert at the end of each frame. Once a complete frame of pixels is transmitted to the LCD display, the value in VFP counts the number of horizontal synchronization lines to wait. After the count has elapsed, the vertical synchronization (CLFP) signal is asserted in active mode, or extra line clocks are inserted as specified by the VSW field in passive mode. VFP generates from 0 – 255 CLLP cycles.
D15:10	R/W	VSW	0x00	**Vertical synchronization pulse width** Number of horizontal synchronization lines. This value must be small (for example, program to 0) for passive STN LCDs. Program this field to the number of lines required minus one. The higher the value, the worse the contrast on STN LCDs. VSW specifies the pulse width of the vertical synchronization pulse. This field is programmed to the number of horizontal synchronization lines minus one.
D09:00	R/W	LPP	0x000	**Lines per panel** Number of active lines per screen. Program this field to number of lines required minus 1. LPP specifies the total number of lines or rows on the LCD panel being controlled; between 1 and 1024 lines are allowed. This field is programmed with the number of lines per LCD panel minus 1. For dual panel displays, this field is programmed with the number of lines on each of the upper and lower panels.

Table 352: LCDTiming1 register

LCDTiming2 register

Address: A080 0008

The LCDTiming2 register provides controls for the timing signals.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					BCD	CPL									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd	IOE	IPC	IHS	IVS	ACB					Rsvd	PCD				

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:27	N/A	Reserved	N/A	N/A
D26	R/W	BCD	0x0	**Bypass pixel clock divider** Set this field to 1 to bypass the pixel clock divider logic. Used primarily for TFT displays.
D25:16	R/W	CPL	0x000	**Clocks per line** Specifies the number of actual CLCP clocks to the LCD panel on each line. This is the number of pixels-per-line divided by 1 (TFT), 4 or 8 (mono STN), or 2 2/3 (color STN), minus one. Be sure this value is programmed properly, in addition to PPL; otherwise, the LCD controller does not work correctly.
D15	N/A	Reserved	N/A	N/A
D14	R/W	IOE	0x0	**Invert output enable** 0 CLAC output pin is active high in TFT mode 1 CLAC output pin is active low in TFT mode Selects the active polarity of the output enable signal in TFT mode. In this mode, the CLAC pin is used as an enable that indicates to the LCD panel when valid display data is available. In TFT mode, data is driven onto the LCD data lines at the programmed edge of CLCP when CLAC is in its active state.

Table 353: LCDTiming2 register

Bits	Access	Mnemonic	Reset	Description
D13	R/W	IPC	0x0	**Invert panel clock** 0 Data changes on the rising edge of CLCP. 1 Data changes on the falling edge of CLCP Controls the phasing of the LCD data relative to the LCD clock (CLCP). The NS9750 changes the data on the opposite edge of the clock used to capture the data.
D12	R/W	IHS	0x0	**Invert horizontal synchronization** 0 CLLP pin is active high and inactive low 1 CLLP pin is active low and inactive high Inverts the polarity of the CLLP signal.
D11	R/W	IVS	0x0	**Invert vertical synchronization** 0 CLFP pin is active high and inactive low 1 CLFP pin is active low and inactive high Inverts the polarity of the CLFP signal.
D10:06	R/W	ACB	0x00	**AC bias pin frequency** Applies only to STN displays, which require the pixel voltage polarity to be reversed periodically to prevent damage due to DC charge accumulation. Program this field with the required value minus one, to apply the number of line clocks between each toggle of the AC bias pin (CLAC). This field has no effect when the LCD controller is using TFT mode, which uses the CLAC pin as a data enable signal.
D05	N/A	Reserved	N/A	N/A

Table 353: LCDTiming2 register

Bits	Access	Mnemonic	Reset	Description
D04:00	R/W	PCD	0x00	**Panel clock divisor** Derives the LCD panel clock frequency CLCP from the CLCDCLK frequency: CLCP = CLCDCLK/(PCD+2) ■ For mono STN displays with a 4- or 8-bit interface, the panel clock is a factor of four and eight down on the actual individual pixel clock rate. ■ For color STN displays, 2 2/3 pixels are output per CLCP cycle, resulting in a panel clock of 0.375 times. ■ For TFT displays, the pixel clock divider can be bypassed by setting the LCDTiming2 BCD bit (D26). See "Panel clock divider restrictions" on page 586 for more information.

Table 353: LCDTiming2 register

Panel clock divider restrictions

The data path latency forces some restrictions on the usable minimum values for the panel clock divider in STN modes:

- Single panel color mode: PCD = 1 (CLCP = CLCDCLK/3)
- Dual panel color mode: PCD = 4 (CLCP = CLCDCLK/6)
- Single panel mono 4-bit interface mode: PCD = 2 (CLCP = CLCDCLK/4)
- Dual panel mono 4-bit interface mode: PCD = 6 (CLCP = CLCDCLK/8)
- Single panel mono 8-bit interface mode: PCD = 6 (CLCP = CLCDCLK/8)
- Dual panel mono 8-bit interface mode: PCD = 14 (CLCP = CLCDCLK/16)

LCDTiming3

Address: A080 000C

LCDTiming3 controls whether the line-end signal, CLLE, is enabled. When enabled, a positive pulse, four CLCDCLK periods wide, is output on CLLE after a programmable delay from the last pixel of each display line. If the line-end signal is disabled, it is held permanently low.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															LEE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									LED						

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:17	N/A	Reserved	N/A	N/A
D16	R/W	LEE	0x0	**LCD line-end enable** 0 CLLE disabled (held low) 1 CLLE signal active
D15:07	N/A	Reserved	N/A	N/A
D06:00	R/W	LED	0x00	**Line-end signal delay** Line-end signal delay from the rising edge of the last panel clock (CLCP). Program this field with number of CLCDCLK clock periods minus 1.

Table 354: LCDTiming3 register

LCDUPBASE and LCDLPBASE

Address: A080 0010 and A080 0014

LCDUPBASE and LCDLPBASE are the DMA base address registers, and program the base address of the frame buffer.

LCDUPBASE is used for these displays:

- TFT
- Single panel STN
- Upper panel of dual panel STN

LCDLPBASE is used for the lower panel of dual panel STN displays.

Important: You must initialize LCDUPBASE (and LCDLPBASE for dual panels) before enabling the LCD controller.

The base address value optionally can be changed mid-frame, to allow double-buffered video displays to be created. These registers are copied to the corresponding current registers at each LCD vertical synchronization, which then cause the LNBU bit (in the LCDStatus register; see page 593) to be set and an optional interrupt to be generated. The interrupt can be used to reprogram the base address when generating double-buffered video.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCDUPBASE															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCDUPBASE														Reserved	

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:02	R/W	LCDUPBASE	0x00000000	**LCD upper panel base address** Starting address of the upper panel frame data in memory; the address is word-aligned.
D01:00	N/A	Reserved	N/A	N/A

Table 355: LCDUPBASE register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCDLPBASE															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCDLPBASE														Reserved	

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:02	R/W	LCDLPBASE	0x00000000	**LCD lower panel base address** Starting address of the lower panel frame data in memory; the address is word-aligned.
D01:00	R/W	Not used	0x0	Read as 0.

Table 356: LCDLPBASE register

LCDINTRENABLE

Address: A080 0018

LCDINTRENABLE is the interrupt enable register. Setting bits within this register enables the corresponding raw interrupt LCDStatus bits to cause an interrupt to the system.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											MBERR INTR ENB	VCOMP INTR ENB	LNBU INTR ENB	Not used	

Register bit assignment

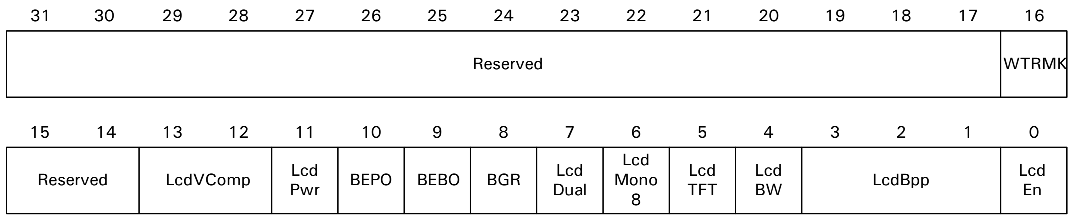
Bits	Access	Mnemonic	Reset	Description
D31:05	N/A	Reserved	N/A	N/A
D04	R/W	MBERRINTRENB	0x0	AHB master bus error interrupt enable.
D03	R/W	VCOMPINTRENB	0x0	Vertical compare interrupt enable.
D02	R/W	LNBUINTRENB	0x0	Next base update interrupt enable.
D01:00	N/A	Not used	0x0	Always write 0.

Table 357: LCDINTRENABLE register

LCDControl register

Address: A080 001C

The LCDControl register controls the mode in which the LCD controller operates.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:17	N/A	Reserved	N/A	N/A

Table 358: LCDControl register

Bits	Access	Mnemonic	Reset	Description
D16	R/W	WATERMARK	0x0	**LCD DMA FIFO watermark level** 0 LCD controller requests AHB bus when either of the DMA FIFOs have at least four empty locations. 1 LCD controller requests AHB bus when either of the DMA FIFOs have at least eight empty locations. (Use this setting for optimum bus bandwidth.)
D15:14	N/A	Reserved	N/A	N/A
D13:12	R/W	LcdVComp	0x0	Generate vertical compare interrupt (VCOMP; see "LCDStatus register" on page 593) at one of the following: 00 Start of vertical synchronization 01 Start of back porch 10 Start of active video 11 Start of front porch
D11	R/W	LcdPwr	0x0	**LCD power enable** 0 Power not gated through to LCD panel and CLD[23:0] signals disabled (held low). 1 Power gated through to LCD panel and CLD[23:0] signals enabled (active). See "LCD power up and power down sequence support" on page 563 for additional information.
D10	R/W	BEPO	0x0	**Big endian pixel ordering within a byte** 0 Little endian pixel ordering within a byte. 1 Big endian pixel ordering within a byte. The BEPO bit selects between Little and Big endian pixel packing for 1, 2, and 4 bpp display modes; the bit has no effect on 8 or 16 bpp pixel formats. See "Pixel serializer" on page 569 for additional information.
D09	R/W	BEBO	0x0	**Big endian byte order** 0 Little endian byte order 1 Big endian byte order
D08	R/W	BGR	0x0	**RGB or BGR format selection** 0 RGB: Normal output 1 BGR: Red and blue swapped

Table 358: LCDControl register

Bits	Access	Mnemonic	Reset	Description
D07	R/W	LcdDual	0x0	**LCD interface is dual panel STN** 0 Single panel LCD is in use. 1 Dual panel LCD is in use.
D06	R/W	LcdMono8	0x0	**Monochrome LCD has 8-bit interface** 0 Mono LCD uses 4-bit interface. 1 Mono LCD uses 8-bit interface. Controls whether monochrome STN LCD uses a 4- or 8-bit parallel interface. Program this bit to 0 for other modes.
D05	R/W	LcdTFT	0x0	**LCD is TFT** 0 LCD is STN display; use grayscaler. 1 LCD is TFT; do *not* use grayscaler.
D04	R/W	LcdBW	0x0	**STN LCD is monochrome (black and white)** 0 STN LCD is color 1 STN LCD is monochrome This bit has no meaning in TFT mode.
D03:01	R/W	LcdBpp	0x00	**LCD bits per pixel** 000 1 bpp 001 2 bpp 010 4 bpp 011 8 bpp 100 16 bpp 101 24 bpp (TFT panel only) 110 Reserved 111 Reserved
D00	R/W	LcdEn	0x0	**LCD controller enable** 0 LCD signals CLLP, CLCP, CLFP, CLAC, and CLLE disabled (held low). 1 LCD signals CLLP, CLCP, CLFP, CLAC, and CLLE enabled (active). See "LCD power up and power down sequence support" on page 563 for more information about LCD power sequencing.

Table 358: LCDControl register

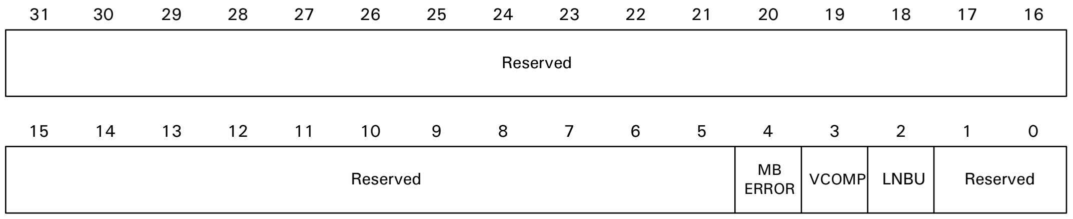
LCDStatus register

Address: A080 0020

The LCDStatus register provides raw interrupt status.

- On a read, the register returns three bits that can generate interrupts when set.
- On writes to the register, a bit value of 1 clears the interrupt corresponding to that bit. Writing a 0 has no effect.

Note: R/C indicates an access of read or clear.



Register bit assignment

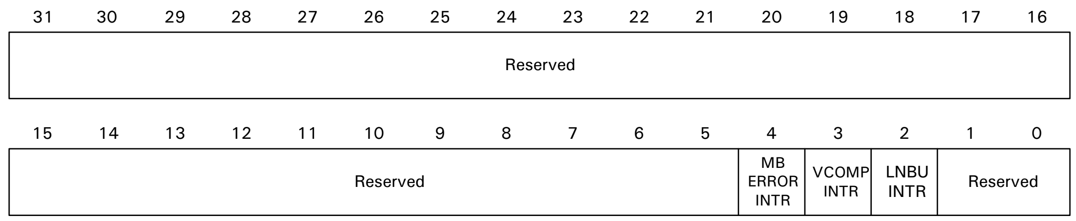
Bits	Access	Mnemonic	Reset	Description
D31:05	N/A	Reserved	N/A	N/A
D04	R/C	MBERROR	0x0	**AHB master bus error status** Set when the AHB master encounters a bus error response from a slave.
D03	R/C	VCOMP	0x0	**Vertical compare** Set when one of the four vertical regions, selected by the LCDControl register, is reached. (See LcdVcomp in "LCDControl register" on page 590).
D02	R/C	LNBU	0x0	**LCD next address base update** This bit is mode-dependent, and is set when the current base address registers have been updated successfully by the next address registers. Signifies that a new next address can be loaded if double buffering is in use.
D01:00	N/A	Reserved	0x0	N/A

Table 359: LCDStatus register

LCDInterrupt register

Address: A080 0024

The LCDInterrupt register is a bit-by-bit logical AND of the LCDStatus register and the LCDINTRENABLE register. Interrupt lines correspond to each interrupt. A logical OR of all interrupts is provided to the system interrupt controller.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:05	N/A	Reserved	N/A	N/A
D04	R	MBERRORINTR	0x0	AHB master bus error interrupt status bit.
D03	R	VCOMPINTR	0x0	Vertical compare interrupt status bit.
D02	R	LNBUINTR	0x0	LCD next base address update interrupt status bit.
D01:00	N/A	Reserved	N/A	N/A

Table 360: LCDInterrupt register

LCDUPCURR and LCDLPCURR

Address: A080 0028 and A080 002C

The LCDUPCURR and LCDLPCURR registers contain an approximate value of the upper and lower panel data DMA addresses when read. The registers can change at any time, and therefore can be used only as a mechanism for coarse delay.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCDUPCURR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCDUPCURR															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R	LCDUPCURR	X	LCD upper panel current address value.

Table 361: LCDUPCURR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCDLPCURR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCDLPCURR															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R	LCDLPCURR	X	LCD lower panel current address value.

Table 362: LCDLPCURR register

LCDPalette register

Address A080 0200 – 03FC

LCDPalette registers contain 256 palette entries organized as 128 locations of two entries per word.

Only TFT displays use all of the palette entry bits.

Each word location contains two palette entries, which means that 128 word locations are used for the palette.

- When configured for little endian byte ordering, bits [15:00] are the lower-numbered palette entry and bits [31:16] are the higher-numbered palette entry.
- When configured for big endian byte ordering, bits [31:16] are the lower-numbered palette entry and bits [15:00] are the higher-numbered palette entry.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Int0	B[4:0]					G[4:0]					R[4:0]				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Int1	B[4:0]					G[4:0]					R[4:0]				

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	Int0	N/A	**Intensity/unused** Can be used as the least significant bit of the R, G, and B inputs to a 6:6:6 TFT display, doubling the number of colors to 64K, where each color has two different intensities.
D30:26	R/W	B[4:0]	N/A	**Blue palette data** ■ For STN color displays, only the four most significant bits (04:01) are used. ■ Not used for monochrome displays.
D25:21	R/W	G[4:0]	N/A	**Green palette data** ■ For STN color displays, only the four most significant bits (04:01) are used. ■ Not used for monochrome displays.

Table 363: LCDPalette register

Bits	Access	Mnemonic	Reset	Description
D20:16	R/W	R[4:0]	N/A	**Red palette data** ■ For STN color displays, only the four most significant bits (04:01) are used. ■ Used for monochrome displays.
D15	R/W	Int1	N/A	**Intensity bit** Can be used as the least significant bit of the R, G, and B inputs to a 6:6:6 TFT display, doubling the number of colors to 64K, where each color has two different intensities.
D14:10	R/W	B[4:0]	N/A	**Blue palette data** ■ For STN color displays, only the four most significant bits (04:01) are used. ■ Not used for monochrome displays.
D09:05	R/W	G[4:0]	N/A	**Green palette data** ■ For STN color displays, only the four most significant bits (04:01) are used. ■ Not used for monochrome displays.
D04:00	R/W	R[4:0]	N/A	**Red palette data** ■ For STN color displays, only the four most significant bits (04:01) are used. ■ Used for monochrome displays.

Table 363: LCDPalette register

Interrupts

The LCD controller drives a single interrupt back to the system, from four interrupt sources.

Each of the three individual maskable interrupt sources is enabled or disabled by changing the mask bits in the LCDINTRENABLE register. The status of the individual interrupt sources can be read from the LCDStatus register.

The interrupt sources are described next.

MBERRORINTR — Master bus error interrupt

The master bus error interrupt is asserted when an error response is received by the master interface during a transaction with a slave. When such an error occurs, the master interface enters an error state and remains in this state until the error is cleared (error clearance has been signalled to the master). When the respective interrupt service routine has completed, the master bus error interrupt can be cleared by writing a 1 to the MBERROR bit in the LCDStatus register. This action releases the master interface from its error state to the start of the frame state, allowing a fresh frame of data display to be initiated.

VCOMPINTR — Vertical compare interrupt

The vertical compare interrupt is asserted when one of the four vertical display regions, selected using the LCDControl register, is reached. The interrupt can occur at the beginning of one of the following:

- Vertical synchronization
- Back porch
- Active video
- Front porch

This interrupt can be cleared by writing a 1 to the Vcomp bit in the LCDStatus register.

LBUINTR — Next base address update interrupt

The LCD next base address update interrupt is asserted when either the LCDUPBASE or LCDLPBASE values have been transferred to the LCDUPCURR or LCDLPCURR incrementers (respectively). This tells the system that it is safe to update the LCDUPBASE or LCDLPBASE registers with new frame base addresses, if required.

This interrupt can be cleared by writing a 1 to the LNBU bit in the LCDStatus register.

Serial Control Module: UART

CHAPTER 13

The NS9750 ASIC supports four independent universal asynchronous/synchronous receiver/transmitter channels. Each channel supports several modes, conditions, and formats.

Features

Each channel supports these features:

- DMA transfers to and from system memory
- Independent programmable bit-rate generator
- High speed data transfer: 1.8432 Mbps (asynchronous)
- 32-byte TX FIFO
- 32-Byte RX FIFO
- Programmable data formats
  - 5 to 8 data bits
  - Odd, even, or no parity
  - 1, 2 stop bits
- Programmable channel modes
  - Normal
  - Loopback
  - Remote loopback
  - Control signal support: RTS, CTS, DTR, DSR, DCD, RI
- Maskable interrupt conditions
  - Receive FIFO ready
  - Receive FIFO half full
  - Transmit FIFO ready
  - Transmit FIFO half empty
  - CTS, DSR, DCD, RI state change detection
- Multi-drop capable

Figure 93 shows the structure of the serial module.

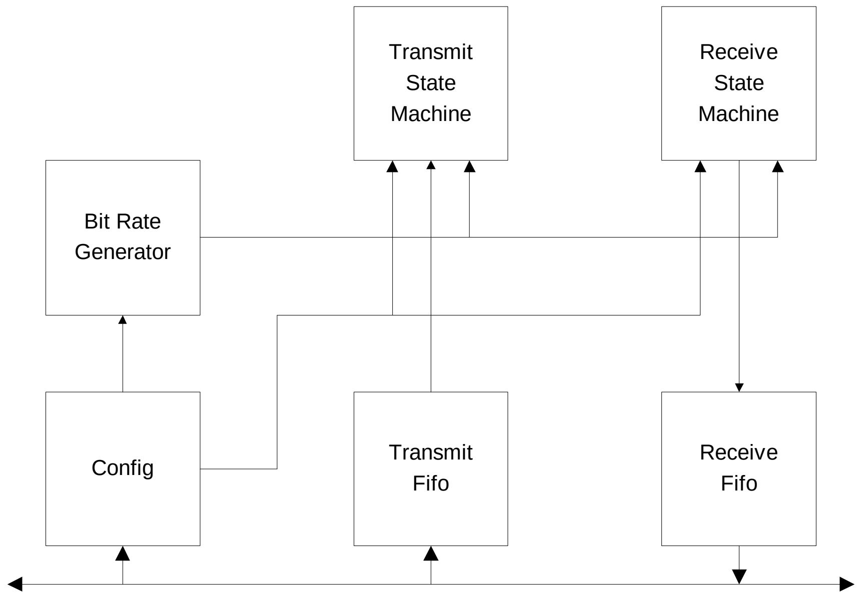


Figure 93: Serial Module structure

Bit-rate generator

Each serial channel supports an independent programmable bit-rate generator. The bit-rate generator runs both the transmitter and receiver of a given channel (there is no split speed support).

You can configure the bit-rate generator to use external clock input or internal system timing as its timing reference. This allows for a wider range of possible bit-rates.

Table 364 describes all possible clock reference sources used by the bit-rate generator.

Name	Description
X1_SYS_OSC/M	The frequency of the external crystal oscillator divided by 2 or 4. ■ The divisor is 2 when the PLLND field in the PLL Configuration register is at least 0x13, producing a nominal frequency of 14.7456 MHz. ■ The divisor is 4 when PLLND is less than 0x13, producing a nominal frequency of 7.3728 MHz. See the System Control Module chapter for information about the PLL Configuration register.
BCLK	The clock source for all peripherals that are attached to the BBus. The frequency of BCLK is the AHB clock frequency divided by 2.
ExtRxClk	External receive clock on GPIO pins gpio[6], gpio[14], gpio[22], and gpio[26] for serial ports B, A, C, and D, respectively.
ExtTxClk	External transmit clock on GPIO pins gpio[7], gpio[15], gpio[23], and gpio[27] for serial ports B, A, C, and D, respectively.

Table 364: Bit-rate generation clock sources

UART mode

Many applications require a simple mechanism for sending low-speed information between two pieces of equipment. The universal asynchronous/synchronous receiver/transmitter (UART) protocol is the de facto standard for simple serial communications. The protocol does not require sending clock information between the two parties; rather, the UART receiver uses an over-sampling technique to find the bit-level framing of the UART protocol. The UART framing structure is as follows:

Start bit:	0
Data bits:	5, 6, 7, or 8
Parity:	Odd, even, or no parity
Stop bits:	1 or 2

Because the transmitter and receiver operate asynchronously, there is no need to connect transmit and receive clocks between them. Instead, the receiver over-samples the receive data stream by a factor of 16. During synchronization, the receiver waits for the start bit. When it finds the high-to-low transition, the receiver counts 8 sample times and uses this point as the bit-center for all remaining bits in the UART frame. Each bit-time is 16 clock ticks apart.

When the UART is not transmitting data, it transmits a continuous stream of ones — referred to as the *IDLE* condition. When data transmission begins, the transmitter sends the start bit and the receiver is enabled.

You can configure the UART to perform the following functions:

- Enable the transmitter using the CTS handshaking signal. In this mode, the transmitter cannot start a new UART data frame unless CTS is active. If CTS is dropped anywhere in the middle of a UART data frame, the current character is completed and the next character is stalled.
- Signal its receiver FIFO status using the RTS handshaking signal. When the receive FIFO has only four characters of available space, the RTS signal is dropped. The RTS and CTS pairs can be used for hardware flow control.

FIFO management

Data flow between a serial controller and memory occurs through the FIFO blocks within each serial controller module. Each serial controller provides both a 32-byte transmit FIFO and a 32-byte receive FIFO. Each FIFO is arranged as eight lines of four bytes to facilitate data transfer across BBus. Both the transmit and receive FIFOs are accessed using the Serial Channel B/A/C/D FIFO registers.

Transmit FIFO interface

The processor can write either 1, 2, 3, or 4 bytes at a time to the transmit FIFO. The number of bytes written is controlled by the data size defined by the HSIZE field on the AMBA AHB bus.

- When the system is configured to operate in big endian mode, the most significant bytes in the word written to the FIFO are transmitted first. For example, the long word 0x11223344 results in the character 0x11 being transmitted first, and 0x44 being transmitted last.

- When the system is configured to operate in little endian mode, the least significant bytes in the word written to the FIFO are transmitted first. For example, the long word 0x11223344 results in the character 0x44 being transmitted first, and 0x11 being transmitted last.

Processor interrupts vs. DMA

The transmit FIFO can be filled using processor interrupts or the DMA controller.

Using processor interrupts

The processor can write one long word (4 bytes) of data to the transmit FIFO when the TRDY field in Serial Channel B/A/C/D Status Register A (see "Serial Channel B/A/C/D Status Register A," beginning on page 617) is active high. If the THALF field in Serial Channel B/A/C/D Status Register A is active high, the processor can write four long words (16 bytes) of data to the transmit FIFO. To facilitate an interrupt when either the TRDY or THALF status bits are active, the processor can set one or both of the corresponding interrupt enables (in "Serial Channel B/A/C/D Control Register A," beginning on page 611).

Using the DMA controller

When using the DMA controller, the processor need not interface with any of the serial port registers for data flow; rather, the processor must interface with the DMA channel registers and DMA buffer descriptor block. To facilitate the use of transmit DMA, the EXTDMA field in Serial Channel B/A/C/D Control Register A must be set active high. When the ETXDMA field is set active high, disable the serial transmitter interrupts.

Receive FIFO interface

The receive FIFO presents up to four bytes of data at a time to the processor interface. The number of valid bytes found in the next read of the FIFO is defined by the information in the RXFDB field (in "Serial Channel B/A/C/D Status Register A" on page 617).

- When the system is configured to operate in big endian mode, the most significant bytes in the word written to the FIFO are read first. For example, the long word 0x11223344 results in the character 0x11 being read first, and 0x44 being read last.

- When the system is configured to operate in little endian mode, the least significant bytes in the word written to the FIFO are read first. For example, the long word 0x11223344 results in the character 0x44 being read first, and 0x11 being read last.

When reading from the receive FIFO, the processor must perform a long word read operation. Each time a read cycle to the receive FIFO is performed, the receive FIFO advances to the next long word entry. The processor cannot read individual bytes from the same FIFO long word entry.

Processor interrupts vs. DMA

The receive FIFO can be emptied using processor interrupts or the DMA controller.

Using processor interrupts

The processor can read one long word (4 bytes) of data from the receive FIFO when the RRDY field (in "Serial Channel B/A/C/D Status Register A" on page 617) is set active high. The long word read may have 1, 2, 3, or 4 bytes of valid data within the word. The number of valid bytes is determined by the bit encoding in the RXFDB field in Serial Channel B/A/C/D Status Register A. The RXFDB field must be read before the FIFO Data register is read.

The RBC bit in Serial Channel B/A/C/D Status Register A indicates that a receive data buffer has been closed and receiver status can be read from this register. Before additional data can be read from the FIFO, the RBC bit must be acknowledged by writing a 1 to the same bit position in Serial Channel B/A/C/D Status Register A.

These steps provide the recommended process flow for the serial port receiver interrupt service routine:

1. Read Serial Channel B/A/C/D Status Register A.
2. If RRDY is true:
   - a. Read the data FIFO.
   - b. Use the RXFDB field to pick out valid bytes.
3. If RBC is true:
   - a. Record receiver buffer closed status (if you want to).
   - b. Write a 1 to the RBC bit position in Serial Channel B/A/C/D Status register A.
   - c. Read Serial Channel B/A/C/D Status Register A again.

To facilitate an interrupt when either the RRDY or RBC status bits are active, the processor must set one or both of the corresponding interrupt enables in Serial Channel B/A/C/D Control Register A.

Using the DMA controller

When using DMA, the processor need not interface with any of the serial port registers for data flow; rather, the processor must interface with the DMA channel registers and the DMA buffer descriptor block. To facilitate use of transmit DMA, the ERXDMA field in Serial Channel B/A/C/D Control register A must be set active high. When ERXDMA is set active high, disable the serial receiver interrupts.

Serial port performance

The serial ports have a finite performance limit on their ability to handle various serial protocols. The performance is limited by the speed of the SYSCLK operating the NS9750 ASIC. The configured speed for the internal PLL defines the BCLK rate; for UART (x8), the serial port maximum rate is 1834200 baud, for UART (x16), the serial port maximum rate is 921600 baud, and for UART (x32), the serial port maximum rate is 460800 baud.

Serial port control and status registers

The configuration registers for serial controller B are located at 0x9020_0000; the configuration registers for serial controller A are located at 0x9020_0040. Table 365 shows a single, two-channel address map for serial controllers B and A.

All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Address	Description
9020 0000	Channel B Control Register A
9020 0004	Channel B Control Register B

Table 365: Serial channel B & A configuration registers

Address	Description
9020 0008	Channel B Status Register A
9020 000C	Channel B Bit-Rate register
9020 0010	Channel B FIFO Data register
9020 0014	Channel B Receive Buffer Gap Timer
9020 0018	Channel B Receive Character Gap Timer
9020 001C	Channel B Receive Match register
9020 0020	Channel B Receive Match Mask register
9020 0034	Channel B Flow Control register
9020 0038	Channel B Flow Control Force register
9020 0040	Channel A Control Register A
9020 0044	Channel A Control Register B
9020 0048	Channel A Status Register A
9020 004C	Channel A Bit-Rate register
9020 0050	Channel A FIFO Data register
9020 0054	Channel A Receive Buffer Gap Timer
9020 0058	Channel A Receive Character Gap Timer
9020 005C	Channel A Receive Match register
9020 0060	Channel A Receive Match Mask register
9020 0074	Channel A Flow Control register
9020 0078	Channel A Flow Control Force register

Table 365: Serial channel B & A configuration registers

The configuration registers for serial controller C are located at 0x9030_0000; the configuration registers for serial controller D are located at 0x9030_0040. Table 366 shows a single, two-channel address map for serial controllers C and D.

Address	Description
9030 0000	Channel C Control Register A
9030 0004	Channel C Control Register B
9030 0008	Channel C Status Register A
9030 000C	Channel C Bit-Rate register
9030 0010	Channel C FIFO Data register
9030 0014	Channel C Receive Buffer Gap Timer
9030 0018	Channel C Receive Character Gap Timer
9030 001C	Channel C Receive Match register
9030 0020	Channel C Receive Match Mask register
9030 0034	Channel C Flow Control register
9030 0038	Channel C Flow Control Force register
9030 0040	Channel D Control Register A
9030 0044	Channel D Control Register B
9030 0048	Channel D Status Register A
9030 004C	Channel D Bit-Rate register
9030 0050	Channel D FIFO Data register
9030 0054	Channel D Receive Buffer Gap Timer
9030 0058	Channel D Receive Character Gap Timer
9030 005C	Channel D Receive Match register
9030 0060	Channel D Receive Match Mask register
9030 0074	Channel D Flow Control register
9030 0078	Channel D Flow Control Force register

Table 366: Serial channel C & D configuration registers

Serial Channel B/A/C/D Control Register A

Address: 9020 0000 / 0040

9030 0000 / 0040

There are two Serial Channel B/A/C/D Control Registers A within each two-channel serial controller module.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CE	BRK	STICK P	EPS	PE	STOP	WLS		CTSTX	RTSRX	RL	LL	Not used		DTR	RTS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used				RIE			ERX DMA	RIC			TIC				ETX DMA

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	CE	0	**Channel enable** 0 Resets the port and the data FIFOs (disables the channel) 1 Enables a serial channel operation The CE field must not be set until all control bits in Serial Channel Control Register A, Control Register B, and Bit-rate register have been defined.
D30	R/W	BRK	0	**Send break** Forces a break condition in UART mode. While BRK is set to 1, the UART transmitter outputs a logic 0 or a space condition on the TXD output signal.
D29	R/W	STICKP	0	**Stick parity** Can be used to force the UART parity field to a certain state as defined by the EPS field (see D28), instead of a parity bit calculated against the data word. STICKP applies only when the PE field (see D27) is also set to 1. Set STICKP to 1 to force transmission of the static parity value.

Table 367: Serial Channel B/A/C/D Control Register A

Bits	Access	Mnemonic	Reset	Description
D28	R/W	EPS	0	**Even parity select** 0 Odd parity 1 Even parity Determines whether the serial channel uses odd or even parity when calculating the parity bit in UART mode. When the STICKP field is set, EPS defines the static state for the parity bit.
D27	R/W	PE	0	**Parity enable** Enables/disables parity generation/checking for the UART transmitter and receiver. The transmitter generates proper parity. The receiver checks for proper parity. If the receiver encounters a bad parity bit, the RPE field is set in the Serial Channel B/A/C/D Status Register A (see page 620).
D26	R/W	STOP	0	**Stop bits** 0 1 stop bit 1 2 stop bits Determines the number of stop bits in each UART transmitter.
D25:24	R/W	WLS	00	**Word length select** 00 5 bits 01 6 bits 10 7 bits 11 8 bits Determines the number of data bits in each UART data word.
D23	R/W	CTSTX	0	**Activate clear to send** Supports hardware handshaking. When CTSTX is set, the transmitter operates only when the external CTS signal is in the active state. An external device, then, can use CTS to temporarily stall data transmission.
D22	R/W	RTSRX	0	**Activate ready to send** Supports hardware handshaking. When RTSRX is set, the RTS output provides the receiver FIFO almost-full condition. When the receiver FIFO backs up, the RTS signal is dropped. The RTS output stalls an external transmitter from delivering data.

Table 367: Serial Channel B/A/C/D Control Register A

Bits	Access	Mnemonic	Reset	Description
D21	R/W	RL	0	**Remote loopback** Provides a remote loopback feature. When RL is set to 1, the TXD transmit output is connected to the RXD receive input. The RL field immediately echoes receive data back as transmit data. This field is used primarily as a test vehicle for external data equipment.
D20	R/W	LL	0	**Local loopback** Provides an internal local loopback feature. When LL is set to 1, the internal receive data stream is connected to the TXD output signal. LL connects the serial channel receiver directly to the serial channel transmitter. This field is used primarily as a test vehicle for the serial channel driver firmware.
D19:18	R/W	Not used	00	This field should be written to 0.
D17	R/W	DTR	0	**Data terminal ready** Controls the state of the external data terminal ready signal. ■ Setting DTR to 1 causes the DTR output to go active. ■ Setting DTR to 0 causes the DTR output to go inactive.
D16	R/W	RTS	0	**Request-to-send** Controls the state of the external request to send signal. ■ Setting RTS to 1 causes the RTS output to go active. ■ Setting RTS to 0 causes the RTS output to go inactive.
D15:12	R/W	Not used	0x0	This field should be written to 0.
D11:09	R/W	RIE	0x00	**Receive interrupt enable** 0 Disables the interrupt 1 Enables the interrupt Allows you to enable interrupts for different receive errors and conditions. [11] Receive register ready [10] Receive FIFO half-full [09] Receive buffer closed

Table 367: Serial Channel B/A/C/D Control Register A

Bits	Access	Mnemonic	Reset	Description
D08	R/W	ERXDMA	0	**Enable receive DMA** Enables the receiver to interact with a DMA channel. The channel is configured to operate in DMA mode when ERXDMA is set to 1. In DMA mode, the DMA controller empties the receive data FIFO and delivers the data to memory. The receive status information from Status Registers B and C are moved automatically to the receive DMA buffer descriptor. This bit is cleared to pause the receiver.
D07:05	R/W	RIC	000	**Receiver interrupt condition** Defines the interrupt enables for a receiver interrupt: [7] Change in DCD interrupt enable [6] Change in RI interrupt enable [5] Change in DSR interrupt enable
D04:01	R/W	TIC	0x0	**Transmitter interrupt condition** Defines the interrupt enables for a transmitter interrupt: [4] Change in CTS interrupt enable [3] Transmit register empty interrupt enable [2] Transmit FIFO half-empty interrupt enable [1] Transmit buffer closed interrupt enable
D00	R/W	ETXDMA	0	**Enable transmit DMA** Enables the transmitter to interact with a DMA channel. The channel is configured to operate in DMA mode when ETXDMA is set to 1. In DMA mode, the DMA controller loads the transmit data FIFO from memory. The transmit status information from Status Register C is moved automatically to the transmit DMA buffer descriptor. This bit is cleared to pause the transmitter.

Table 367: Serial Channel B/A/C/D Control Register A

Serial Channel B/A/C/D Control Register B

Address: 9020 0004 / 0044
9030 0004 / 0044

There are two Serial Channel B/A/C/D Control Registers B within each two-channel serial controller module.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDM				RBGT	RCGT	Not used				MODE		BIT ORDR	Not used		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTSTX	Not used			Reserved						Not used	Reserved				

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	RDM	0x0	**Enable receive data match** [31] RDM1 [30] RDM2 [29] RDM3 [28] RDM4 Enables the receive data match comparators. A receive data match comparison detection can be used to close the current receive buffer descriptor. The last byte in the current receive data buffer contains the match character. Each of these bits enables the respective byte found in the Receive Match register.
D27	R/W	RBGT	0	**Receive buffer GAP timer** Detects the maximum allowed time from when the first byte is placed into the receive data buffer and when the receive data buffer is closed. When RBGT is set to 1, the BGAP field in Serial Channel B/A/C/D Status Register A is set when the timeout value defined in the Receive Buffer GAP Timer register has expired.
D26	R/W	RCGT	0	**Receive character GAP timer** Detects the maximum allowed time from when the last byte is placed into the receive data buffer and when the data buffer is closed. When RCGT is set to 1, the CGAP field in Serial Channel B/A/C/D Status Register A is set when the timeout value defined in the Receive Character GAP Timer register has expired.
D25:22	R/W	Not used	0	Must be written as 0.
D21:20	R/W	MODE	00	**Serial channel mode** 00 UART mode 01 Reserved 10 SPI master mode 11 SPI slave mode Configures the serial channel to operate in UART or SPI mode. The MODE field must be set before the CE bit in Serial Channel B/A/C/D Control Register A is set to 1.

Table 368: Serial Channel B/A/C/D Control Register B

Bits	Access	Mnemonic	Reset	Description
D19	R/W	BITORDR	0	**Bit ordering** 0 Bits are processed LSB first, MSB last 1 Bits are processed MSB first, LSB last Controls the order in which bits are transmitted and received in the Serial Shift register.
D18:16	R/W	Not used	0	Must be written as 0.
D15	R/W	RTSTX	0	**Enable active RTS only while transmitting** Controls the RTS indicator. When RTSTX is set, the RTS output goes active only when the transmitter is actively sending a transmit character. RTSTX allows external hardware to use the RTS signal as a transmit line driver enable signal in multi-drop applications. **Note:** The RTS field in Serial Channel Control Register A must also be set. If the RTSRX field in Serial Channel Control Register A is set, however, do ***not*** set this — the RTSTX — field.
D14:12	R/W	Not used	000	Must be written as 0.
D11:06	N/A	Reserved	N/A	N/A
D05	R/W	Not used	0	Must be written as 0.
D04:00	N/A	Reserved	N/A	N/A

Table 368: Serial Channel B/A/C/D Control Register B

Serial Channel B/A/C/D Status Register A

Address: 9020 0008 / 0048
9030 0008 / 0048

The fields in Serial Channel B/A/C/D Status Register A operate differently when DMA mode is used. Many fields are not required for DMA mode, as they are copied to the status field in the DMA buffer descriptor. See the discussion of the DMA Buffer Descriptor register status field in the BBus DMA Controller chapter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Match				BGAP	CGAP	Not used				RXFDB		DCD	RI	DSR	CTS
1	2	3	4												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBRK	RFE	RPE	ROVER	RRDY	RHALF	RBC	RFS	DCDI	RII	DSRI	CTSI	TRDY	THALF	Not used	T EMPTY

Bits	Access	Mnemonic	Reset	Description
D31:28	R	MATCH	0x0	**Match bit** [31] Match1 [30] Match2 [29] Match3 [28] Match4 Set when a match character in the Receive Match register is configured at the same time that the enable receive data match bit is set in Serial Channel Control Register B (see "RDM" on page 616). The match bit indicates that a data match was found in the receive data stream, and the current receive data buffer has been closed. The last character in the receive data buffer contains the actual match character found. In DMA mode, the MATCH field is copied to bits [15:12] in the DMA buffer descriptor.
D27	R	BGAP	0	**Buffer GAP timer** Set when the receive buffer GAP timer is set in Serial Channel Control Register B (see "RBGT" on page 616) and the timeout value defined in the Receive Buffer GAP Timer register has expired. This bit indicates that the maximum allowed time has passed since the first byte was placed into the receive data buffer. The receive data buffer is closed under this condition. In DMA mode, this field is copied to bit [11] in the DMA buffer descriptor.

Table 369: Serial Channel B/A/C/D Status Register A

Bits	Access	Mnemonic	Reset	Description
D26	R	CGAP	0	**Character GAP timer** Set when the enable receive character GAP timer is set in Serial Channel Control Register B (see "RCGT" on page 616) and the timeout value defined in the Receive Character GAP Timer register (see "Serial Channel B/A/C/D Receive Character GAP Timer" on page 632) has expired. This bit indicates that the maximum allowed time has passed since the previous byte was placed into the receive data buffer. The receive data buffer is closed under this condition. In DMA mode, this field is copied to bit [10] in the DMA buffer descriptor.
25:22	R	Not used	0x0	This field is always read as 0x0.
D21:20	R	RXFDB	00	**Receive FIFO data available** 00 Full word 01 One byte 10 Half word 11 Three bytes This field is valid only when RRDY = 1. Identifies the number of valid bytes contained in the next long word to be read from the Serial Channel FIFO Data register. The next read of the FIFO can contain one, two, three, or four valid bytes of data. This field must be read before the FIFO is read to determine which bytes of the 4-byte long word contain valid data. Normal endian byte ordering rules apply to the Serial Channel FIFO Data register.
D19	R	DCD	0	**Data carrier detect** 0 Inactive 1 Active Indicates the current state of the EIA data carrier detect signal.
D18	R	RI	0	**Ring indicator** 0 Inactive 1 Active Indicates the current state of the EIA ring indicator signal.

Table 369: Serial Channel B/A/C/D Status Register A

Bits	Access	Mnemonic	Reset	Description
D17	R	DSR	0	**Data set ready** 0 Inactive 1 Active Indicates the current state of the EIA data set ready signal.
D16	R	CTS	0	Clear to send 0 Inactive 1 Active Indicates the current state of the EIA clear-to-send signal.
D15	R	RBRK	0	**Receive break condition** Indicates that a receive break condition has been found. The receive data buffer is closed under this condition. In DMA mode, this field is copied to bit [3] in the DMA buffer descriptor.
D14	R	RFE	0	**Receive framing error** Indicates that a receive framing error condition has been found. The receive buffer is closed under this condition. In DMA mode, this field is copied to bit [2] in the DMA buffer descriptor.
D13	R	RPE	0	**Receive parity error** Indicates that a receive parity error has been found. The receive data buffer is closed under this condition. In DMA mode, this field is copied to bit [1] in the DMA buffer descriptor.

Table 369: Serial Channel B/A/C/D Status Register A

Bits	Access	Mnemonic	Reset	Description
D12	R	ROVER	0	**Receive overrun** Indicates that a receive *overrun* error condition has been found. An overrun condition indicates that the FIFO was full while data needed to be written by the receiver. When the FIFO is full, any new receive data is discarded; the contents of the FIFO prior to the overrun condition remain the same. The receive data buffer is closed under this condition. In DMA mode, this field is copied to bit [0] in the DMA buffer descriptor. **Be aware:** The overrun status may not be captured properly in the status FIFO for a serial RX FIFO overrun. if this situation, the overrun condition does not result in a buffer closure and the overrun status bit is not set properly when the receive data is read from the FIFO.
D11	R	RRDY	0	**Receive register ready** Indicates that data is available to be read from the FIFO Data register. Before reading the FIFO Data register, the RXFDB field in this (Serial Channel Status Register A) register (see D21:20) must be read to determine how many active bytes are available during the next read of the FIFO Data register. RRDY typically is used only in interrupt-driven applications; this field is not used for DMA operation. The RRDY status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A. The RRDY bit is never active when the RBC (D09) bit is active. The RBC bit must be acknowledged by writing a 1 to the same bit position in this register to activate the RRDY bit. When the receiver is configured to operate in DMA mode, hardware automatically handles the interlock between RBC and RRDY.

Table 369: Serial Channel B/A/C/D Status Register A

Bits	Access	Mnemonic	Reset	Description
D10	R	RHALF	0	**Receive FIFO half full** Indicates that the receive data FIFO contains at least 20 bytes (5 lines). RHALF typically is used only in interrupt-driven applications; this field is not used for DMA operation. The RHALF status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A.
D09	R	RBC	0	**Receive buffer closed** Indicates a receive buffer closed condition. Hardware automatically acknowledges this field when the receiver is configured to operate in DMA mode. The RBC status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A. While the RBC field is active, the RRDY field is not active. To activate RRDY (to read the data FIFO), the RBC bit must be acknowledged by writing a 1 to the RBC field. This interlock between RBC and RRDY allows the firmware driver to read the status bits in Serial Channel Status Register A or Status Register B. When operating in DMA mode, hardware automatically handles the interlock between RBC and RRDY.
D08	R	RFS	0	**Receive FIFO status** Reflects the current state of the receive FIFO. When set to 1, the receive FIFO has room for only one more line of data.
D07	RW1TC	DCDI	0	**Change in DCD** Indicates a state change in the EIA data carrier detect signal. A 1 indicates that a state change has occurred. This field is asserted only when the corresponding IE bit — RIC field, D[07] — is set to 1 in Serial Channel Control Register A.
D06	R	RII	0	**Change in RI** Indicates a state change in the EIA ring indicator signal. A 1 indicates that a state change has occurred. This field is asserted only when the corresponding IE bit — RIC field, D[06] — is set to 1 in Serial Channel Control Register A.

Table 369: Serial Channel B/A/C/D Status Register A

Bits	Access	Mnemonic	Reset	Description
D05	R	DSRI	0	**Change in DSR** Indicates a state change in the EIA data set ready signal. A 1 indicates that a state change has occurred. This field is asserted only when the corresponding IE bit — RIC field, D[05] — is set to 1 in Serial Channel Control Register A.
D04	R	CTSI	0	**Change in CTS** Indicates a state change in the EIA clear-to-send signal. A 1 indicates that a state change has occurred. This field is asserted only when the corresponding IE bit — TIC field, D[04] — is set to 1 in Serial Channel Control Register A.
D03	R	TRDY	0	**Transmit register empty** Indicates that data can be written to the FIFO Data register. TRDY typically is used only in interrupt-driven applications; this field is not used for DMA operation. The TRDY status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A.
D02	R	THALF	0	**Transmit FIFO half empty** Indicates that the transmit data FIFO contains room for at least 16 bytes. THALF typically is used only in interrupt-driven applications; this field is not used for DMA operation. The THALF status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A.
D01	R	Not used	N/A	**Must be written to 0.**
D00	R	TEMPTY	0	**Transmit FIFO empty** Indicates that the transmit data FIFO currently is empty. TEMPTY simply reports the status of the FIFO; this bit does not indicate that the character currently in the Transmit Shift register has been transmitted.

Table 369: Serial Channel B/A/C/D Status Register A

Serial Channel B/A/C/D Bit-rate register

Address: 9020 000C / 004C

9030 000C / 004C

The Serial Channel B/A/C/D Bit-rate register contains the serial channel timing reference control bits and the data rate control bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EBIT	T MODE	RXSRC	TX SRC	RX EXT	TX EXT	CLKMUX		TXC INV	RXC INV	Rsvd	TDCR		RDCR		Not used

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used	N (divisor value)														

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	EBIT	0	**Bit-rate generator enable** Enables the internal bit-rate generator when set to 1.
D30	R/W	TMODE	0	**Timing mode** Must be set to 1. Use the additional timing configuration provided by the TDCR and RDCR fields (D[20:19] and D[18:17] in this register) to configure the channel for 1x, 8x, 16x, or 32x mode.
D29	R/W	RXSRC	0	**Receive timing source** 0 Internal 1 External (input using GPIO pin) Controls the source of the receiver clock. The receive clock can be provided by an internal source selected using the RICS field (see D15). As an alternative, the receiver clock can be provided by an input on GPIO pins gpio[6], gpio[14], gpio[22], and gpio[26] for serial ports B, A, C, and D, respectively.

Table 370: Serial Channel B/A/C/D Bit-rate register

Bits	Access	Mnemonic	Reset	Description
D28	R/W	TXSRC	0	**Transmit clock source** 0 Internal 1 External (input using GPIO pin) Controls the source of the transmitter clock. The transmitter clock can be provided by an internal source selected using the TICS field (see D16). As an alternative, the transmitter clock can be provided by an input on GPIO pins gpio[7], gpio[15], gpio[23], and gpio[27] for serial ports B, A, C, and D, respectively.
D27	R/W	RXEXT	0	**Drive receive clock external** 0 Disable 1 Enable Enables the receiver clock to be driven on GPIO pins gpio[6], gpio[14], gpio[22], and gpio[26] for serial ports B, A, C, and D, respectively.‘
D26	R/W	TXEXT	0	**Drive transmit clock external** 0 Disable 1 Enable Enables the transmitter clock to be driven on GPIO pins gpio[7], gpio[15], gpio[23], gpio[27] for serial ports B, A, C, and D, respectively.

Table 370: Serial Channel B/A/C/D Bit-rate register

Bits	Access	Mnemonic	Reset	Description
D25:24	R/W	CLKMUX	00	**Bit-rate generator clock source** Controls the bit-rate generator clock source. The bit-rate generator can be configured to use one of four clock sources: 00 x1_sys_osc/M (see Table 364, “Bit-rate generation clock sources,” on page 604 for more information). This is the recommended setting for standard UART baud rate generation. This selection is not valid when the PLLBP field in the PLL Configuration register is set to 1. 01 BCLK This is the recommended setting for SPI operation. 10 Input clock defined by external receive clock on GPIO pins gpio[6], gpio[14], gpio[22], and gpio[26] for serial ports B, A, C, and D, respectively. 11 Input clock defined by external transmit clock on GPIO pins gpio[7], gpio[15], gpio[23], and gpio[27] for serial ports B, A, C, and D, respectively.
D23	R/W	TXCINV	0	**Transmit clock invert** Controls the relationship between transmit clock and transmit data: ■ When set to 0, transmit data changes relative to the falling edge transition of the transmit clock. ■ When set to 1, transmit data changes relative to the rising edge transition of the transmit clock.
D22	R/W	RXCINV	0	**Receive clock invert** Controls the relationship between receive clock and receive data: ■ When set to 0, the receive data input is sampled at the rising edge transition of the receive clock. ■ When set to 1, the receive data input is sampled at the falling edge transition of the receive clock.
D21	N/A	Reserved	N/A	N/A

Table 370: Serial Channel B/A/C/D Bit-rate register

Bits	Access	Mnemonic	Reset	Description
D20:19	R/W	TDCR	00	**Transmit clock divide rate** 00 Not valid for UART 01 8x clock mode 10 16x clock mode 11 32x clock mode Determines the divide ratio for the transmitter clock.
D18:17	R/W	RDCR	00	**Receive clock divide rate** 00 Not valid for UART 01 8x clock mode 10 16x clock mode 11 32x clock mode Determines the divide ratio for the receiver clock.
D16	R/W	Not used	0	Always write 0 to this field.
D15	R/W	Not used	0	**Always write 0** to this field.
D14:00	R/W	N	0x0000	**Divisor value** Defines the divisor value used in the bit-rate generator to determine effective frequency of the bit-rate generator. The divisor value for UART operation is defined as follows: N = ((F_{CLK} / (UM * BR)) – 1) where: ■ F_{CLK} = Determined by CLKMUX field ■ UM = UartMode = 8, 16, or 32 as defined by RDCR and TDCR ■ BR = BaudRate = Required baud rater See Table 371 and Table 372 for examples.

Table 370: Serial Channel B/A/C/D Bit-rate register

The next tables show sample UART baud rates. These rates can be produced using the recommended PLL reference oscillator frequency of 29.4912 MHz and setting the CLKMUX field in the Bit Rate register to 0. The first table shows the range of available baud rates when the PLLND field in the PLL Configuration register is at least 0x13. The second table shows the range of available baud rates when the PLLND field is less then 0x13.

Baud rate	N field		
	x8 UART mode	x16 UART mode	x32 UART mode
75	N/A	12287	6143
150	12287	6143	3071
300	6143	3071	1535
600	3071	1535	767
1200	1535	767	383
2400	767	383	191
4800	383	191	95
7200	255	127	63
9600	191	95	47
14400	127	63	31
19200	95	47	23
28800	63	31	15
38400	47	23	11
57600	31	15	7
115200	15	7	3
230400	7	3	1
460800	3	1	0
921600	1	0	N/A
1843200	0	N/A	N/A

Table 371: Bit-rate examples for X1_SYS_OSC/2

Baud rate	N field		
	x8 UART mode	x16 UART mode	x32 UART mode
75	12287	6143	3071

Table 372: Bit-rate examples for X1_SYS_OSC/4

Baud rate	N field		
	x8 UART mode	x16 UART mode	x32 UART mode
150	6143	3071	1535
300	3071	1535	767
600	1535	767	383
1200	767	383	191
2400	383	191	95
4800	191	95	47
7200	127	63	31
9600	95	47	23
14400	63	31	15
19200	47	23	11
28800	31	15	7
38400	23	11	5
57600	15	7	3
115200	7	3	1
230400	3	1	0
460800	1	0	N/A
921600	0	N/A	N/A
1843200	N/A	N/A	N/A

Table 372: Bit-rate examples for X1_SYS_OSC/4

Serial Channel B/A/C/D FIFO Data register

Address: 9020 0010 / 0050
9030 0010 / 0050

The Serial Channel B/A/C/D FIFO Data registers manually interface with the serial controller FIFOs instead of using DMA support.

Writing to the transmit register loads the transmit FIFO. This register can be written only when the TRDY field is set in Serial Channel Status Register A. Writing to the Serial Channel FIFO Data register automatically clears the TRDY bit.

Reading from the receive register empties the receive FIFO. Data is available when the RRDY bit is set in Serial Channel Status Register A. The RXFDB field in Serial Channel Status Register A identifies how many bytes are available to be read. Reading the Serial Channel FIFO Data register automatically clears the RRDY bit in Serial Channel Status Register A.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	DATA	0x00000000	Serial channel FIFO data field.

Table 373: Serial Channel B/A/C/D FIFO Data register

Serial Channel B/A/C/D Receive Buffer GAP Timer

Address: 9020 0014 / 0054

9030 0014 / 0054

The Receive Buffer GAP Timer closes out a receive serial data buffer. This timer can be configured to provide an interval in the range of 34.7uS to 2.27 S. The timer is reset when the first character is received in a new buffer. New characters are received while the timer operates; when the timer reaches its programmed threshold, the receive buffer is closed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRUN	Not used														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Buffer GAP timer (BT)															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	TRUN	0	**Buffer GAP timer enable** 0 Disables the timer 1 Enables the timer
D30:16	R/W	Not used	0x0000	Must be written as 0.
D15:00	R/W	BT	0x0000	Buffer GAP timer Defines the required value for the receive buffer GAP timer. BT is a function of the channel bit-rate and the receive buffer size. **Recommended approach:** Set the buffer GAP timer to be a value that is slightly larger than the amount of time required to fill the maximum buffer size using the channel bit-rate. Compute the BT value as shown: BT = (((Timeout * F_{CLK}) / 512) - 1) where: ■ F_{CLK} = (x1_sys_osc / M) (see Table 364 for examples) ■ Timeout = Appropriate timeout in seconds Use the following equation to define the recommended buffer GAP timeout value. Timeout = (((BufferSize + 1) * 8) / DataRate) where: ■ BufferSize = receive buffer size in bytes ■ DataRate = interface data rate in bits

Table 374: Serial Channel B/A/C/D Receive Buffer GAP Timer

Serial Channel B/A/C/D Receive Character GAP Timer

Address: 9020 0018 / 0058
9030 0018 / 0058

The receive character GAP timer closes out a receive serial data buffer due to a gap between characters. This timer is configured to provide an interval in the range of 0.27us to 0.28S. The timer is reset when a character is received. When the timer reaches its programmed threshold, the receive data buffer is closed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRUN	Not used											Character GAP timer (CT)			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Character GAP timer (CT)															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	TRUN	0x0	**Character GAP timer enable** 0 Disables the timer 1 Enables the timer
D30:20	R/W	Not used	0x000	Must be written as 0.

Table 375: Serial Channel B/A/C/D Receive Character GAP Timer

Bits	Access	Mnemonic	Reset	Description
D19:00	CT	CT	0x00000	**Character GAP timer** Defines the required value for the receive character GAP timer. Compute the CT value as shown: CT = (((Timeout * F_{CLK})/4) - 1 where: ■ F_{CLK} = (x1_sys_osc / M) (see Table 364 for examples) ■ Timeout = Appropriate timeout in seconds Use the following equation to define the recommended character GAP timeout value: Timeout = (NumBitGap / DataRate) where: ■ NumBitGap = The number of bits in a character plus any start, stop, or parity bits. ■ DataRate = Interface data rate in bits per second. **Note:** When PLL is bypassed, F_{CLK} equals BCLK which equals x1_sys_osc/8. In most designs BCLK = (398.1312Mhz / 8) = 49.7664 Mhz. Examples: Timeout=0.1ms ; CT field=0x004DB Timeout=1.0ms ; CT field=0x03098 Timeout=5.0ms ; CT field=0x0F2FF The maximum value that can be programmed into the CT field is 0x7FFFF.

Table 375: Serial Channel B/A/C/D Receive Character GAP Timer

Serial Channel B/A/C/D Receive Match register

Address: 9020 001C / 005C

9030 001C / 005C

The Serial Channel B/A/C/D Receive Match register contains the four receive data match bytes used in UART mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDMB1								RDMB2							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDMB3								RDMB4							

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:24	R/W	RDMB1	0x00	Receive data match byte1
D23:16	R/W	RDMB2	0x00	Receive data match byte2
D15:08	R/W	RDMB3	0x00	Receive data match byte3
D07:00	R/W	RDMB4	0x00	Receive data match byte4

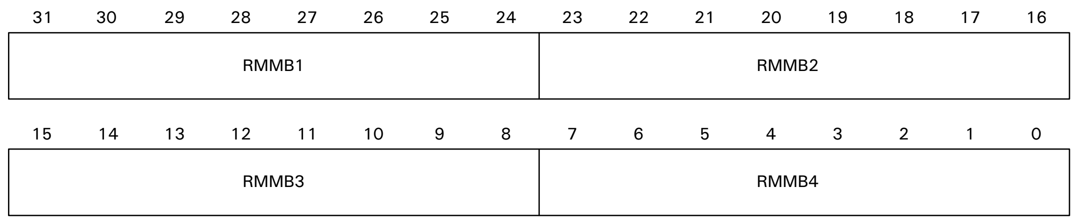
Table 376: Serial Channel B/A/C/D Receive Match register

Serial Channel B/A/C/D Receive Match MASK register

Address: 9020 0020 / 0060

9030 0020 / 0060

The Serial Channel B/A/C/D Receive Match MASK register contains the four receive match mask bytes that specify which bits in the Receive Match Data register should not be included in the match comparison. To mask a bit in the match comparison function, place a 1 in the same bit position in this register.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:24	R/W	RMMB1	0x00	Receive mask match byte1
23:16	R/W	RMMB2	0x00	Receive mask match byte2
15:08	R/W	RMMB3	0x00	Receive mask match byte3
07:00	R/W	RMMB4	0x00	Receive mask match byte4

Table 377: Serial Channel B/A/C/D Receive Match MASK register

Serial Channel B/A/C/D Flow Control register

Address: 9020 0034 / 0074
9030 0034 / 0074

The Serial Channel B/A/C/D Flow Control register allows you to define the flow control operation of the serial controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used						FLOW_ STATE		FLOW4		FLOW3		FLOW2		FLOW1	

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:10	R	Not used	0x000000	Always read as0x000000.
D09:08	R/W	FLOW_STATE	10	**Flow control state** 00 Software-initiated XON 01 Software-initiated XOFF 10 Hardware-initiated XON 11 Hardware-initiated XOFF Defines the current state of the flow control logic.
D07:06	R/W	FLOW4	10	**Flow control enable** 00 Disabled 01 Disabled 10 Change field FLOW_STATE to XON upon match 11 Change field FLOW_STATE to XOFF upon match Allows you to define the flow control characteristics using fields RDMB4 (see "RDMB4" on page 634) and RMMB4 (see "RMMB4" on page 635).

Table 378: Serial Channel B/A/C/D Flow Control register

Bits	Access	Mnemonic	Reset	Description
D05:04	R/W	FLOW3	10	**Flow control enable** 00 Disabled 01 Disabled 10 Change field FLOW_STATE to XON upon match 11 Change field FLOW_STATE to XOFF upon match Allows you to define the flow control characteristics using fields RDMB3 (see "RDMB3" on page 634) and RMMB3 (see "RMMB3" on page 635).
D03:02	R/W	FLOW2	10	**Flow control enable** 00 Disabled 01 Disabled 10 Change field FLOW_STATE to XON upon match 11 Change field FLOW_STATE to XOFF upon match Allows you to define the flow control characteristics using fields RDMB2 (see "RDMB2" on page 634) and RMMB2 (see "RMMB2" on page 635).
D01:00	R/W	FLOW1	10	**Flow control enable** 00 Disabled 01 Disabled 10 Change field FLOW_STATE to XON upon match 11 Change field FLOW_STATE to XOFF upon match Allows you to define the flow control characteristics using fields RDMB1 (see "RDMB1" on page 634) and RMMB1 (see "RMMB1" on page 635).

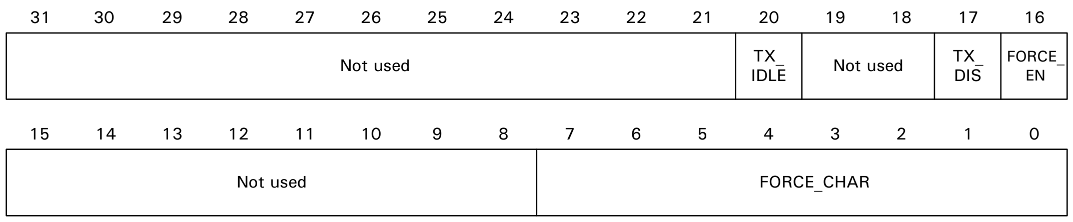
Table 378: Serial Channel B/A/C/D Flow Control register

Serial Channel B/A/C/D Flow Control Force register

Address: 9020 0038 / 0078

9030 0038 / 0078

The Serial Channel B/A/C/D Flow Control Force register allows you to override the normal flow of transmit data.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:21	R	Not used	0x000	Always read as 0x000.
D20	R	TX_IDLE	0	**Transmit idle** Indicates whether the transmit state machine has been forced into an idle state by writing a 1 to the TX_DIS field (see D17).
D19:18	R	Not used	00	Always read as 00.
D17	R/W	TX_DIS	0	**Transmit disable** Allows you to force the transmit state machine into the idle state. If a transmission is in progress, the current byte will complete before the state machine moves into the idle state. Write a 1 to enable this feature.

Table 379: Serial Channel B/A/C/D Flow Control Force register

Bits	Access	Mnemonic	Reset	Description
D16	R/W	FORCE_EN	0	**Force transmit** Allows you to force the transmitter to send the character specified by the FORCE_CHAR (see D07:00) field. All user-specified rules, such as bit order, parity, and number of stop bits, are enforced. Write this field only when TX_IDLE is set to 1. Hardware clears this field once the character has been transmitted. Write a 1 to enable this feature.
D15:08	R	Not used	0x00	Always read as 0x00.
D07:00	R/W	FORCE_ CHAR	0x00	**Force character** Defines the character that is to be forced out of the transmitter.

Table 379: Serial Channel B/A/C/D Flow Control Force register

Serial Control Module: SPI

CHAPTER 14

The NS9750 ASIC supports four independent universal asynchronous/synchronous receiver/transmitter channels. Each channel supports several modes, conditions, and formats.

Features

Each channel supports these features:

- DMA transfers to and from system memory
- Independent programmable bit-rate generator
- High speed data transfer (synchronous)
  - SPI master: 8.33 Mbps
  - SPI slave: 6.25 Mbps
- 32-byte TX FIFO
- 32-Byte RX FIFO

Figure 94 shows the structure of the serial module.

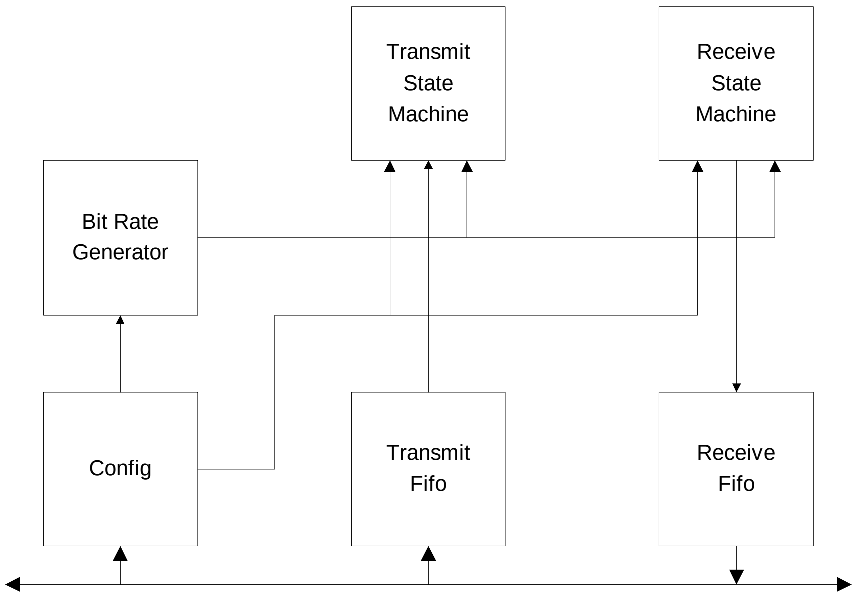


Figure 94: Serial module structure

Bit-rate generator

Each serial channel supports an independent programmable bit-rate generator. The bit-rate generator runs both the transmitter and receiver of a given channel (there is no split speed support).

You can configure the bit-rate generator to use external clock input or internal system timing as its timing reference. This allows for a wider range of possible bit-rates.

Table 380 describes all possible clock reference sources used by the bit-rate generator.

Name	Description
BCLK	The clock source for all peripherals that are attached to the BBus. The frequency of BCLK is the AHB clock frequency divided by 2.
ExtRxClk	External receive clock on GPIO pins gpio[6], gpio[14], gpio[22], and gpio[26] for serial ports B, A, C, and D, respectively.
ExtTxClk	External transmit clock on GPIO pins gpio[7], gpio[15], gpio[23], and gpio[27] for serial ports B, A, C, and D, respectively.

Table 380: Bit-rate generation clock sources

SPI mode

The NS9750 ASIC SPI controller provides these key features:

- Four-wire interface (DATA_OUT, DATA_IN, CLK, ENABLE)
- Master or slave configuration
- Programmable MSB/LSB formatting
- Programmable ENABLE polarity
- Programmable SPI mode (0, 1, 2, 3)

The SPI controller provides a full-duplex, synchronous, character-oriented data channel between master and slave devices, using a four-wire interface (DATA_OUT, DATA_IN, CLK, ENABLE). The master interface operates in a broadcast mode. The slave interface is activated using the ENABLE signal. You can configure the master interface to address various slave interfaces using GPIO pins.

The transmitter and receiver use the same clock. When configured in master mode, the channel's bit-rate generator (see "Bit-rate generator" on page 645) provides the timing reference.

SPI is useful for providing simple parallel/serial data conversion to stream serial data between memory and a peripheral. The SPI port has no protocol associated with it other than that it transfers information in multiples of 8 bits.

The SPI port simultaneously is capable of full duplex operation. The transfer of information is controlled by a single clock signal.

- For the SPI master interface, the clock signal is an output.
- For the SPI slave interface, the clock signal is an input.

The ENABLE signal also qualifies the transfer of information. The SPI ENABLE signal must be active for data transfers to occur, regardless of the SPI clock signal.

SPI modes

The four SPI modes are distinguished by the polarity in which the SPI CLK idles and the SPI CLK data phase used to capture SPI DATA_IN and drive SPI DATA_OUT. Table 381 describes the four modes and the register settings used to select the modes.

	Serial channel B/A/C/D bit rate register settings			Mode functionality		
SPI mode	**SPCPOL**	**TXCINV**	**RXCINV**	**SPI CLK Idle**	**SPI DATA_IN capture edge**	**SPI DATA-OUT drive edge**
0	1	0	0	Low	Rising	Falling
1	0	1	1	High	Falling	Rising
2	1	1	1	Low	Falling	Rising
3	0	0	0	High	Rising	Falling

Table 381: SPI mode definitions

FIFO management

Data flow between a serial controller and memory occurs through the FIFO blocks within each serial controller module. Each serial controller provides both a 32-byte transmit FIFO and a 32-byte receive FIFO. Each FIFO is arranged as eight lines of four bytes to facilitate data transfer across BBus. Both the transmit and receive FIFOs are accessed using the Serial Channel B/A/C/D FIFO registers.

Transmit FIFO interface

The processor can write either 1, 2, 3, or 4 bytes at a time to the transmit FIFO. The number of bytes written is controlled by the data size defined by the HSIZE field on the AMBA AHB bus.

- When the system is configured to operate in big endian mode, the most significant bytes in the word written to the FIFO are transmitted first. For example, the long word 0x11223344 results in the character 0x11 being transmitted first, and 0x44 being transmitted last.
- When the system is configured to operate in little endian mode, the least significant bytes in the word written to the FIFO are transmitted first. For example, the long word 0x11223344 results in the character 0x44 being transmitted first, and 0x11 being transmitted last.

Processor interrupts vs. DMA

The transmit FIFO can be filled using processor interrupts or the DMA controller.

Using processor interrupts

The processor can write one long word (4 bytes) of data to the transmit FIFO when the TRDY field in Serial Channel B/A/C/D Status Register A (see "Serial Channel B/A/C/D Status Register A," beginning on page 657) is active high. If the THALF field in Serial Channel B/A/C/D Status Register A is active high, the processor can write four long words (16 bytes) of data to the transmit FIFO. To facilitate an interrupt when either the TRDY or THALF status bits are active, the processor can set one or both of the corresponding interrupt enables (in "Serial Channel B/A/C/D Control Register A," beginning on page 652).

Using the DMA controller

When using the DMA controller, the processor need not interface with any of the serial port registers for data flow; rather, the processor must interface with the DMA channel registers and DMA buffer descriptor block. To facilitate the use of transmit DMA, the EXTDMA field in Serial Channel B/A/C/D Control Register A must be set active high. When the ETXDMA field is set active high, disable the serial transmitter interrupts.

Receive FIFO interface

The receive FIFO presents up to four bytes of data at a time to the processor interface. The number of valid bytes found in the next read of the FIFO is defined by the information in the RXFDB field (in "Serial Channel B/A/C/D Status Register A" on page 657).

- When the system is configured to operate in big endian mode, the most significant bytes in the word written to the FIFO are read first. For example, the long word 0x11223344 results in the character 0x11 being read first, and 0x44 being read last.
- When the system is configured to operate in little endian mode, the least significant bytes in the word written to the FIFO are read first. For example, the long word 0x11223344 results in the character 0x44 being read first, and 0x11 being read last.

When reading from the receive FIFO, the processor must perform a long word read operation. Each time a read cycle to the receive FIFO is performed, the receive FIFO advances to the next long word entry. The processor cannot read individual bytes from the same FIFO long word entry.

Processor interrupts vs. DMA

The receive FIFO can be emptied using processor interrupts or the DMA controller.

Using processor interrupts

The processor can read one long word (4 bytes) of data from the receive FIFO when the RRDY field (in "Serial Channel B/A/C/D Status Register A" on page 657) is set active high. The long word read may have 1, 2, 3, or 4 bytes of valid data within the word. The number of valid bytes is determined by the bit encoding in the RXFDB field in Serial Channel B/A/C/D Status Register A. The RXFDB field must be read before the FIFO Data register is read.

The RBC bit in Serial Channel B/A/C/D Status Register A indicates that a receive data buffer has been closed and receiver status can be read from this register. Before additional data can be read from the FIFO, the RBC bit must be acknowledged by writing a 1 to the same bit position in Serial Channel B/A/C/D Status Register A.

These steps provide the recommended process flow for the serial port receiver interrupt service routine:

1. Read Serial Channel B/A/C/D Status Register A.
2. If RRDY is true:
   a. Read the data FIFO.
   b. Use the RXFDB field to pick out valid bytes.
3. If RBC is true:
   a. Record receiver buffer closed status (if you want to).
   b. Write a 1 to the RBC bit position in Serial Channel B/A/C/D Status register A.
   c. Read Serial Channel B/A/C/D Status Register A again.

To facilitate an interrupt when either the RRDY or RBC status bits are active, the processor must set one or both of the corresponding interrupt enables in Serial Channel B/A/C/D Control Register A.

Using the DMA controller

When using DMA, the processor need not interface with any of the serial port registers for data flow; rather, the processor must interface with the DMA channel registers and the DMA buffer descriptor block. To facilitate use of transmit DMA, the ERXDMA field in Serial Channel B/A/C/D Control register A must be set active high. When ERXDMA is set active high, disable the serial receiver interrupts.

Serial port performance

The serial ports have a finite performance limit on their ability to handle various serial protocols. The performance is limited by the speed of the SYSCLK operating the NS9750 ASIC. The configured speed for the internal PLL defines the BCLK rate; for SPI, the serial port maximum rate is BCLK/8.

Serial port control and status registers

The configuration registers for serial controller B are located at 0x9020_0000; the configuration registers for serial controller A are located at 0x9020_0040. Table 382 shows a single, two-channel address map for serial controllers B and A.

All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Address	Description
9020 0000	Channel B Control Register A
9020 0004	Channel B Control Register B
9020 0008	Channel B Status Register A
9020 000C	Channel B Bit-Rate register
9020 0010	Channel B FIFO Data register
9020 0040	Channel A Control Register A

Table 382: Serial channel B & A configuration registers

Address	Description
9020 0044	Channel A Control Register B
9020 0048	Channel A Status Register A
9020 004C	Channel A Bit-Rate register
9020 0050	Channel A FIFO Data register

Table 382: Serial channel B & A configuration registers

The configuration registers for serial controller C are located at 0x9030_0000; the configuration registers for serial controller D are located at 0x9030_0040. Table 383 shows a single, two-channel address map for serial controllers C and D.

Address	Description
9030 0000	Channel C Control Register A
9030 0004	Channel C Control Register B
9030 0008	Channel C Status Register A
9030 000C	Channel C Bit-Rate register
9030 0010	Channel C FIFO Data register
9030 0040	Channel D Control Register A
9030 0044	Channel D Control Register B
9030 0048	Channel D Status Register A
9030 004C	Channel D Bit-Rate register
9030 0050	Channel D FIFO Data register

Table 383: Serial channel C & D configuration registers

Serial Channel B/A/C/D Control Register A

Address: 9020 0000 / 0040

9030 0000 / 0040

There are two Serial Channel B/A/C/D Control Registers A within each two-channel serial controller module.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CE	Not used					WLS		Not used		RL	LL	Not used			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used				RIE			ERX DMA	Reserved			TIC				ETX DMA

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	CE	0	**Channel enable** 0 Resets the port and the data FIFOs (disables the channel) 1 Enables a serial channel operation The CE field cannot be set until all control bits in Serial Channel Control Register A, Control Register B, and Bit-rate register are stable.
D30:26	R/W	Not used	0	Always write as 0.
D25:24	R/W	WLS	00	**Word length select** This value must be 2'b11 to select a word length of 8 data bits. SPI mode requires this word length.
D23:22	R/W	Not used		Always write as 0.
D21	R/W	RL	0	**Remote loopback** Provides a remote loopback feature. When RL is set to 1, the TXD transmit output is connected to the RXD receive input. The RL field immediately echoes receive data back as transmit data. This field is used primarily as a test vehicle for external data equipment.

Table 384: Serial Channel B/A/C/D Control Register A

Bits	Access	Mnemonic	Reset	Description
D20	R/W	LL	0	**Local loopback** Provides an internal local loopback feature. When LL is set to 1, the internal receive data stream is connected to the TXD output signal. LL connects the serial channel receiver directly to the serial channel transmitter. This field is used primarily as a test vehicle for the serial channel driver firmware.
D19:12	R/W	Not used	0	This field should be written to 0.
D11:09	R/W	RIE	0x00	**Receive interrupt enable** 0 Disables the interrupt 1 Enables the interrupt Allows you to enable interrupts for different receive errors and conditions. [11] Receive register ready [10] Receive FIFO half-full [09] Receive buffer closed
D08	R/W	ERXDMA	0	**Enable receive DMA** Enables the receiver to interact with a DMA channel. The channel is configured to operate in DMA mode when ERXDMA is set to 1. In DMA mode, the DMA controller empties the receive data FIFO and delivers the data to memory. The receive status information from Status Registers B and C are moved automatically to the receive DMA buffer descriptor. This bit is cleared to pause the receiver.
D07:05	N/A	Reserved	N/A	N/A
D04:01	R/W	TIC	0x0	**Transmitter interrupt condition** Defines the interrupt enables for a transmitter interrupt: [4] Change in CTS interrupt enable [3] Transmit register empty interrupt enable [2] Transmit FIFO half-empty interrupt enable [1] Transmit buffer closed interrupt enable

Table 384: Serial Channel B/A/C/D Control Register A

Bits	Access	Mnemonic	Reset	Description
D00	R/W	ETXDMA	0	**Enable transmit DMA** Enables the transmitter to interact with a DMA channel. The channel is configured to operate in DMA mode when ETXDMA is set to 1. In DMA mode, the DMA controller loads the transmit data FIFO from memory. The transmit status information from Status Register C is moved automatically to the transmit DMA buffer descriptor. This bit is cleared to pause the transmitter.

Table 384: Serial Channel B/A/C/D Control Register A

Serial Channel B/A/C/D Control Register B

Address: 9020 0004 / 0044

9030 0004 / 0044

There are two Serial Channel B/A/C/D Control Registers B within each two-channel serial controller module.

Note: The CE field in Serial Channel Control register A should not be set until these control bits are stabilized.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						CS POL	Not used			MODE		BIT ORDR	Not used		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used				Reserved						Not used	Reserved				

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:26	R/W	Reserved	N/A	N/A
D25	R/W	CSPOL	0	**SPI chip select polarity** 0 Active low chip select 1 Active high chip select Defines the polarity of the SPI chip select signal. **Note:** If you are using SPI slave mode, only a value of 0 is valid; the SPI slave is fixed to an active low chip select. Both values apply to SPI master mode, however.
D24:22	R/W	Not used	0	Must be written as 0.

Table 385: Serial Channel B/A/C/D Control Register B

Bits	Access	Mnemonic	Reset	Description
D21:20	R/W	MODE	00	**Serial channel mode** 00 UART mode 01 Reserved 10 SPI master mode 11 SPI slave mode Configures the serial channel to operate in UART or SPI modes. The MODE field must be set before the CE bit in Serial Channel B/A/C/D Control Register A is set to 1.
D19	R/W	BITORDR	0	**Bit ordering** 0 Bits are processed LSB first, MSB last 1 Bits are processed MSB first, LSB last Controls the order in which bits are transmitted and received in the Serial Shift register.
D18:12	R/W	Not used	0	Always write to 0.
D11:06	N/A	Reserved	N/A	N/A
D05	R/W	Not used	0	Must be written as 0.
D04:00	N/A	Reserved	N/A	N/A

Table 385: Serial Channel B/A/C/D Control Register B

Serial Channel B/A/C/D Status Register A

Address: 9020 0008 / 0048
9030 0008 / 0048

The fields in Serial Channel B/A/C/D Status Register A operate differently when DMA mode is used. Many fields are not required for DMA mode, as they are copied to the status field in the DMA buffer descriptor. See the discussion of the DMA Buffer Descriptor register status field in the BBus DMA Controller chapter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						Not used				RXFDB		Not used			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			ROVER	RRDY	RHALF	RBC	RFS	Not used				TRDY	THALF	Rsvd	T EMPTY

Bits	Access	Mnemonic	Reset	Description
D31:26	N/A	Reserved	N/A	N/A
D25:22	R	Not used	0x0	This field is always read as 0x0.
D21:20	R	RXFDB	00	**Receive FIFO data available** 00 Full word 01 One byte 10 Half word 11 Three bytes This field is valid only when RRDY = 1. Identifies the number of valid bytes contained in the next long word to be read from the Serial Channel FIFO Data register. The next read of the FIFO can contain one, two, three, or four valid bytes of data. This field must be read before the FIFO is read to determine which bytes of the 4-byte long word contain valid data. Normal endian byte ordering rules apply to the Serial Channel FIFO Data register.
D19:16	R	Not used	0	Always write as 0.
D15:13	N/A	Reserved	N/A	N/A

Table 386: Serial Channel B/A/C/D Status Register A

Bits	Access	Mnemonic	Reset	Description
D12	R	ROVER	0	**Receive overrun** Indicates that a receive *overrun* error condition has been found. An overrun condition indicates that the FIFO was full while data needed to be written by the receiver. When the FIFO is full, any new receive data is discarded; the contents of the FIFO prior to the overrun condition remain the same. The receive data buffer is closed under this condition. In DMA mode, this field is copied to bit [0] in the DMA buffer descriptor. **Be aware:** The overrun status may not be captured properly in the status FIFO for a serial RX FIFO overrun. if this situation, the overrun condition does not result in a buffer closure and the overrun status bit is not set properly when the receive data is read from the FIFO.
D11	R	RRDY	0	**Receive register ready** Indicates that data is available to be read from the FIFO Data register. Before reading the FIFO Data register, the RXFDB field in this (Serial Channel Status Register A) register (see D21:20) must be read to determine how many active bytes are available during the next read of the FIFO Data register. RRDY typically is used only in interrupt-driven applications; this field is not used for DMA operation. The RRDY status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A. The RRDY bit is never active when the RBC (D09) bit is active. The RBC bit must be acknowledged by writing a 1 to the same bit position in this register to activate the RRDY bit. When the receiver is configured to operate in DMA mode, hardware automatically handles the interlock between RBC and RRDY.

Table 386: Serial Channel B/A/C/D Status Register A

Bits	Access	Mnemonic	Reset	Description
D10	R	RHALF	0	**Receive FIFO half full** Indicates that the receive data FIFO contains at least 20 bytes (5 lines). RHALF typically is used only in interrupt-driven applications; this field is not used for DMA operation. The RHALF status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A.
D09	R	RBC	0	**Receive buffer closed** Indicates a receive buffer closed condition. Hardware automatically acknowledges this field when the receiver is configured to operate in DMA mode. The RBC status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A. While the RBC field is active, the RRDY field is not active. To activate RRDY (to read the data FIFO), the RBC bit must be acknowledged by writing a 1 to the RBC field. This interlock between RBC and RRDY allows the firmware driver to read the status bits in Serial Channel Status Register A or Status Register B. When operating in DMA mode, hardware automatically handles the interlock between RBC and RRDY.
D08	R	RFS	0	**Receive FIFO status** Reflects the current state of the receive FIFO. When set to 1, the receive FIFO has room for only one more line of data.
D07:04	R	Not used	0	Always write as 0.
D03	R	TRDY	0	**Transmit register empty** Indicates that data can be written to the FIFO Data register. TRDY typically is used only in interrupt-driven applications; this field is not used for DMA operation. The TRDY status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A.

Table 386: Serial Channel B/A/C/D Status Register A

Bits	Access	Mnemonic	Reset	Description
D02	R	THALF	0	**Transmit FIFO half empty** Indicates that the transmit data FIFO contains room for at least 16 bytes. THALF typically is used only in interrupt-driven applications; this field is not used for DMA operation. The THALF status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A.
D01	N/A	Reserved	N/A	N/A
D00	R	TEMPTY	0	**Transmit FIFO empty** Indicates that the transmit data FIFO currently is empty. TEMPTY simply reports the status of the FIFO; this bit does not indicate that the character currently in the Transmit Shift register has been transmitted.

Table 386: Serial Channel B/A/C/D Status Register A

Serial Channel B/A/C/D Bit-rate register

Address: 9020 000C / 004C

9030 000C / 004C

The Serial Channel B/A/C/D Bit-rate register contains the serial channel timing reference control bits and the data rate control bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EBIT	T MODE	RXSRC	TX SRC	RX EXT	TX EXT	CLKMUX		TXC INV	RXC INV	SPC POL	TDCR		RDCR		TICS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RICS	N (divisor value)														

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	EBIT	0	**Bit-rate generator enable** Enables the internal bit-rate generator when set to 1.
D30	R/W	TMODE	0	**Timing mode** Must be set to 1. Use the additional timing configuration provided by the TDCR and RDCR fields (D[20:19] and D[18:17] in this register) to configure the channel for 1x, 8x, 16x, or 32x mode.
D29	R/W	RXSRC	0	**Receive timing source** 0 Internal 1 External (input using GPIO pin) Controls the source of the receiver clock. The receive clock can be provided by an internal source selected using the RICS field (see D15). As an alternative, the receiver clock can be provided by an input on GPIO pins gpio[6], gpio[14], gpio[22], and gpio[26] for serial ports B, A, C, and D, respectively. For SPI master and slave, set to 0 for internal.
D28	R/W	TXSRC	0	**Transmit clock source** 0 Internal 1 External (input using GPIO pin) Controls the source of the transmitter clock. The transmitter clock can be provided by an internal source selected using the TICS field (see D16). As an alternative, the transmitter clock can be provided by an input on GPIO pins gpio[7], gpio[15], gpio[23], and gpio[27] for serial ports B, A, C, and D, respectively. For SPI master and slave, set to 0 for internal.
D27	R/W	RXEXT	0	**Drive receive clock external** 0 Disable 1 Enable Enables the receiver clock to be driven on GPIO pins gpio[6], gpio[14], gpio[22], and gpio[26] for serial ports B, A, C, and D, respectively. For SPI master and slave, set to 0 to disable.

Table 387: Serial Channel B/A/C/D Bit-rate register

Bits	Access	Mnemonic	Reset	Description
D26	R/W	TXEXT	0	**Drive transmit clock external** 0 Disable 1 Enable Enables the transmitter clock to be driven on GPIO pins gpio[7], gpio[15], gpio[23], gpio[27] for serial ports B, A, C, and D, respectively. For SPI master mode, set to 1 to enable. For SPI slave mode, set to 0 to disable.
D25:24	R/W	CLKMUX	00	**Bit-rate generator clock source** Controls the bit-rate generator clock source. The bit-rate generator can be configured to use one of four clock sources: 00 x1_sys_osc/M (see Table 380, "Bit-rate generation clock sources," on page 645 for more information). This selection is not valid when the PLLBP field in the PLL Configuration register is set to 1. 01 BCLK This is the recommended setting for SPI master and slave operation. 10 Input clock defined by external receive clock on GPIO pins gpio[6], gpio[14], gpio[22], and gpio[26] for serial ports B, A, C, and D, respectively. 11 Input clock defined by external transmit clock on GPIO pins gpio[7], gpio[15], gpio[23], and gpio[27] for serial ports B, A, C, and D, respectively.
D23	R/W	TXCINV	0	**Transmit clock invert** Controls the relationship between transmit clock and transmit data: ■ When set to 0, transmit data changes relative to the falling edge transition of the transmit clock. Use 0 for SPI modes 0 and 3. ■ When set to 1, transmit data changes relative to the rising edge transition of the transmit clock. Use 1 for SPI modes 1 and 2.

Table 387: Serial Channel B/A/C/D Bit-rate register

Bits	Access	Mnemonic	Reset	Description
D22	R/W	RXCINV	0	**Receive clock invert** Controls the relationship between receive clock and receive data: ■ When set to 0, the receive data input is sampled at the rising edge transition of the receive clock. Use 0 for SPI modes 0 and 3. ■ When set to 1, the receive data input is sampled at the falling edge transition of the receive clock. Use 1 for SPI modes 1 and 2.
D21	R/W	SPCPOL	0/1	**SPI transmit polarity** 0 Idle high operation; use this value for SPI modes 1 and 3 1 Idle low operation; use this value for SPI modes 0 and 2 Defines the idle polarity of the SPI transmit clock.
D20:19	R/W	TDCR	00	**Transmit clock divide rate** 00 1x clock mode (only NRZ or NRZI allowed) 01 8x clock mode 10 16x clock mode 11 32x clock mode Determines the divide ratio for the transmitter clock. If the DPLL is not used, use the 1x clock mode value (00). When DPLL is used in the application, selecting TDCR/RDCR is a function of the transmitter encoding. The NRZ and NRZI modes can use the 1x configuration; all other encoding must use 8x, 16x, or 32x configuration mode. The 8x configuration provides the highest possible data rate; the 32x mode provides the highest possible resolution. The TMODE bit in this register is maintained for NET+Arm family backward compatibility. When setting the TDCR or RDCR register to a non-zero value, the TMODE bit must be set to 1. When TMODE, TDCR, and RDCR are all set to 0, the port defaults to 16x mode of operation.

Table 387: Serial Channel B/A/C/D Bit-rate register

Bits	Access	Mnemonic	Reset	Description
D18:17	R/W	RDCR	00	**Receive clock divide rate** 00 1x clock mode (only NRZ or NRZI allowed) 01 8x clock mode 10 16x clock mode 11 32x clock mode Determines the divide ratio for the receiver clock. If the DPLL is not used, use the 1x clock mode value (00). When the DPLL is used in the application, selecting TDCR/RDCR is a function of the receiver encoding. The NRXZ and NRZI modes can use the 1x configuration; all other encoding must use the 8x, 16x, or 32x configuration mode. The 8x configuration provides the highest possible data rate; the 32x mode provides the highest possible resolution. The TMODE bit in this register is maintained for NET+Arm family backward compatibility. When setting the TDCR or RDCR register to a non-zero value, the TMODE bit must be set to 1.When the TMODE, TDCR, and RDCR fields are all set to 0, the port defaults to the 16x mode of operation.
D16	R/W	TICS	0	**Transmit internal clock source** 0 Transmitter uses the bit-rate generator output for its clock. 1 Transmitter uses the extracted clock provided by DPLL. Defines the transmit clock source when the TXSRC (D28) field is set to 0. There are two sources for internal clocks: the bit-rate generator (BRG) and the receiver digital phase lock loop (DPLL). The bit-rate generator uses a divider mechanism for clock generation. The DPLL extracts the clock from the incoming receive data stream.

Table 387: Serial Channel B/A/C/D Bit-rate register

Bits	Access	Mnemonic	Reset	Description
D15	R/W	RICS	0	**Receive internal clock source** 0 Receiver uses the bit-rate generator output for the clock. 1 Receiver uses the extracted clock provided by the DPLL. Defines the receive clock source when the RXSRC (D29) field is set to 0. There are two sources for internal clocks: the bit-rate generator (BRG) and the receiver digital phase lock loop (DPLL). The bit-rate generator uses a divider mechanism for clock generation. The DPLL extracts the clock from the incoming receive data stream.
D14:00	R/W	N	0x0000	**Divisor value** Defines the divisor value used in the bit-rate generator to determine effective frequency of the bit-rate generator. The divisor value for SPI master (synchronous operation) is defined as follows: $N = ((F_{CLK}/2 * DR) - 1)$ where: ■ FCLK = Determined by CLKMUX field ■ DR = DataRate = Required data rate For SPI slave mode, N is not used. Bit rate is defined by incoming clock edges.

Table 387: Serial Channel B/A/C/D Bit-rate register

Serial Channel B/A/C/D FIFO Data register

Address: 9020 0010 / 0050
9030 0010 / 0050

The Serial Channel B/A/C/D FIFO Data registers manually interface with the serial controller FIFOs instead of using DMA support.

Writing to the transmit register loads the transmit FIFO. This register can be written only when the TRDY field is set in Serial Channel Status Register A. Writing to the Serial Channel FIFO Data register automatically clears the TRDY bit.

Reading from the receive register empties the receive FIFO. Data is available when the RRDY bit is set in Serial Channel Status Register A. The RXFDB field in Serial Channel Status Register A identifies how many bytes are available to be read. Reading the Serial Channel FIFO Data register automatically clears the RRDY bit in Serial Channel Status Register A.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	DATA	0x00000000	Serial channel FIFO data field.

Table 388: Serial Channel B/A/C/D FIFO Data register

IEEE 1284 Peripheral Controller

C H A P T E R 1 5

The IEEE 1284 peripheral port supports compatibility mode, nibble mode, byte mode, and ECP mode of operations as a peripheral device. The IEEE 1284 port does not support EPP/mode daisy chain or multiplexer operations.

Requirements

Two components are required to run the IEEE 1284 peripheral-to-host interface:

- **Clock divider.** Required to generate the 1284-port operating clock from the BBus clock. The operating range of the port clock typically is 100 KHz–2 MHz. The clock divider is set using the granularity counter (see "Granularity Count register" on page 702).
- **External transceivers.** The data flow direction control is provided using a GPIO pin under software control.

Overview

Figure 95 shows the block diagram of the IEEE 1284 peripheral port control module.

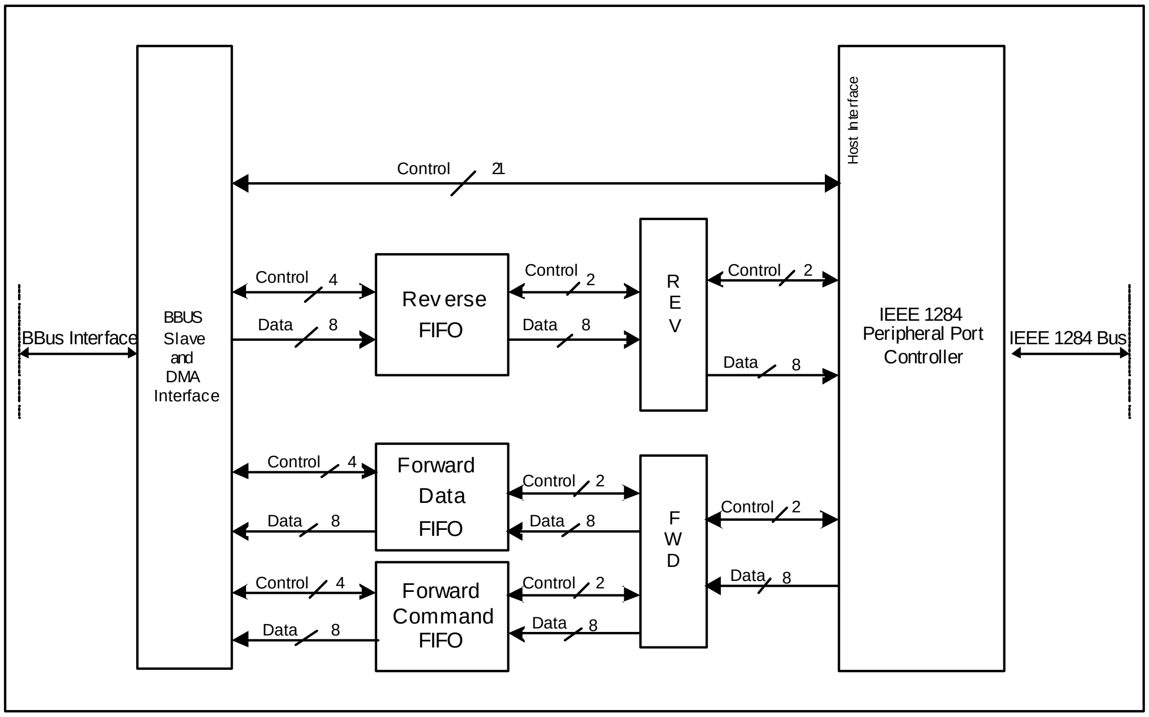


Figure 95: IEEE 1284 peripheral port control module

Note: Traffic direction in the IEEE 1284 is classified as either forward or reverse. The forward direction is equivalent to NS9750 receive. Similarly, the reverse direction is equivalent to NS9750 transmit.

Compatibility mode

Compatibility mode is the standard parallel port (SPP) forward transmission mode (from the host), also known as the *Centronics* mode. The incoming data is routed into the FORWARD DATA IN FIFO, an acknowledge signal is generated by the 1284 peripheral, and DMA requests are issued to the BBus until the FORWARD DATA IN FIFO is empty.

Figure 96 shows the timing relationship on the port interface.

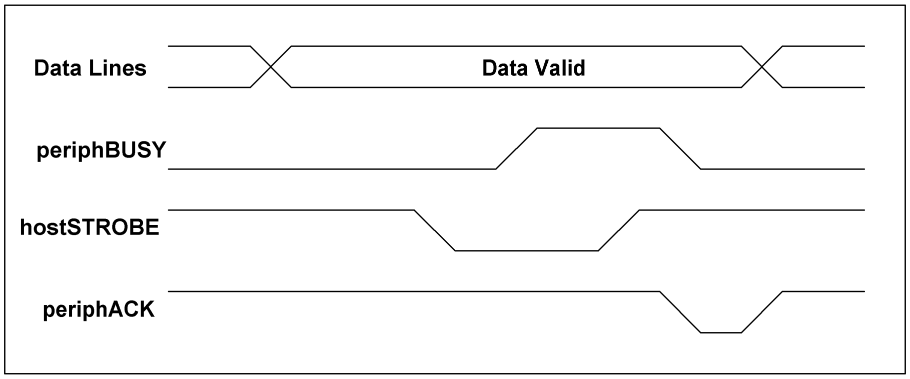


Figure 96: Compatibility mode data transfer cycle

Nibble mode

Nibble mode can send a byte of information to the host by sending two nibbles. This mode operates only in reverse mode.

Figure 97 shows the timing relationship on the port interface.

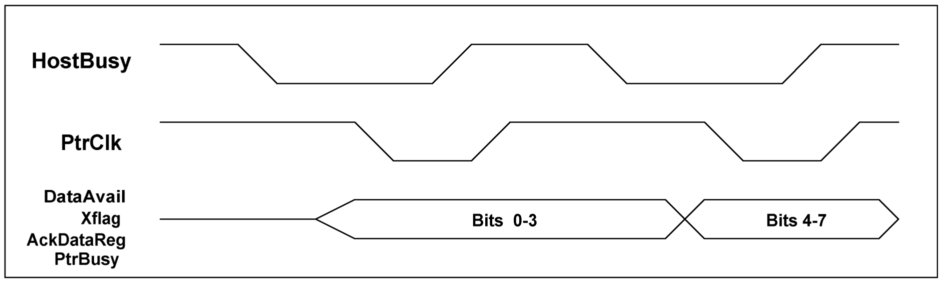


Figure 97: Nibble mode data transfer cycles

Byte mode

Byte mode sends information to the host over the data lines, at 8 bits per cycle. The peripheral sets the PtrClk bit high to acknowledge the host.

Figure 98 shows the timing relationship on the port interface.

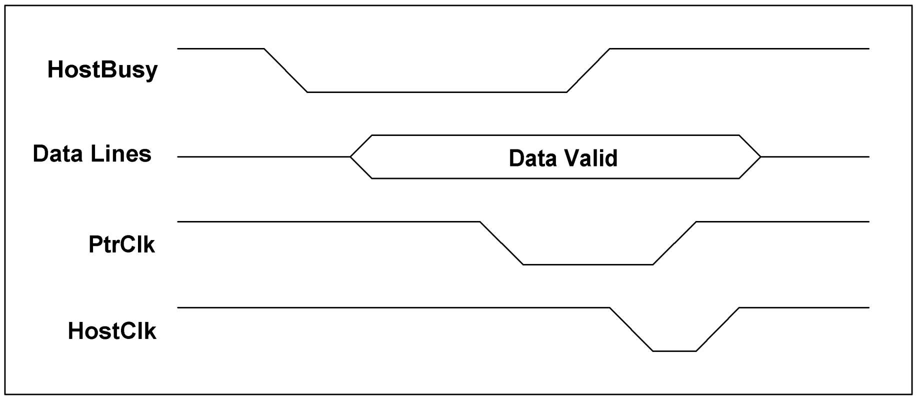


Figure 98: IEEE 1284 byte mode data transfer cycle

ECP mode

ECP (extended capability port) mode provides a high performance bi-directional communication path between the host and the peripheral. The ECP protocol provides two cycle types in both the forward and reverse directions: data cycles and command cycles.

Two types of command cycles are supported by the IEEE peripheral: run length count and channel address. The transfer direction is controlled by the host until a ReverseRequest signal is issued by the host. The peripheral can set the PeriphRequest signal low to indicate that reverse data is available.

Run_Length_Encoding (RLE) data compression enables real time data compression that can achieve ratios up to 64:1. NS9750 uses RLE decoding to enable large raster images with large strings of identical data to be transferred to system memory.

Forward transfer cycles

Be advised: In forward ECP mode, the PeriphAck signal indicates to the host the ability of the peripheral to accept data. At event #6 of the negotiation process, the NS9750 indicates to the host that it cannot accept data and holds its PeriphAck signal high for approximately 1200ns.

To differentiate the data cycles from the command cycles, the host sets the HostAck signal at the beginning of the cycle. When HostAck is asserted low, a command cycle is occurring and the data represents either an RLE count or a channel address. Bit 7 of the data byte indicates what is represented:

- If bit 7is 0, the data is an RLE count and (bits[6:0] + 1) versions of the subsequent byte are placed into the appropriate forward FIFO based on the value of HostAck while it is being transferred.
- If bit 7 is 1, the data is a channel address (0–127), and bits [6:0] are written into the Forward Address register ("Forward Address register" on page 703).

Figure 99 shows a data cycle followed by a command cycle.

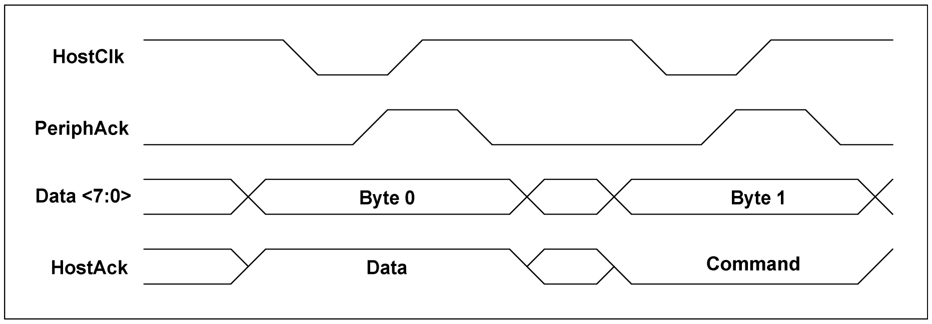


Figure 99: ECP mode forward transfer cycles

▶ **Host processing sequence example:**

1 The host puts the data on the data lines and indicates a data cycle by setting HostAck high.

2 The host asserts HostClk low to indicate valid data.

3 The peripheral acknowledges the host by setting PeriphAck to high.

4 The host sets HostClk to high. This edge should be used to latch the data into the peripheral.

5 The peripheral sets PeriphAck low, indicating that it's ready for the next data byte.

6 The host sets HostAck to low to start the command transfer.

Note: A forward transfer does not have to be a data transfer followed by a command transfer. When HostClk is low, the transfer will be data or command depending on whether HostAck is high or low.

Reverse transfer cycles

With the ECP protocol, changes in the data direction must be negotiated.

The host must request a reverse channel transfer by asserting the ReverseRequest signal. The host waits for the peripheral to acknowledge the request by asserting the AckReverse signal. Only then can a reverse channel data transfer take place.

Figure 100 shows a reverse channel data cycle followed by a command cycle. In this case, PeriphClk is the data strobe and HostAck is the acknowledge signal.

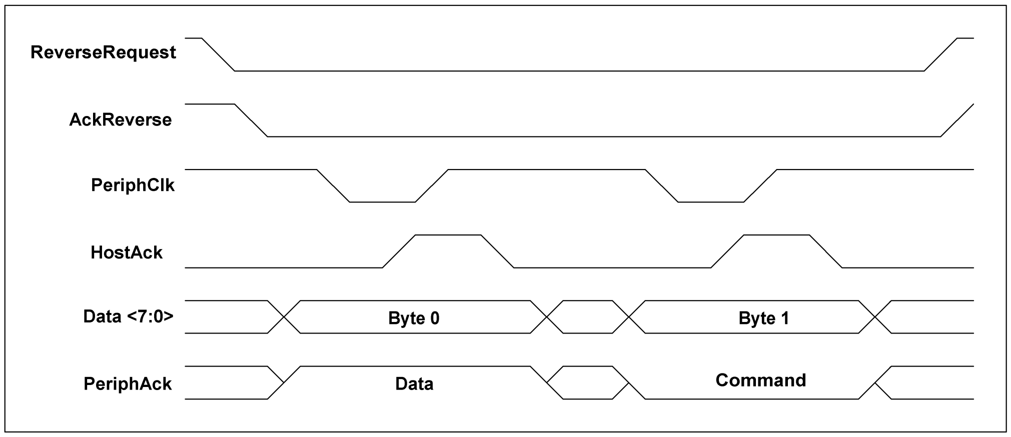


Figure 100: ECP reverse channel transfer cycles

Data and command FIFOs

Separate data and command FIFOs are provided in the forward direction, and a single FIFO is provided in the reverse direction.

These FIFOs can be accessed either through the appropriate DMA channel (see Table 310: "DMA channel assignments" on page 509) or directly by the CPU using access registers provided in this 1284 interface.

- Reading or writing these direct access FIFO registers with DMA control selected results in a bus error.
- Direct access registers, as well as all 1284 registers, use little endian byte ordering, where byte 3 [31:24] is the most significant byte and byte 0 [7:0] is the least significant byte. Before accessing these registers through the CPU, however, you must first indicate the endianness of the AHB to the 1284 peripheral. Do this using the Endian Configuration register in the BBus utility.
- For normal operation, it is recommended that you configure this 1284 interface for DMA control. DMA provides a faster and more efficient interface between IEEE 1284 and the rest of the NS9750. CPU mode is more suitable for diagnostic and testing purposes.

The forward command FIFO is provided solely for the user's benefit, to pass any user-defined, non-IEEE 1284 compliant commands from the host to the NS9750.

Because the NS9750 functions only as a slave, it is not necessary to provide the capability of driving any non-IEEE 1284 compliant commands back to the host.

Important: The 1284 commands are not designed to be stored and passed along. To store a non-IEEE 1284 command in the forward command FIFO, send an RLE command of count one (0x0), followed by the command you want to send. Both bytes must be transferred while HostAck is low and bit[14] of the IEEE 1284 General Configuration register (see page 679) is set to 0.

IEEE 1284 negotiation

The negotiation process is a mechanism by which the host determines the capabilities of the attached peripheral. The module can be programmed to interrupt the software when the host begins negotiation. The module automatically completes negotiation into byte, nibble, and ECP modes. The host uses an extensibility byte to communicate to the module which mode is being negotiated into.

Warning: During a negotiation into either nibble or byte mode, the behavior of the pError signal driven by the NS9750 is out of compliance with the IEEE 1284 standard. The pError signal does not match the value of nFault at event #6 of the negotiation process.

Table 389 defines the extensibility byte values.

Extensibility byte	Definition	Description
1000 0000	Reserved	Reserved
0100 0000	Reserved	Reserved
0011 0000	Request ECP mode with RLE	
0001 0000	Request ECP mode without RLE	
0000 1000	Reserved	Reserved
0000 0100	Request device ID using nibble mode	Receive the device ID a nibble at a time across the status lines.
0000 0101	Request device ID using byte mode	Receive the device ID a byte at a time across the data lines.
0001 0100	Request device ID using ECP mode without RLE	Receive device ID without ECP data compression.

Table 389: Extensibility byte values

Extensibility byte	Definition	Description
0011 0100	Request device ID using ECP mode with RLE	Receive device ID with ECP data compression.
0000 0010	Reserved	Reserved
0000 0001	Byte mode reverse channel transfer	
0000 0000	Nibble mode reverse channel transfer	

Table 389: Extensibility byte values

The NS9750 directly supports RLE compression. The device ID can be returned in any supported reverse channel mode. The device ID is a length field followed by a string of ASCII characters that define the peripheral's characters and/or capabilities.

BBus slave and DMA interface

The BBus slave and DMA interface module controls accesses from the BBus to the IEEE peripheral. The interface can operate in two modes: DMA and CPU.

- In DMA mode, three BBus DMA channels are used for forward data and forward command traffic, and all reverse traffic.
- In CPU mode, the CPU can access the forward data, forward command, and reverse FIFOs directly.

BBus slave and DMA interface register map

The IEEE 1284 module uses the control and status registers listed in Table 390.

All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Address	Register	Description
9040 0000	GenConfig	General Configuration register
9040 0004	InterruptStatusandControl	Interrupt Status and Control register

Table 390: 1284 Control and Status registers

Address	Register	Description
9040 0008	FIFO Status	FIFO Status register
9040 000C	FwdCmdFifoReadReg	Forward Command FIFO Read register
9040 0010	FwDatFifoReadReg	Forward Data FIFO Read register
9040 0014 – 9040 0018		Reserved
9040 001C	RvFifoWriteReg	Reverse FIFO Write register
9040 0020	RvFifoWriteReg - Last	Reverse FIFO Write Register - Last
9040 0024	FwdCmdDmaControl	Forward Command DMA Control register
9040 0028	FwDatDmaControl	Forward Data DMA Control register
9040 0100 – 9040 017C CSRs (8-bit wide)		
9040 0100	pd	Printer Data Pins register
9040 0104	psr	Port Status register (host)
9040 0108	pcr	Port Control register
9040 010C	pin	Port Status register (peripheral)
9040 0110	Reserved	
9040 0114	fea	Feature Control Register A
9040 0118	feb	Feature Control Register B
9040 011C	fei	Interrupt Enable register
9040 0120	fem	Master Enable register
9040 0124	exr	Extensibility Byte Requested by Host — UART and SPI
9040 0128	ecr	Extended Control register
9040 012C	sti	Interrupt Status register
9040 0130	Reserved	
9040 0134	msk	Pin Interrupt Mask register
9040 0138	pit	Pin Interrupt Control register
9040 013C – 9040 0164		Reserved
9040 0168	grn	Granularity Count register

Table 390: 1284 Control and Status registers

Address	Register	Description
9040 016C – 9040 0170		Reserved
9040 0174	eca	Forward Address register
9040 0178	pha	Core Phase register

Table 390: 1284 Control and Status registers

IEEE 1284 General Configuration register

Address: 9040 0000

The IEEE 1284 General Configuration register contains miscellaneous control settings for the IEEE 1284 module.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd	AFSH	CPS	Rsvd	FCRT		FDRT		Reserved		RRT		FCM	Rsvd	FDM	RM

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:15	N/A	Reserved	N/A	N/A
D14	R/W	AFSH	0x0	**HostAck signal handling** 0 HostAck=1: Forward data bits 7 to 0 are stored in data FIFO HostAck=0: Forward data bits 7 to 0 are stored in command FIFO 1 All forward data bits stored in data FIFO You can use the core interrupt capability to detect transitions on HostAck.

Table 391: IEEE 1284 General Configuration register

Bits	Access	Mnemonic	Reset	Description
D13	R/W	CPS	0x0	**Connector *PLH* signal** 0 Indicates to the host that this interface is not ready to operate as an IEEE 1284 slave. 1 Indicates to the host that this interface is ready to operate as an IEEE 1284 slave. This bit should be set by software when the initialization of the 1284 interface is complete.
D12	N/A	Reserved	N/A	N/A
D11:10	R/W	FCRT	0x3	**Forward command ready threshold** (FwCmdReadyThreshold)
D:09:08	R/W	FDRT	0x3	**Forward data ready threshold** (FwDatReadyThreshold) 00 4 bytes 01 8 bytes 10 16 or more bytes 11 28 or more bytes Enables transfer from the corresponding FIFO. DMA is inhibited until the FIFO contains the corresponding number of bytes. Data in the FIFO beneath the threshold is transferred only if the buffer gap timer is used.
D07:06	N/A	Reserved	N/A	N/A
D05:04	R/W	RRT	0x3	**Reverse ready threshold** (RvReadyThreshold) 00 1–4 bytes 01 5–8 bytes 10 13–16 bytes 11 29–32 bytes Enables transfer from the corresponding FIFO. DMA is inhibited until the FIFO can accept the corresponding number of bytes.
D03	R/W	FCM	0x0	**Forward command mode (FwdCmdMode)** 0 Direct CPU access 1 DMA control
D02	NA	Reserved	N/A	N/A
D01	R/W	FDM	0x0	**Forward data mode (FwdDataMode)** 0 Direct CPU access 1 DMA control

Table 391: IEEE 1284 General Configuration register

Bits	Access	Mnemonic	Reset	Description
D00	R/W	RM	0x0	**Reverse mode (RevMode)** 0 Direct CPU access 1 DMA control

Table 391: IEEE 1284 General Configuration register

Interrupt Status and Control register

Address: 9040 0004

The Interrupt Status and Control register contains miscellaneous control settings for the IEEE 1284 module. Bits with an access type of R/C (read/clear) can be set only by hardware, and are cleared by software by writing a 1 to the corresponding bit location. The software is expected to clear the condition causing the interrupt before clearing the interrupt; if the condition is not cleared, the bit and (maskable) interrupt will be reasserted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					RFRIM	Reserved		FDBGM	FCBGM	FDMBM	FCMBM	FDRIM	FCRIM	I1M	Rsvd

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RFRI	Reserved		FDFBG	FCFBG	FDFMB	FCFMB	FDFRI	FCFRI	PC1I	Rsvd

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:27	R	Reserved	0x0	N/A
D26	R/W	RFRIM	0x0	**Reverse FIFO ready interrupt mask** (RvFifoRdyInterruptMask) 0 Mask the interrupt 1 Enable the interrupt
D25:24	N/A	Reserved	N/A	N/A

Table 392: Interrupt Status and Control register

Bits	Access	Mnemonic	Reset	Description
D23	R/W	FDBGM	0x0	**Forward data FIFO byte gap mask** (FwDatFifoByteGapMask) 0 Mask the interrupt 1 Enable the interrupt
D22	R/W	FCBGM	0x0	**Forward command FIFO byte gap mask** (FwCmdFifoByteGapMask) 0 Mask the interrupt 1 Enable the interrupt
D21	R/W	FDMBM	0x0	**Forward data FIFO max buffer mask** (FwDatFifoMaxBufMask) 0 Mask the interrupt 1 Enable the interrupt
D20	R/W	FCMBM	0x0	**Forward command FIFO max buffer mask** (FwCmdFifoMaxBufMask) 0 Mask the interrupt 1 Enable the interrupt
D19	R/W	FDRIM	0x0	**Forward data FIFO ready interrupt mask** (FwDatFifoRdyInterruptMask) 0 Mask the interrupt 1 Enable the interrupt
D18	R/W	FCRIM	0x0	**Forward command FIFO ready interrupt mask** (FwCmdFifoRdyInterruptMask) 0 Mask the interrupt 1 Enable the interrupt
D17	R/W	I1M	0x0	**Peripheral controller interrupt 1 mask** 0 Mask the interrupt 1 Enable the interrupt
D16:11	N/A	Reserved	N/A	N/A
D10	R/C	RFRI	0x1	**Reverse FIFO ready interrupt (RvFifoRdyInterrupt)** Asserted when the reverse FIFO can accept the number of bytes specified in the reverse ready threshold bit in the IEEE 1284 General Configuration register.
D09	N/A	Reserved	0x1	N/A
D08	N/A	Reserved	0x0	N/A

Table 392: Interrupt Status and Control register

Bits	Access	Mnemonic	Reset	Description
D07	R/C	FDFBG	0x0	**Forward data FIFO byte gap** (FwDatFifoByteGap) The forward data byte gap timer expired and the buffer closed. Set to 1 to clear this bit.
D06	R/C	FCFBG	0x0	**Forward command FIFO byte gap** (FwCmdFifoByteGap) The forward command byte gap timer expired and the buffer closed. Set to 1 to clear this bit.
D05	R/C	FDFMB	0x0	**Forward data FIFO max buffer** (FwDatFifoMaxBug) The forward data maximum buffer length has been reached and the buffer closed. Set to 1 to clear this bit.
D04	R/C	FCFMB	0x0	**Forward command FIFO max buffer** (FwCmdFifoMaxBuf) The forward command maximum buffer length has been reached and the buffer closed. Set to 1 to clear this bit.
D03	R/C	FDFRI	0x0	**Forward data FIFO ready interrupt** (FwDatFifoRdyInterrupt) Contains data from the host. Set to 1 to clear this bit.
D02	R/C	FCFRI	0x0	**Forward command FIFO ready interrupt** (FwCmdFifoRdyInterrupt) Contains data from the host. Set to 1 to clear this bit.
D01	R/C	PC1I	0x0	**Peripheral controller interrupt 1** Read the peripheral controller Interrupt Status register to determine the source. Set to 1 to clear this bit.
D00	N/A	Reserved	0x1	N/A

Table 392: Interrupt Status and Control register

FIFO Status register

Address: 9040 0008

The FIFO Status register allows the CPU to determine that status of all FIFOs in the 1284 module. You can ignore this register when running the 1284 interface in DMA mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCFDR		FCFE	FCFA	FCFR	Reserved			FDFDR		FDFE	FDFAE	FDFR	RFF	RFAF	RFR

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A
D15:14	R	FCFDR	0x0	**Forward command FIFO depth remain** (FwCmdFifoDepthRemain) 00 4 bytes 01 1 byte 10 2 bytes 11 3 bytes Determines how many bytes are valid in the current forward command FIFO entry. The current value in the field is not valid if the FIFO is empty.
D13	R	FCFE	0x1	**Forward command FIFO empty** (FwCmdFifoEmpty) 0 FIFO is not empty 1 FIFO is empty
D12	R	FCFA	0x1	**Forward command FIFO almost empty** (FwCmdFifoAlmostEmpty) 0 FIFO has more than 1–4 bytes 1 FIFO has only one 1–4 byte entry This field is not valid if the FIFO is empty.

Table 393: FIFO Status register

Bits	Access	Mnemonic	Reset	Description
D11	R	FCFR	0x0	**Forward command FIFO ready** (FwCmdFifoReady) Asserted if forward command in FIFO is enabled to move data. Determined by FwCmdReadyThreshold (in the IEEE 1284 General Configuration register).
D10:08	N/A	Reserved	N/A	N/A
D07:06	R	FDFDR	0x0	**Forward data FIFO depth remain** (FwDatFifoDepthRemain) 00 4 bytes 01 1 byte 10 2 bytes 11 3 bytes Determines how many bytes are valid in the current forward data FIFO entry. The current value in the field is not valid if the FIFO is empty.
D05	R	FDFE	0x1	**Forward data FIFO empty** (FwDatFifoEmpty) 0 FIFO is not empty 1 FIFO is empty
D04	R	FDFAE	0x0	**Forward data FIFO almost empty** (FwDatFifoAlmostEmpty) 0 FIFO has more than 1–4 bytes 1 FIFO has only one 1–4 byte entry This field is not valid if the FIFO is empty.
D03	R	FDFR	0x0	**Forward data FIFO ready** (FwDatFifoReady) Asserted if forward data in FIFO is enabled to move data. Determined by FwDatReadyThreshold (in the IEEE 1284 General Configuration register).
D02	R	RFF	0x0	**Reverse FIFO full (RvFifoFull)** 0 FIFO is not full 1 FIFO is full

Table 393: FIFO Status register

Bits	Access	Mnemonic	Reset	Description
D01	R	RFAF	0x0	**Reverse FIFO almost full (RvFifoAlmostFull)** 0 FIFO can take more than 1–4 bytes 1 FIFO can take only one 1–4 byte entry This field is not valid id the FIFO is full.
D00	R	RFR	0x0	**Reverse FIFO ready (RvFifoReady)** Asserted if reverse data out FIFO is enabled to move data. Determined by RvDatReadyThreshold (in the IEEE 1284 General Configuration register).

Table 393: FIFO Status register

Forward Command FIFO Read register

Address: 9040 000C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FwCmdFifoReadReg															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FwCmdFifoReadReg															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R	FwCmdFifoReadReg	N/A	Reads up to four bytes from the Forward Command FIFO when in CPU mode. The CPU must read the FIFO Status register (see page 684) to determine how many bytes are remaining before issuing the read.

Table 394: Forward Command FIFO Read register

Forward Data FIFO Read register

Address: 9040 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FwDatFifoReadReg															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FwDatFifoReadReg															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R	FwDatFifoReadReg	N/A	Reads up to four bytes from the Forward Data FIFO when in CPU mode. The CPU must read the FIFO Status register (see page 684) to determine how many bytes are remaining before issuing the read.

Table 395: Forward Data FIFO Read register

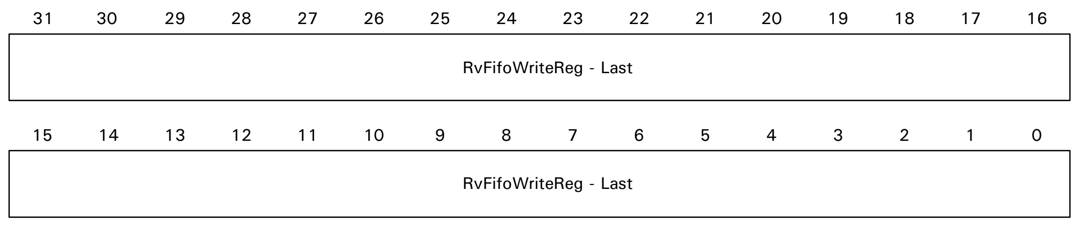
Reverse FIFO Write register/Reverse FIFO Write Register — Last

Address: 9040 001C / 9040 0020

Both registers are 32 bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RvFifoWriteReg															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RvFifoWriteReg															



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	W	RvFifoWriteReg	N/A	Write one to four bytes to the Reverse FIFO when in CPU mode. ■ A FIFO entry containing one byte or two bytes is written to Reverse FIFO Write Register — Last. ■ A FIFO entry containing three bytes is written in two steps: Step 1: The lowest 16 bits are written to the Reverse FIFO Register. Step 2: The high byte is written to the Reverse FIFO Write Register — Last. ■ A FIFO entry containing four bytes is written to either register.
D31:00	W	RvFifoWrite Reg — Last	N/A	

Table 396: Reverse Data FIFO Write register/Reverse Data FIFO Write Register — Last

Forward Command DMA Control register

Address: 9040 0024

The Forward Command DMA Control register controls when the Forward command DMA buffer is closed, using two components:

- **16-bit maximum buffer counter.** The maximum buffer counter increments each time a DMA transfer occurs, by the number of bytes in the transfer. The counter is reset each time a DMA is completed. If the counter reaches or exceeds the forward command maximum buffer size (FwCmdMaxBufSize), the 1284 module signals the DMA channel to close the buffer and start a new one. A (maskable) interrupt is generated when FwCmdMaxBufSize is reached. Future bytes are moved using DMA when the next DMA is initiated by the DMA controller.

  Note: This counter should not be set to a value greater than the buffer length field value set in the 1284 forward command channel descriptor.

- **16-bit byte gap counter.** The byte gap counter increments on each clock cycle when a byte is not read from the host, with a maximum programmable interval of 1.3 ms based on a 50 MHz BBus clock. The counter is reset when a byte is read from the host. If the counter reaches the forward command byte gap timeout (FwCmdByteGapTimer), the following occurs:

  a Where the FIFOs are written with dwords containing four bytes each, the gap timeout forces an incomplete dword (that is, 1–3 bytes) to be written to the FIFO.

  b Forward command FIFO ready, which usually means the threshold has been met, is asserted. This results in continuation of the currently active DMA until the FIFO is empty.

  c When the data in the FIFO, including the incomplete dwords in Step 1, is output through DMA, the DMA is terminated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FwCmdMaxBufSize															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FwCmdByteGapTimer															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	R/W	FwCmdMaxBufSize	0x0	**Forward command maximum buffer size** Maximum buffer size in bytes.
D15:00	R/W	FwCmdByteGapTimer	0x0	**Forward command byte gap timeout** 16-bit byte gap timer in BBus clock cycles.

Table 397: Forward Command DMA Control register

Forward Data DMA Control register

Address: 9040 0028

The Forward Data DMA Control register controls when the forward data DMA buffer is closed, using two components:

- **16-bit maximum buffer counter.** The maximum buffer counter increments each time a DMA transfer occurs, by the number of bytes in the transfer. The counter is reset each time a DMA completes. If the counter reaches or exceeds the forward data maximum buffer size (FwDatMaxBufSize), the 1284 module signals the DMA channel to close the buffer and start a new one. A (maskable) interrupt is generated when FwDatMaxBufSize is reached. Future bytes are moved using DMA when the next DMA is initiated by the DMA controller.

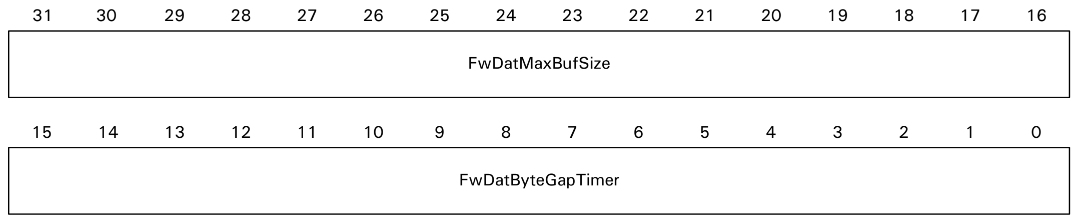
  Note: This counter should not be set to a value greater than the buffer length field value set in the 1284 forward data DMA channel descriptor.

- **16-bit byte gap counter.** The byte gap counter increments on each clock cycle when a byte is not read from the host, with a maximum programmable interval of 1.3 ms based on a 50 MHz BBus clock. The counter is reset when a byte is read from the host. If the counter reaches the forward data byte gap timeout (FwDatByteGapTimer), the following occurs:

  a Where the FIFOs are written with dwords containing four bytes each, the gap timeout forces an incomplete dword (that is, 1–3 bytes) to be written to the FIFO.

b Forward data FIFO ready, which normally means the threshold has been met, is asserted. This results in continuation of the currently active DMA until the FIFO is empty.

c When the data in the FIFO, including the incomplete dwords in Step 1, is output through DMA, the DMA is terminated.



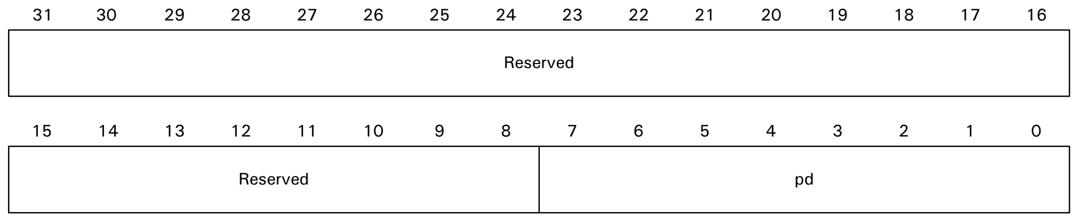
Register bit assignment

Bits	**Access**	**Mnemonic**	**Reset**	**Description**
D31:16	R/W	FwDatMaxBufSize	0x0	**Forward data maximum buffer size** Maximum buffer size in bytes.
D15:00	R/W	FwDatByteGapTimer	0x0	**Forward data byte gap timeout** 16-bit byte gap timer in BBus clock cycles.

Table 398: Forward Data DMA Control register

Printer Data Pins register

Address: 9040 0100



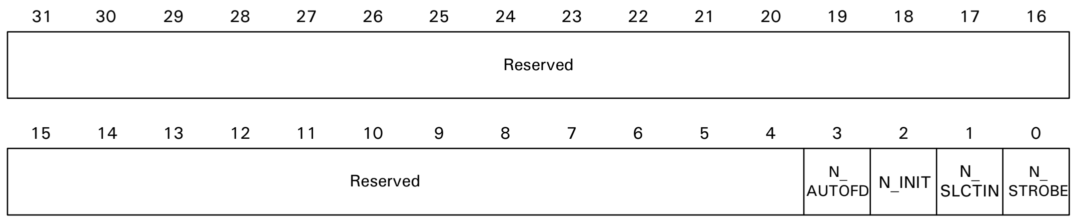
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A
D07:00	R	pd	N/A	**Printer data pins** Allows the CPU to read the status of the 8-bit data bus directly.

Table 399: pd — Printer Data Pins register

Port Status register, host

Address: 9040 0104



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	N/A	Reserved	N/A	N/A
D03	R	N_AUTOFD	N/A	Allows the CPU to read the status of the host control pins directly. The meaning of each bit varies, depending on whether the mode is compatibility, nibble, byte, or ECP.
D02	R	N_INIT	N/A	
D01	R	N_SLCTIN	N/A	
D00	R	N_STROBE	N/A	

Table 400: psr — Port Status register, host

Port Control register

Address: 9040 0108

Note: The Port Control register can control IEEE 1284 pins *only* if no modes are enabled in the Master Enable register (see"Master Enable register" on page 697).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BUSY	N_ACK	PERR	SEL	N_FLT	Reserved		

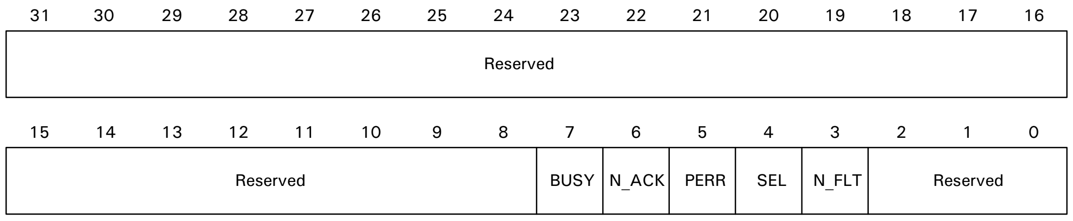
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A
D07	R/W	BUSY	0x0	Allows the CPU to directly control the IEEE 1284 pin values. The meaning of each bit varies, depending on whether the mode is compatibility, nibble, byte, or ECP. **Notes:** ■ Bits [07:03] should be set to 1 before the printer is enabled (ecr[0]='1') to avoid driving IEEE 1284 pins during initialization. ■ AutoNegotiate and AutoTransfer must be turned off before software can take control of these control signals (see "Master Enable register" on page 697).
D06	R/W	N_ACK	0x0	
D05	R/W	PERR	0x0	
D04	R/W	SEL	0x0	
D03	R/W	N_FLT	0x0	
D02:00	N/A	Reserved	N/A	N/A

Table 401: pcr — Port Control register

Port Status register, peripheral

Address: 9040 010C



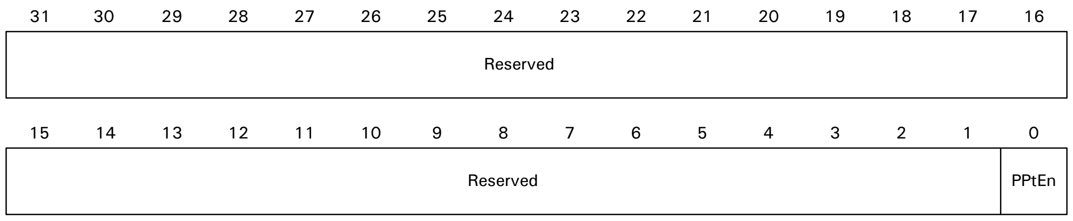
Bits	Access	Mnemonic	Reset	Description
D31:08	R	Reserved	0x0	N/A
D07	R	BUSY	0x0	Allows the CPU to read the status of the peripheral control pins directly. The meaning of each bit varies, depending on whether the mode is compatibility, nibble, byte, or ECP.
D06	R	N_ACK	0x0	
D05	R	PERR	0x0	
D04	R	SEL	0x0	
D03	R	N_FLT	0x0	
D02:00	R	Reserved	0x0	N/A

Table 402: pin — Port Status register, peripheral

Feature Control Register A

Address: 9040 0114

Feature Control Register A enables buffer trigger levels for printer port operations.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:01	N/A	Reserved	N/A	N/A
D00	R/W	PPtEn	0x0	**Printer port enable** 0 Force IEEE 1284 outputs to high impedance 1 Enable normal operation, depending on mode

Table 403: fea — Feature Control Register A

Feature Control Register B

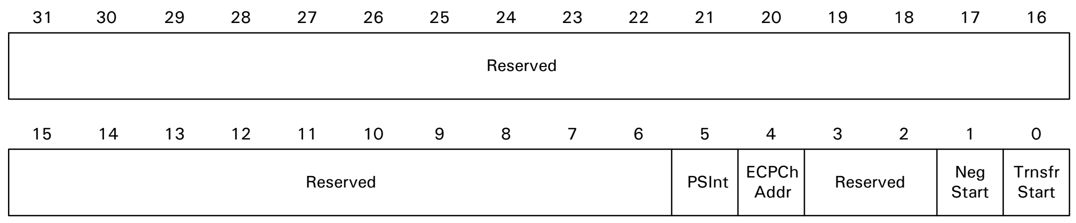
Address: 9040 0118

You must set bit[0] to 1 in Feature Control Register B. Bits[31:01] are reserved.

Interrupt Enable register

Address: 9040 011C

The Interrupt Enable register enables interrupts to be generated on certain conditions.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:06	N/A	Reserved	N/A	N/A

Table 404: fei — Interrupt Enable register

Bits	Access	Mnemonic	Reset	Description
D05	R/W	PinSelectInterrupt	0x0	**Pin select interrupt enable** 0 Disable 1 Enable
D04	R/W	ECPChannel Address	0x0	**Channel address update detect interrupt enable** 0 Disable 1 Enable
D03:02	N/A	Reserved	N/A	N/A
D01	R/W	NegotiationStart	0x0	**Negotiation start interrupt enable** 0 Disable 1 Enable This interrupt is triggered when the rising edge of SELECTIN is found while in compatibility mode.
D00	R/W	TransferStart	0x0	**Transfer start interrupt enable** 0 Disable 1 Enable This interrupt is triggered when the falling edge of STROBE is found while in compatibility mode.

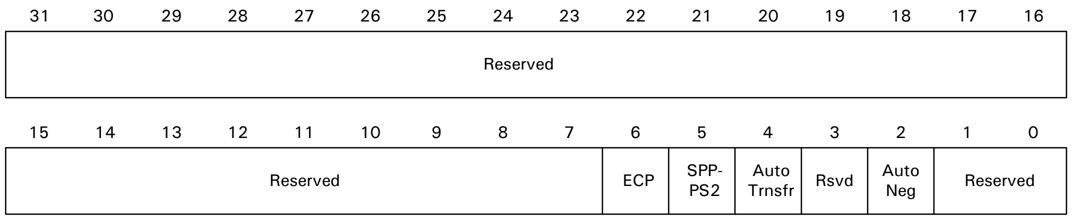
Table 404: fei — Interrupt Enable register

Master Enable register

Address: 9040 0120

The Master Enable register enables different IEEE 1284 modes and automatic transfer modes.

Note: Set both AutoTransfer and AutoNegotiate to enable hardware to control the 1284 peripheral interface signals.



Register bit assignment

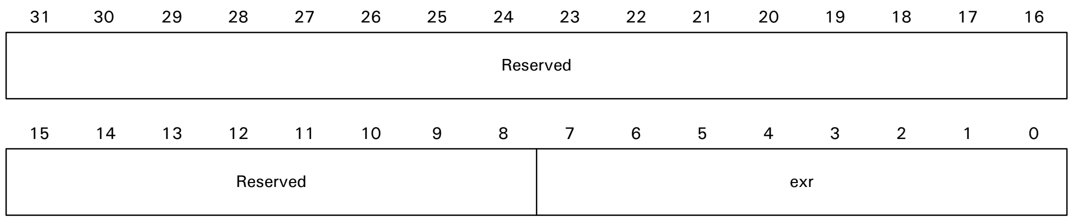
Bits	Access	Mnemonic	Reset	Description
D31:07	N/A	Reserved	N/A	N/A
D06	R/W	ECP	0x0	**ECP mode** 0 Disable 1 Enable
D05	R/W	SPP–PS2	0x0	**SPP–PS2 mode** 0 Disable 1 Enable
D04	R/W	AutoTransfer	0x0	**Auto transfer mode** 0 Disable 1 Enable
D03	N/A	Reserved	N/A	N/A
D02	R/W	AutoNegotiate	0x0	**Auto negotiate mode** 0 Disable 1 Enable
D01:00	N/A	Reserved	N/A	N/A

Table 405: fem — Master enable register

Extensibility Byte Requested by Host

Address: 9040 0124

This register is updated shortly after a new negotiation occurs (event 4 of the negotiation process; see the IEEE 1284 standard for more information).



Register bit assignment

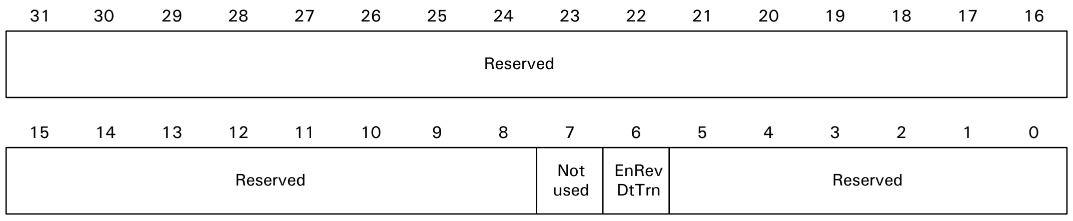
Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A
D07:00	R/W	exr	0x0	**Extensibility byte** Stores the extensibility byte received from the host.

Table 406: exr — Extensibility Byte Requested by Host register

Extended Control register

Address: 9040 0128

The Extended Control register enables additional core features.



Register bit assignment

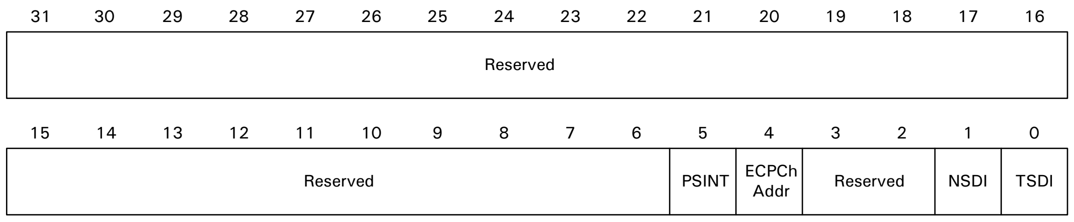
Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A
D07	N/A	Not used	0x0	Set to 1.
D06	R/W	Enable reverse data transfers	0x0	0 Disable 1 Enable
D05:00	N/A	Reserved	N/A	N/A

Table 407: ecr — Extended Control register

Interrupt Status register

Address: 9040 012C

Interrupts are cleared when this register is read. These interrupts are needed by software no matter which mode (DMA or CPU) is being used.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:06	N/A	Reserved	N/A	N/A
D05	R	PSINT	0x0	Pin select interrupt
D04	R	ECP Channel Address	0x0	Channel address update detect interrupt
D03:02	N/A	Reserved	N/A	N/A
D01	R	NSDI	0x0	Negotiation start detect interrupt

Table 408: sti — Interrupt Status register

Bits	Access	Mnemonic	Reset	Description
D00	R	TSDI	0x0	Transfer start detect interrupt

Table 408: sti — Interrupt Status register

Pin Interrupt Mask register

Address: 9040 0134

The Pin Interrupt Mask register enables IEEE 1284 pin interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												N_ AUTOFD edge det	N_INIT edge detect	N_ SLCTIN edge det	N_ STROBE edge det

Bits	Access	Mnemonic	Reset	Description
D31:04	N/A	Reserved	0x0	N/A
D03	R/W	n_autofd edge detect	0x0	0 Disable 1 Enable
D02	R/W	n_init edge detect	0x0	
D01	R/W	n_selectin edge detect	0x0	
D00	R/W	n_strobe edge detect	0x0	

Table 409: msk — Pin Interrupt Mask register

Pin Interrupt Control register

Address: 9040 0138

The Pin Interrupt Control register configures IEEE 1284 pin interrupt edge levels.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												n_ autofd edge	n_init edge	n_ slctin edge	n_ strobe edge

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	R	Reserved	0x0	N/A
D03	R/W	n_autofd edge	0x0	0 Falling edge 1 Rising edge
D02	R/W	n_init edge	0x0	
D01	R/W	n_slctin edge	0x0	
D00	R/W	n_strobe edge	0x0	

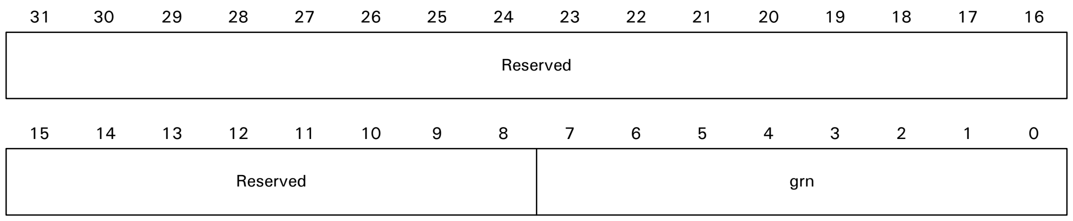
Table 410: pit — Pin Interrupt Control register

Granularity Count register

Address: 9040 0168

The Granularity Count register controls the value of the granularity counter for automatic processing modes.

Note: According to the IEEE 1284 standards spec, the peripheral has a T_p (500ns) minimum setup pulse width for some signals. If, for example, the BBus is set to run at 50 MHz (20ns clock period), the Granularity Count register should be set to 25 (0x19) [500ns / 20ns].



Bits	Access	Mnemonic	Reset	Description
D31:08	R	Reserved	0x0	N/A
D07:00	R/W	grn	0x0	**Granularity counter** Determines the number of BBus clock periods between peripheral signal changes on the IEEE 1284 bus.

Table 411: grn — Granularity Count register

Forward Address register

Address: 9040 0174

The Forward Address register is updated when a channel address command is received during a forward ECP transfer.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								eca							

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	R	Reserved	0	N/A
D07:00	R	eca	0	Forward address.

Table 412: eca — Forward Address register

Core Phase (IEEE1284) register

Address: 9040 0178

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								pha							

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	R	Reserved	0x0	N/A
D07:00	R	pha	0x0	0x00spp forward idle
				0x01 spp forward data transfer
				0x0f spp reset
				0x14 negotiate phase
				0x18 terminate phase
				0x24 nibble/byte reverse idle
				0x26 nibble/byte reverse data transfer (see **Warning** below**)**
				0x28 nibble/byte host busy data not available
				0x2C nibble/byte host busy data available
				0x2E nibble/byte host interrupt
				0x30 ecp forward idle
				0x31 ecp forward data transfer
				0x34 ecp reverse idle
				0x36 ecp reverse data transfer
				0x38 ecp host recovery
				0x3C ecp reverse to forward phase transition
				0x3E ecp forward to reverse phase transition
				0x3F ecp setup phase

Table 413: Core Phase register

Warning: The NS9750 cannot proceed to the Host Busy Data Available protocol state directly from negotiation state. If the host software supports the 1284 protocol reverse idle and interrupt host states (events 7, 18 – 20), the impact is a one-time loss of approximately 2,000ns at the beginning of data transfer. If the host does not support these states, however, the NS9750 will be unable to transmit data in nibble or byte mode. Note that there is no impact for a host that fully implements IEEE 1284.

USB Controller Module

C H A P T E R 1 6

USB 2.0 provides a standard “plug-and-play” interface for desktop communications at low to moderate speeds. The USB module in NS9750 supports both full-speed (12Mbps) and low-speed (1.5 Mbps) operation.

Overview

USB consists of point-to-point connections between one host and any number of hubs and devices; the number of hubs and devices combined cannot exceed 127. Point-to-point connections are established between a *downstream* port and an *upstream* port, as shown:

Host/hub/device	Description
Host	■ Has one downstream port ■ If has multiple downstream ports, constitutes multiple *networks*
Hub	■ Has one upstream port and multiple downstream ports ■ Connects to devices or other hubs using downstream ports ■ Connects to hosts or other hubs using upstream ports
Device	■ Has one upstream port ■ Connects to hubs or hosts

USB module architecture

The USB module is comprised of a separate host block and device block (see Figure 101). You can enable the device block or the host block — but never both at the same time.

The device block provides the resources for 12 logical endpoints that can be reconfigured by the device driver. The device block includes a DMA function to preload data for *in* endpoints, stuff data for *out* endpoints into system memory, and stuff data for some packets into system memory.

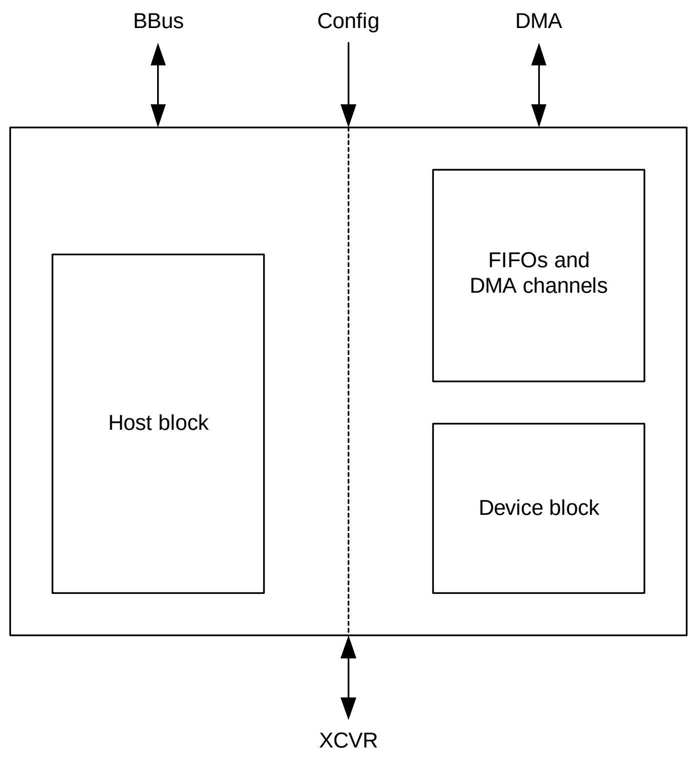


Figure 101: USB module architecture

The device block handles most packets that contain control and/or configuration information requiring device driver interaction to maintain only the DMA descriptor lists. The device block provides the appropriate handshake responses to USB. Control commands that require a response require the device driver to prepare a response based on a unique identifier supplied by the device block. Control commands not handled by the device block require the device driver to inspect the data provided by the setup phase of the packet, construct the required response, and initiate the DMA of the response data to the device block. The initial "try" of such commands typically is greeted with a NAK due to the lack of an available response. Later retries eventually result in the transmission of the response. Packets that contain data or require data transfer are handled by a multi-channel DMA function in the device block. Each pipe is associated with a specific DMA channel. If the DMA channel for a given pipe has not been configured or updated, or otherwise is not ready to send or receive the required data, the device block issues NAKs to the USB.

The host block supports the Open Host Controller Interface (OHCI) interface model for USB communications. The host block provides the registers and BBus mastering capability to traverse transaction lists in system memory per the OHCI interface, as well as transferring the in/out data to and from system memory.

USB device block

Figure 102 shows the architecture of the USB device block:

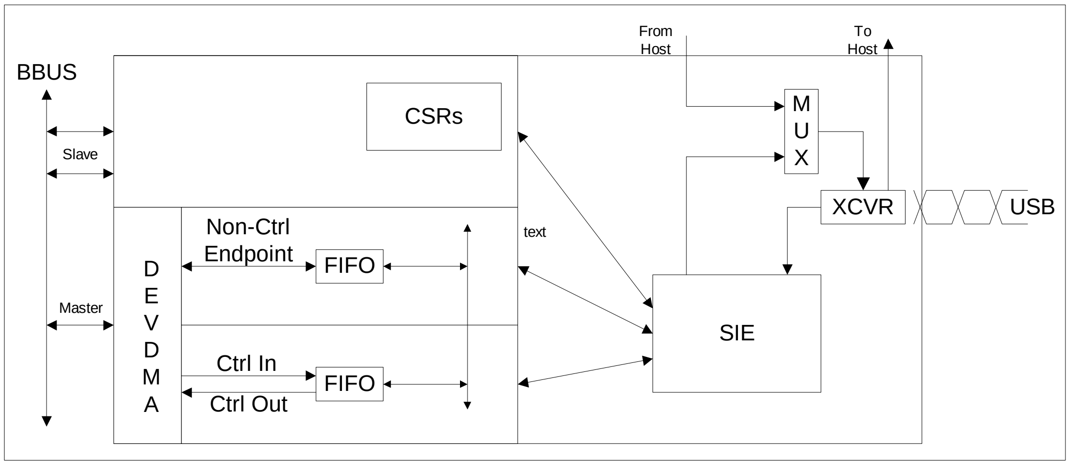


Figure 102: USB device block

Control and status

The control and status for the USB device block is provided through the slave interface. The device block contains the registers used to program each of the 12 logical endpoints and the five configurations.

Packet and data flow

The device block responds to packets initiated by the host. There are 16 DMA channels in this block. DMA channels 1 and 2 move data to and from system memory for control transfers for logical endpoint 0. DMA channels 3 through 13 move data for logical endpoints 1 through 11, respectively. DMA channels 14 through 16 are not used in the USB module.

Data for each logical endpoint travels through a four-word FIFO. Logic included with the FIFOs interface the Master I/F to the DMA channels. The serial interface engine (SIE) performs the tasks required to receive and send packets to and from a USB host.

The device driver writes the endpoint descriptor, configuration identifier register, string identifiers, DMA parameters, and the direction of data transmission for logical endpoints 1–11. From that point, the device driver monitors packet reception/ transmission success, and maintains the DMA descriptor lists. DMA Channel 1 DMA descriptors are *incoming*; that is, they move data from USB to memory. DMA channel 2 DMA descriptors are *outgoing*; that is, they move data from memory to the USB. The remainder of the DMA descriptors must follow the direction of data flow for the corresponding logical endpoint, which is also specified in the control/status register for the channel.

When a packet is received, a DMA descriptor must be available to move the required data; otherwise, the packet fails. If a descriptor remains unavailable, the FIFOs will be exhausted and an underrun or overrflow condition occurs. Bits in the corresponding FIFO Status register are set. DMA channels can be used to signal a DMA with no descriptors available.

Control in packets

Control in packets require device driver intervention. An “identifier” is stored in the status field of a retired DMA descriptor doe DMA channel 1. The device driver constructs a response and moves the response from system memory to the USB with DMA channel 2. The USB response to the initial reception of the control in packet is to store the identifier, then NAK the packet; this causes the host to retry the packet. If the outgoing data has been moved to the USB, the data is output upon retry. Once a control in packet has been received, all control in packets are NAK’d, and their identifiers are not stored. The first control in packet received following the successful transmission of data for the previous “pending” packet becomes the new “pending” packet.

Logical and physical endpoints

Each alternate of each interface of each configuration can use as many as 16 logical endpoints. Physical endpoints in the device block are programmable by software for configuration/interface/alternate number. type, direction, and MaxPacketSize. A physical endpoint, then, functions as one logical endpoint in one alternate of one interface of one configuration. The number of physical endpoints required is therefore equal to the total number of logical endpoints in each alternate of each interface of each configuration.

The device block is configured to hold 12 physical endpoints, endpoint 0, and 11 non-control endpoints, with a maximum of 5 interfaces. This allows simultaneous operation of the printer class, still image class, and three instances of the data class. Programming of the physical endpoints and the descriptors provided allow you to reassign any of the physical endpoints to any one of the five interfaces. A “multi-function” fax/printer scanner device can take the place of three instances of the data class if no more than six physical endpoints are required.

Slew rates

USB PHY has a greater than specified slew rate in low speed device mode. Faster slew rates may cause compatibility issues with low speed hosts.

Host block

The USB host block contains the resources to implement an OHCI host. These resources include a power control signal for USB port power. Figure 103 shows the architecture for the USB host block.

Control and status

The control and status for the USB host block is provided using the slave interface. The registers support the OHCI specification.

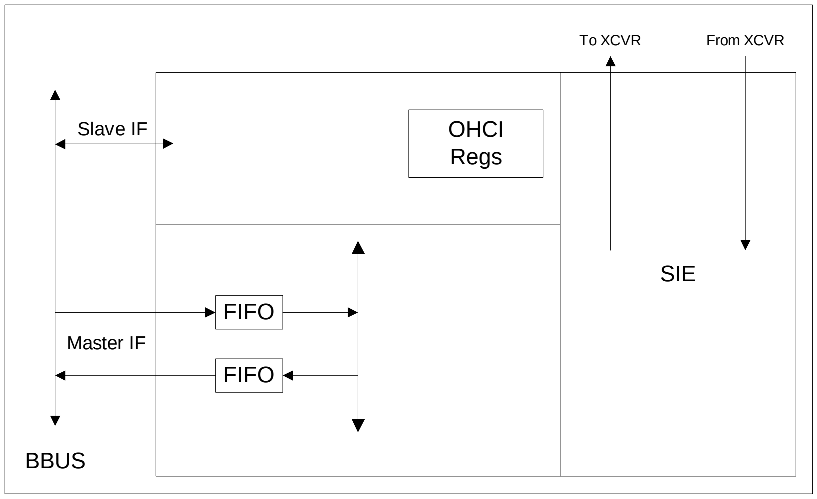


Figure 103: USB host architecture

Packet data flow

The host block initiates all transfers on the USB. Data travels through a four-word FIFO in either direction. The Serial Interface Engine (SIE) performs the tasks required to receive and send packets on the USB.

The host block is the master of the USB. At the highest level, the host maintains a database of the existing network topology, and can reconfigure any associated device.

At the next level, there are four linked lists of endpoint descriptors, each tied to one or more transfer descriptors. The four lists correspond to the four endpoint types that are accessed in a specified priority. The host begins processing these lists each millisecond, after sending the SOF packet. When a given endpoint descriptor is processed, the host generates a packet to move data and the transfer descriptor(s) defines the source or destination of the data in system memory. Interrupts generally are generated as each transfer descriptor is retired.

The core is connected with a master interface on the BBus to move incoming or outgoing data to system memory. Based on expected timing of four microseconds for bus turnaround, the four-word FIFOs included in the core prevent underrun and/or overrun occurrences. Bursting is supported to minimize the bus bandwidth that is consumed. Burst transfers move a maximum of 8 bytes in long word transactions. FIFO content from more than one transfer descriptor is broken into separate memory operations.

USB device endpoint

Any USB device requires a bidirectional control endpoint located at endpoint #0. The data FIFOs in the USB device application logic are unidirectional, so two data FIFOs are used to create a single bidirectional endpoint. FIFO #1 handles the USB-OUT packets and FIFO #2 handles the USB-IN packets. Table 414 describes the fields in the FIFO Status and Control registers that *must be set* in order for the bidirectional control endpoint to operate. The remaining Control register fields are application-specific.

FIFO	Mnemonic	Value	Description
1	TYPE	00	Control endpoint
1	DIR	0	USB-OUT direction
2	TYPE	00	Control endpoint
2	DIR	1	USB-IN direction

Table 414: Control endpoint register requirements

Transmission error handling

USB protocol supports the error-free transmission of packets for bulk and interrupt endpoints. The USB device application logic contains some control structures to help the software with this task. Transmission errors are defined as bit-stuffing errors or CRC check errors.

Handling USB-IN packet errors

USB-IN packet errors are sent from the USB device to the USB host. The USB host either responds with an ACK packet to indicate successful transmission or does not respond at all to indicate that there was an error in transmission.

These are the steps that must be taken to retransmit the packet in error:

1. Determine that the error has occurred through an interrupt. The hardware places the endpoint FIFO into reset state and flushes the contents.
2. Determine which endpoint is involved, using the FIFO Status registers.
3. Read the valid packets-transmitted-count from the Packet Control register and use it to compute a new buffer descriptor source address and buffer length. Update the current buffer descriptor in system memory. The current DMA buffer descriptor can be determined by reading the index field in the DMA Control register (see "DMA Control register" on page 514).
4. Write a 1 to the BDR (buffer descriptor refetch) and CE (channel enable) fields in the DMA Control register to force a refetch of the buffer descriptor when the channel next wins arbitration.
5. Take the USB device endpoint FIFO out of reset. At this point, the DMA controller will refetch the buffer descriptor as well as the packet in error and all remaining data in the DMA buffer.

Handling USB-OUT packet errors

USB-OUT packets are sent from the USB host to the USB device. The USB device either responds with an ACK packet to indicate successful transmission or does not respond at all to indicate that there was an error in transmission. If there was an error, the current DMA buffer is closed and the DMA buffer descriptor status field indicates that an error has occurred. Through this status information, software knows to discard the last packet in the buffer, as the USB host will retransmit the packet in error. The DMA controller fetches the next buffer descriptor, and the retransmitted packet will be the first packet in the new DMA data buffer.

USB block registers

The USB module configuration registers are located at base address 9010_0000. Table 415 provides the address register map for the USB “modules” within the USB block.

All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Address range	Register space (module)
0x90100000–0x90100FFF	USB Global Control and Status
0x90101000–0x90101FFF	USB Host Block
0x90102000–0x90102FFF	USB Device Block
0x90103000–0x90103FFF	USB Device Endpoint FIFOs
0x90110000–0x9011FFFF	USB Device DMA (see the BBus DMA Controller chapter for details about these registers)

Table 415: USB register address map

USB Global registers

Table 416 provides the addresses for the USB Global registers.

Address	Register
9010 0000	Global Control/Status register
9010 0004	Device Control/Status register
9010 000C	Global Interrupt Enable
9010 0010	Global Interrupt Status
9010 0014	Device IP Programming Control/Status

Table 416: USB Global registers address map

Global Control and Status register

Address: 9010 0000

The Global Control and Status register contains all USB global and status information. The USB can operate as a device or host, but cannot operate as both simultaneously unless used in *feedback mode*. Feedback mode is useful for development and testing only. The inactive block (host or device) is held in reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used			DSABL	HRST	DRST		Reserved			SUSP	Not used		WKUP	Not used	HST DV

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:13	N/A	Not used	0x00000	Always read as 0.
D12	R	DISABLE (DSABL)	0	**Disable** 0 Indicates that USB is enabled 1 Indicates that USB is disable Indicates whether USB is disabled or enabled.
D11	R	HRST	1	**Host reset** 0 Indicates that the host block is enabled 1 Indicates that the host block is in reset Provides the host block reset status.
D10:09	R	DRST	11	**Device reset** 00 Indicates that the device block is enabled 01 Indicates that the device block is in reset Provides the device block reset status.
D08:06	N/A	Reserved	N/A	N/A

Table 417: Global Control and Status register

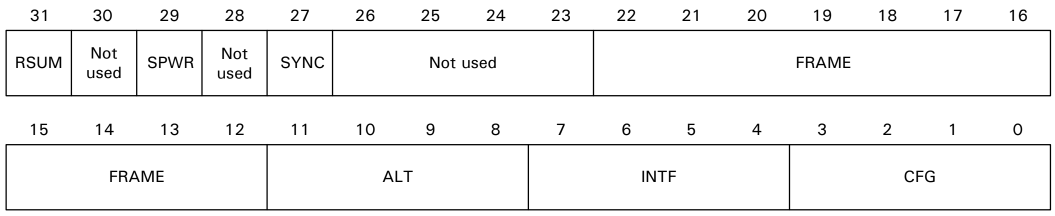
Bits	Access	Mnemonic	Reset	Description
D05	R	SUSP	0	**Suspend** 0 The USB device is not in a suspended state 1 The USB device is in a suspended state Indicates whether the USB device is in a suspended state.
D04:03	N/A	Not used	0	Always write a 0.
D02	R/W	WKUP	0	**Wakeup** Value depends on the value written to the HST_DEV field (bit 0 in this register). ■ When the NS9750 is in device mode, the device driver can write a 1 to this field to enable remote wakeup. ■ When the NS9750 is in host mode, this field is ignored.
D01	R	Not used	1	Always read as 1.
D00	R/W	HSTDV	1	**Host/device (HST_DEV)** 0 USB host mode 1 USB device mode Indicates the mode of the USB controller.

Table 417: Global Control and Status register

Device Control and Status register

Address: 9010 0004

The Device Control and Status register contains the USB device-block-specific control and status information.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	RESUME (RSME)	0	**Resume** Set to 1 by the device driver to initiate a resume sequence. This field is cleared to a 0 to end a resume sequence.
D30	N/A	Not used	0	Always read as 0.
D29	R/W	SPWR	0	**Self-powered (SELF_PWR)** This field should always be written as 1, since the NS9750 is always self-powered.
D28	N/A	Not used	0	Always write to 1.
D27	R/W	SYNC	0	**SYNC_FRAME support** Indicates whether the device block supports the SYNC_FRAME packet. 0 SYNC_FRAME packet is not supported 1 SYNC_FRAME packet is supported
D26:23	N/A	Not used	0	Always read as 0.
D22:12	R	FRAME	0x000	**Frame number** Contains the current frame number. **Note:** This field is used for diagnostic purposes only.
D11:08	R	ALT	0x0	**Alternate value** Contains the current alternate value for the device block from a SET INTERFACE packet.
D07:04	R	INTF	0x0	**Interface value** Contains the current interface value for the device block from a SET INTERFACE packet.
D03:00	R	CFG	0x0	**Configuration value** Contains the current configuration value for the device block from a SET CONFIGURATION packet.

Table 418: Device Control and Status register

Global Interrupt Enable register

Address: 9010 000C

The Global Interrupt Enable register contains the global interrupt enable information. *All interrupts are enabled by writing a 1 and disabled by writing a 0*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GLB_ EN	Not used			GBL_ DMA	DMA 13	DMA 12	DMA 11	DMA 10	DMA 9	DMA 8	DMA 7	DMA 6	DMA 5	DMA 4	DMA 3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA 2	DMA 1	Not used	FIFO	URST	SOF	SSPND	SET INTF	SET CFG	WAKE UP	Not used				OHCI_ IRQ	Not used

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	GBL_EN	0	**Global interrupt enable** Enables all interrupts. For normal operation, this field should be written to 1.
D30:28	R/W	Not used	0	Always write to 0.
D27	R/W	GBL_DMA	0	**Global DMA interrupt enable** Enables all DMA-generated interrupts. For normal operation, this field should be written to 1.
D26	R/W	DMA13	0	DMA channel 13 interrupt
D25	R/W	DMA12	0	DMA channel 12 interrupt
D24	R/W	DMA11	0	DMA channel 11 interrupt
D23	R/W	DMA10	0	DMA channel 10 interrupt
D22	R/W	DMA9	0	DMA channel 9 interrupt
D21	R/W	DMA8	0	DMA channel 8 interrupt
D20	R/W	DMA7	0	DMA channel 7 interrupt
D19	R/W	DMA6	0	DMA channel 6 interrupt
D18	R/W	DMA5	0	DMA channel 5 interrupt

Table 419: Global Interrupt Enable register

Bits	Access	Mnemonic	Reset	Description
D17	R/W	DMA4	0	DMA channel 4 interrupt
D16	R/W	DMA3	0	DMA channel 3 interrupt
D15	R/W	DMA2	0	DMA channel 2 interrupt
D14	R/W	DMA1	0	DMA channel 1 interrupt
D13	R/W	Not used	0	Always write to 0.
D12	R/W	FIFO	0	Generate an interrupt when any FIFO interrupt Status field is set and the corresponding interrupt is enabled using the FIFO Interrupt Enable register.
D11	R/W	URST	0	Generate an interrupt when the NS9750 is in device mode and receives an interrupt from the host.
D10	R/W	SOF	0	Generate an interrupt when the NS9750 is in device mode and receives an SOF (start of frame) packet.
D09	R/W	SUSPEND (SSPND)	0	Generate an interrupt when the SUSPEND (SUSP) field in the Global Interrupt Status register (see page 723) is asserted.
D08	R/W	SETINTF	0	Generate an interrupt when the SETINTF field in the Global Interrupt Status register (see page 723) is asserted.
D07	R/W	SETCFG	0	Generate an interrupt when the SETCFG in the Global Interrupt Status register (see page 723) is asserted.
D06	R/W	WAKEUP	0	Generate an interrupt when the WAKEUP field in the Global Interrupt Status register (see page 723) is asserted.
D05:02	N/A	Not used	0	Always write to 0.
D01	R/W	OHCI_IRQ	0	Generate an interrupt when the OHCI_IRQ field in the Global Interrupt Status register (see page 724) is asserted.
D00	N/A	Not used	N/A	Always write to 0.

Table 419: Global Interrupt Enable register

Global Interrupt Status register

Address: 9010 0010

The Global Interrupt Status register contains the global interrupt status information. *All status bits are active high (1) and all interrupts that are serviced here are cleared by writing a 1 to the appropriate field*.

For diagnostics, each bit serviced here can also be set to 1 by writing a 1 when the bit is set to 0.

Note: The DMA interrupts must be serviced in the USB DMA device block. The FIFO interrupts must be serviced in the FIFO Interrupt Status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used	Rsvd	Not used		GBL_ DMA	DMA 13	DMA 12	DMA 11	DMA 10	DMA 9	DMA 8	DMA 7	DMA 6	DMA 5	DMA 4	DMA 3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA 2	DMA 1	Rsvd	FIFO	URST	SOF	SSPND	SET INTF	SET CFG	WAKE UP	Reserved				OHCI_ IRQ	Rsvd

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	N/A	Not used	0	Always read as 0.
D30	N/A	Reserved	N/A	N/A
D29:28	N/A	Not used	0	Always read as 00.
D27	R	GBL_DMA	0	Bit-wise logical OR of the DMA# fields.
D26	R	DMA13	0	DMA channel 13 interrupt. Service in the USB DMA block.
D25	R	DMA12	0	DMA channel 12 interrupt. Service in the USB DMA block.
D24	R	DMA11	0	DMA channel 11 interrupt. Service in the USB DMA block.
D23	R	DMA10	0	DMA channel 10 interrupt. Service in the USB DMA block.
D22	R	DMA9	0	DMA channel 9 interrupt. Service in the USB DMA block.
D21	R	DMA8	0	DMA channel 8 interrupt. Service in the USB DMA block.
D20	R	DMA7	0	DMA channel 7 interrupt. Service in the USB DMA block.

Table 420: Global Interrupt Status register

Bits	Access	Mnemonic	Reset	Description
D19	R	DMA6	0	DMA channel 6 interrupt. Service in the USB DMA block.
D18	R	DMA5	0	DMA channel 5 interrupt. Service in the USB DMA block.
D17	R	DMA4	0	DMA channel 4 interrupt. Service in the USB DMA block.
D16	R	DMA3	0	DMA channel 3 interrupt. Service in the USB DMA block.
D15	R	DMA2	0	DMA channel 2 interrupt. Service in the USB DMA block.
D14	R	DMA1	0	DMA channel 1 interrupt. Service in the USB DMA block.
D13	N/A	Reserved	N/A	N/A
D12	R	FIFO	0	Bit-wise logical OR of the FIFO interrupt status fields.
D11	RW1TC	URST	0	Asserted when the NS9750 is in device mode and receives an interrupt from the host.
D10	RW1TC	SOF	0	Asserted when the NS9750 is in device mode and receives an SOF (start of frame) packet.
D09	RW1TC	SUSPEND SSPND	0	**Suspend** Asserted when either the device or the host has entered the suspend state.
D08	RW1TC	SETINTF	0	**Set interface** Asserted when the USB is configured for device operation and a set interface packet is received.
D07	RW1TC	SETCFG	0	**Set configuration** Asserted when the USB is configured for device operation and a set configuration packet is received.
D06	RW1TC	WAKEUP	0	**Wakeup** Asserted when the USB is configured for host operation and the NS9750 moves from the suspend state to the resume state.
D05:02	N/A	Reserved	N/A	**N/A**

Table 420: Global Interrupt Status register

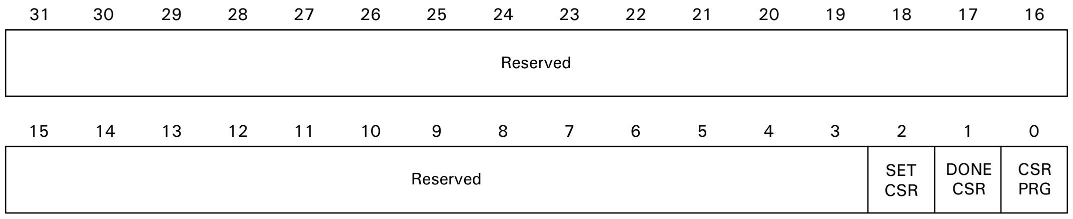
Bits	Access	Mnemonic	Reset	Description
D01	RW1TC	OHCI_IRQ	0	**OHCI_IRQ** Asserted when the USB is configured for host operation and the OHCI asserts an interrupt.
D00	N/A	Reserved	N/A	N/A

Table 420: Global Interrupt Status register

Device IP Programming Control/Status register

Address: 9010 0014

The Device IP Programming Control/Status register contains the USB device CSR dynamic programming control and status information. The register allows you to reconfigure the USB device when a SET_CFG or SET_INTF packet is received from the USB host.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:03	R	Not used	0x00000000	This field is always read back as 0x00000000.
D02	R	SETCSR	0	**CSR programming start** Indicates when software can safely start programming the registers. Must be set to 1 before programming can begin.
D01	R/W	DONECSR	0	**CSR programming done** Indicates to the device IP that software has finished programming the CSRs. A value of 1 indicates that software is finished.

Table 421: Device IP Programming Control/Status register

Bits	Access	Mnemonic	Reset	Description
D00	R/W	CSRPRG	0	**CSR dynamic programming support** Enables dynamic programming support in the USB device IP. Program this field on powerup, then leave it unchanged. Write a 1 to enable this feature.

Table 421: Device IP Programming Control/Status register

USB host block registers

The USB Host Block registers are for the host controller defined in the Open HCI specification for USB. All references to *HC* refer to the USB host block in the NS9750.

Reserved bits

The Host Controller Driver (HCD) should always preserve the value(s) of the reserved field. When a R/W register is modified, the HCD should first read the register, modify the appropriate bits, then write the register with the reserved bits still containing the read value. As an alternative, the HCD can maintain an in-memory copy of previously written values that can be modified and then written to the Host Controller (HC) register. When a write to set/clear a register is written, bits written to reserved fields should be 0.

USB host block register address map

Table 422 provides the addresses of the USB host block registers.

Address	Register
9010 1000	HcRevision register
9010 1004	HcControl register

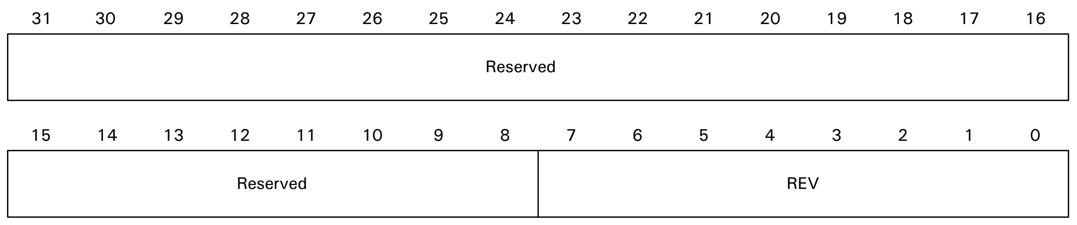
Table 422: USB Host Block registers address map

Address	Register
9010 1008	HcCommandStatus register
9010 100C	HcInterrupt Status register
9010 1010	HcInterruptEnable register
9010 1014	HcInterruptDisable register
9010 1018	HcHCCA (Host Controller Communications Area) register
9010 101C	HcPeriodCurrentED (Endpoint Descriptor) register
9010 1020	HcControlHeadED register
9010 1024	HcControlCurrentED register
9010 1028	HcBulkHeadED register
9010 102C	HcBulkCurrentED register
9010 1030	HcDoneHead register
9010 1034	HcFmInterval (Fm=Frame)
9010 1038	HcFmRemaining register
9010 103C	HcFmNumber register
9010 1040	HcPeriodicStart register
9010 1044	HcLSThreshold register
9010 1048	HcRhDescriptorA register (Rh=Root hub)
9010 104C	HcRhDescriptorB register
9010 1050	HcRhStatus register
9010 1054	HcRhPortStatus[1] register

Table 422: USB Host Block registers address map

HCRevision register

Address: 9010 1000



Register bit assignment

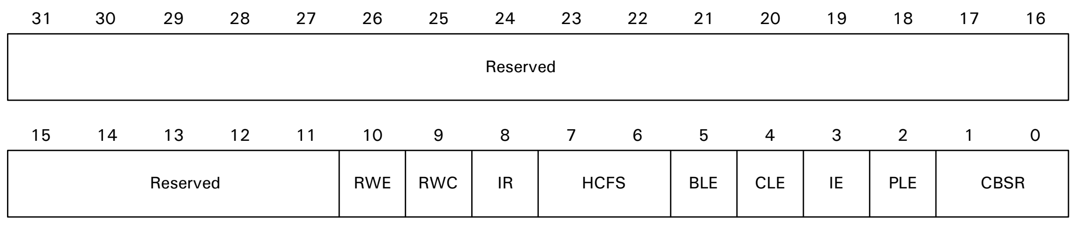
Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A
D07:00	R	REV	10h	Version of the OHCI specification being used.

Table 423: HcRevision register

HcControl register

Address: 9010 1004

The HcControl register defines the operating modes for the host controller. Most of the fields in this register are modified only by the host controller driver, with the exception of the HostControllerFunctionalState and RemoteWakeupConnected fields.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:11	N/A	Reserved	N/A	N/A

Table 424: HcControl register

Bits	Access	Mnemonic	Reset	Description
D10	R/W	RWE	0b	**RemoteWakeupEnable** Enables or disables the remote wakeup feature when upstream resume signaling is found. When this bit is set and the ResumeDetected bit in the HcInterruptStatus register is set, a remote wakeup is signaled to the host system. Setting this bit has no impact on the generation of hardware interrupts.
D09	R/W	RWC	0b	**RemoteWakeupConnected** Indicates whether the host controller supports remote wakeup signaling. If remote wakeup is supported and used by the system, it is the system firmware's responsibility to set this bit during POST. The host controller clears the bit on a hardware reset, but does not alter the bit on a software reset.
D08	R/W	IR	0b	**InterruptRouting** Determines the routing of interrupts generated by events registered in the HcInterruptStatus register. ■ If clear, all interrupts are routed to the normal host bus interrupt mechanism. ■ If set, interrupts are routed to the system management interrupt. The host controller driver clears this bit on a hardware reset, but does not alter the bit on a software reset. The host controller driver uses this bit as a tag to indicate the ownership of the host controller.

Table 424: HcControl register

Bits	Access	Mnemonic	Reset	Description
D07:06	R/W	HCFS	00b	**HostControllerFunctionalState** (b = binary) 00b USBRESET (initial state) 01b USBRESUME 10b USBOPERATIONAL 11b USBSUSPEND A transition to USBOPERATIONAL from another state causes SOF generation to begin 1 ms later. The host controller driver can determine whether the host controller has begun sending SOFs by reading the StartofFrame field of the HcInterruptStatus register. The host controller can change this field only when in the USBSUSPEND state. The host controller can move from the USBSUSPEND state to the USBRESUME state after detecting the resume signal from a downstream port. The host controller enters the USBSUSPEND state after a software reset, whereas it enters USBRESET after a hardware reset. A hardware reset also resets the root hub, and asserts subsequent reset signaling to downstream ports.
D05	R/W	BLE	0b	**BulkListEnable** Set to enable processing of the bulk list in the next frame. If cleared by the host controller driver, the bulk list is not processed after the next SOF. The host controller checks this bit whenever it determines to process this list. When disabled, the host controller driver can modify the list. If HcBulkCurrentED is pointing to an ED to be removed, the host controller driver must advance the pointer by updating HcBulkCurrentED before re-enabling processing of the list.
D04	R/W	CLE	0b	**ControlListEnable** Set to enable processing the control list in the next frame. If cleared by the host controller driver, the control list is not processed after the next SOF. The host controller must check this bit whenever it wants to process the list. When disabled, the host controller driver can modify the list. If HcControlCurrentED is pointing to an ED to be removed, the host controller driver must advance the pointer by updating HcControlCurrentED before re-enabling processing of the list.

Table 424: HcControl register

Bits	Access	Mnemonic	Reset	Description
D03	R/W	IE	0b	**IsochronousEnable** Enables/disables processing of the isochronous EDs. While processing the periodic list in a frame, the host controller checks the status of this bit when it finds an isochronous ED (F=1). ■ If set (enabled), the host controller continues processing the EDs. ■ If cleared (disabled), the host controller stops processing of the periodic list (which now contains only isochronous EDs) and begins processing the bulk/control lists. ■ Setting this bit is guaranteed to take effect in the next frame — not the current frame.
D02	R/W	PLE	0b	**PeriodicListEnable** Set to enable the processing of the periodic list in the next frame. If cleared by the host controller driver, the periodic list is not processed after the next SOF. The host controller must check this bit before it starts processing the list.
D01:00	R/W	CBSR	00b	**ControlBulkServiceRatio** Specifies the service ratio between control and bulk endpoint descriptors (EDs). Before processing any of the nonperiodic lists, the host controller must compare the ratio specified with its internal count on how many nonempty control EDs have been processed, to determine whether to continue serving another control ED or switch to a bulk ED. The internal count is retained when crossing the frame boundary. In the case of a reset, the host controller driver is responsible for restoring this value. **CBSR** — **# of control EDs over bulk EDs served** 0 — 1 : 1 1 — 2 : 1 2 — 3 : 1 3 — 4 : 1

Table 424: HcControl register

HcCommandStatus register

Address: 9010 1008

The host controller uses the HcCommandStatus register to receive commands issued by the host controller driver, as well as to reflect the current status of the host controller. The HcCommandStatus register appears to the host controller driver as a *write to set* register. The host controller must ensure that bits written as 1 become set in the register while bits written as 0 remain unchanged in the register. The host controller driver can issue multiple distinct commands to the host controller without concern for corrupting previously-issued commands. The host controller driver has normal read access to all bits. The host controller driver has normal read access to all bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														SOC	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												OCR	BLF	CLF	HCR

Register bit description

Bits	Access	Mnemonic	Reset	Description
D31:18	N/A	Reserved	N/A	N/A
D17:16	R	SOC	00b	**SchedulingOverrunCount** Indicates the number of frames with which the host controller has found a scheduling overrun error. A scheduling overrun error occurs when the periodic list does not complete before EOF. When a scheduling overrun error is found, the host controller increments the counter and sets the Scheduling Overrun field in the HcInterruptStatus register (see "HcInterruptStatus register," beginning on page 733). This field initializes to 00b and wraps around at 11b. This field is incremented on each scheduling overrun error, even if the SchedulingOverrun field in the HcInterruptStatus register has already been set. The host controller driver uses this field to monitor any persistent scheduling problems.

Table 425: HcCommandStatus register

Bits	Access	Mnemonic	Reset	Description
D15:04	N/A	Reserved	N/A	N/A
D03	R/W	OCR	0b	**OwnershipChangeRequest** Set by an OS host controller to request a change of control for the host controller. When set, the host controller sets the OwnershipChange field in the HcInterruptStatus register (see "HcInterruptStatus register," beginning on page 733). After the change is made, the OCR bit is cleared and remains so until the next request from OS host controller.
D02	R/W	BLF	0b	**BulkListFilled** Indicates whether there are any TDs on the bulk list. This bit is set by the host controller driver whenever it adds a TD to an ED in the bulk list. When the host controller begins to process the head of the bulk list, it checks the BLF field. As long as the BLF field is 0, the host controller will not start processing the bulk list. If BLF is 1, the host controller starts processing the bulk list and sets the BLF field to 0. If the host controller finds a TD on the list, it sets BLF to 1, which causes bulk list processing to continue. If no TD is found on the bulk list, and if the host controller driver does not set the BLF field, the BLF value will still be 0 when the host controller completes processing the bulk list; the bulk list processing then stops.
D01	R/W	CLF	0b	**ControlListFilled** Indicates whether there are any TDs (task descriptors) on the control list. This bit is set by the host controller driver whenever it adds a TD to an ED in the control list. When the host controller begins to process the head of the control list, it checks the CLF field. As long as CLF is 0, the host controller will not start processing the control list. If CLF is set to 1, the host controller starts processing the control list and sets the CLF field to 0. If the host controller finds a TD on the list, it sets CLF to 1, which causes control list processing to continue. If no TD is found on the control list, and if the host controller does not set the CLF field, the CLF value will still be 0 when the host controller completes processing the control list; control list processing then stops.

Table 425: HcCommandStatus register

Bits	Access	Mnemonic	Reset	Description
D00	R/W	HCR	0b	**HostControllerReset** Set by the host controller driver to initiate a software reset of the host controller. Regardless of the functional state of the host controller, it moves to USBSUSPEND state. This bit is cleared by the host controller on completion of the reset operation.

Table 425: HcCommandStatus register

HcInterruptStatus register

Address: 9010 100C

The HcInterruptStatus register provides status on various events that cause hardware interrupts. When an event occurs, the NS9750 sets the corresponding bit in this register. When a bit is set, a hardware interrupt is generated if the interrupt is enabled in the HcInterruptEnable register (see "HcInterruptEnable register," beginning on page 735) and the MasterInterruptEnable bit (in the HcInterruptEnable register) is set. The host controller driver can clear specific bits in this register by writing a 1 to the bit positions to be cleared, but cannot set any of these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Rsvd	OC	Reserved													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RHSC	FNO	UE	RD	SF	WDH	SO

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	N/A	Reserved	N/A	N/A
D30	R/W	OC	0b	**OwnershipChange** Set by the host controller when the host controller driver sets the OwnershipChangeRequest field in the HcCommandStatus register (see "HcCommandStatus register," beginning on page 730). This event, when unmasked, always generates a system management interrupt (SMI) immediately. This bit is tied to 0b when the SMI pin is not implemented.
D29:07	N/A	Reserved	N/A	N/A
D06	R/W	RHSC	0b	**RootHubStatusChange** Set when the content of the HcRhStatus register (see "HcRhStatus register," beginning on page 756) or any HcRhPortStatus[NumberofDownstreamPort] register (see "HcRhPortStatus[1] register," beginning on page 759) has changed.
D05	R/W	FNO	0b	**FrameNumberOverflow** Set when the most significant bit (MSB), bit 15, of the HcFmNumber (see "HcFmNumber register," beginning on page 749) changes value, from 0 to 1 or 1 to 0, and after HccaFrameNumber has been updated.
D04	R/W	UE	0b	**UnrecoverableError** Set when the host controller finds a system error not related to USB. The host controller should not proceed with any processing nor signaling before the system error has been corrected. The host controller driver clears this bit after the host controller has been reset.
D03	R/W	RD	0b	**ResumeDetected** Set when the host controller finds that a device on the USB is asserting resume signaling. The transition from no resume signaling to resume signaling causes this bit to be set. This bit is not set when the host controller driver sets the USBRESUME state.

Table 426: HcInterruptStatus register

Bits	Access	Mnemonic	Reset	Description
D02	R/W	SF	0b	**StartofFrame** Set by the host controller at each start of a frame and after the update of HccaFrameNumber. The host controller generates a SOF token at the same time.
D01	R/W	WDH	0b	**WritebackDoneHead** Set immediately after the host controller has written HcDoneHead (see "HcDoneHead register," beginning on page 746) to HccaDoneHead. The host controller driver should clear this bit only after it has saved the content of HccaDoneHead.
D00	R/W	SO	0b	**SchedulingOverrun** Set when the USB schedule for the current frame overruns and after the update of HccaFrameNumber. A scheduling overrun also causes the SchedulingOverrunCount in the HcCommandStatus register (see "HcCommandStatus register," beginning on page 730) to be incremented.

Table 426: HcInterruptStatus register

HcInterruptEnable register

Address: 9010 1010

Each enable bit in the HcInterruptEnable register corresponds to an associated interrupt bit in the HcInterrupt Status register (see "HcInterruptStatus register," beginning on page 733). The HcInterruptEnable register controls which events generate a hardware interrupt. When a bit is set in the HcInterruptStatus register **and** the corresponding bit in the HcInterruptEnable register is set **and** the MasterInterruptEnable bit (D31) in this register), a hardware interrupt is requested on the host bus.

Writing a 1 to a bit in this register sets the corresponding bit; setting a bit to 0 leaves the corresponding bit unchanged. On a read, the current value of this register is returned.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MIE	OC	Reserved													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RHSC	FNO	UE	RD	SF	WDH	SO

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	MIE	0b	**Master interrupt enable** 0 Ignored by the host controller 1 Enables interrupt generation due to events specified in the other bits of this register.
D30	R/W	OC	0b	**Ownership change** 0 Ignore 1 Enable interrupt generation due to ownership change.
D29:07	N/A	Reserved	N/A	N/A
D06	R/W	RHSC	0b	**Root hub status change** 0 Ignore 1 Enable interrupt generation due to root hub status change.
D05	R/W	FNO	0b	**Frame number overflow** 0 Ignore 1 Enable interrupt generation due to frame number overflow.
D04	R/W	UE	0b	**Unrecoverable error** 0 Ignore 1 Enable interrupt generation due to unrecoverable error.
D03	R/W	RD	0b	**Resume detect** 0 Ignore 1 Enable interrupt generation due to resume detect.

Table 427: HcInterruptEnable register

Bits	Access	Mnemonic	Reset	Description
D02	R/W	SF	0b	**Start of frame** 0 Ignore 1 Enable interrupt generation due to start of frame.
D01	R/W	WDH	0b	**HcDoneHead writeback** 0 Ignore 1 Enable interrupt generation due to HcDoneHead writeback.
D00	R/W	SO	0b	**Scheduling overrun** 0 Ignore 1 Enable interrupt generation due to scheduling overrun.

Table 427: HcInterruptEnable register

HcInterruptDisable register

Address: 9010 1014

Each disable bit in the HcInterruptDisable register corresponds to an associated interrupt bit in the HcInterruptStatus register (see "HcInterruptStatus register," beginning on page 733). The HcInterruptDisable register works in conjunction with the HcInterruptEnable register (see "HcInterruptEnable register," beginning on page 735). Writing a 1 in the HcInterruptDisable register clears the corresponding bit in the HcInterruptEnable register; writing a 0 to a bit in the HcInterruptDisable register leaves the corresponding bit in the HcInterruptEnable register unchanged.

On a read, the current value of the HcInterruptEnable register is returned.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MIE	OC	Reserved													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RHSC	FNO	UE	RD	SF	WDH	SO

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	MIE	0b	**Master interrupt enable** 0 Ignored by the host controller. 1 Disables interrupt generation due to events specified in other bits in this register. This field is set after a hardware or software reset.
D30	R/W	OC	0b	**Ownership change** 0 Ignore 1 Disable interrupt generation due to ownership change.
D29:07	N/A	Reserved	N/A	N/A
D06	R/W	RHSC	0b	**Root hub status change** 0 Ignore 1 Disable interrupt generation due to root hub status change.
D05	R/W	FNO	0b	**Frame number overflow** 0 Ignore 1 Disable interrupt generation due to frame number overflow.
D04	R/W	UE	0b	**Unrecoverable error** 0 Ignore 1 Disable interrupt generation due to unrecoverable error.
D03	R/W	RD	0b	**Resume detect** 0 Ignore 1 Disable interrupt generation due to resume detect.
D02	R/W	SF	0b	**Start of frame** 0 Ignore 1 Disable interrupt generation due to start of frame.
D01	R/W	WDH	0b	**HcDoneHead writeback** 0 Ignore 1 Disable interrupt generation due to HcDoneHead writeback.

Table 428: HcInterruptDisable register

Bits	Access	Mnemonic	Reset	Description
D00	R/W	SO	0b	**Scheduling overrun** 0 Ignore 1 Disable interrupt generation due to scheduling overrun.

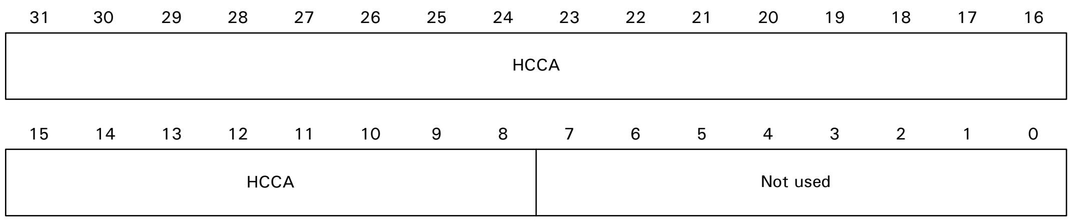
Table 428: HcInterruptDisable register

HcHCCA register

Address: 9010 1018

The HcHCCA register contains the physical address of the host controller communication area (HCCA), which is a RAM area with a defined format. The host controller driver determines the alignment restrictions by writing all 1s to HcHCCA and reading the content of HcHCCA. The alignment is evaluated by examining the number of zeros in the lower order bits. The minimum alignment is 256 bytes; bits 0 through 7, then, must always return 0 when read.

The host controller communication area holds the control structures and the interrupt table that are accessed by both the host controller and the host controller driver.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	R/W	HCCA	0h	Base address of the host controller communication area.
D07:00	R/W	Not used	0	Must be written to 0.

Table 429: HcHCCA register

HcPeriodCurrentED register

Address: 9010 101C

The HcPeriodCurrentED register contains the physical address of the current isochronous or interrupt endpoint descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCED															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCED												Not used			

Register bit assignment

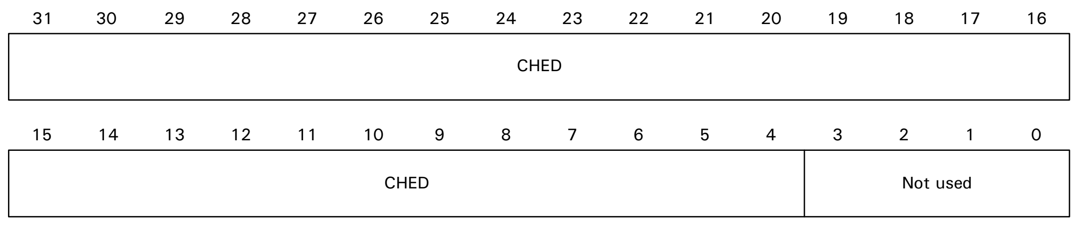
Bits	Access	Mnemonic	Reset	Description
D31:04	R	PCED	0h	**PeriodCurrentED** Used by the host controller to point to the head of one of the periodic lists that will be processed in the current frame. The content of this register is updated by the host controller after a periodic endpoint has been processed. The host controller driver can read the content to determine which endpoint currently is being processed at the time of the reading.
D03:00	N/A	Not used	0	Must be written to 0.

Table 430: HcPeriodCurrentED

HcControlHeadED register

Address: 9010 1020

The HcHeadControlED register contains the physical address of the first endpoint descriptor of the control list.



Register bit assignment

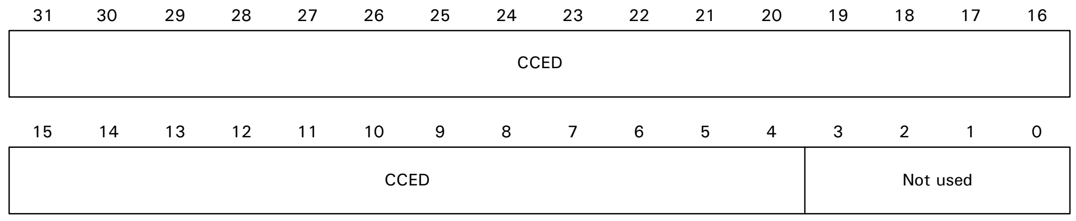
Bits	Access	Mnemonic	Reset	Description
D31:04	R/W	CHED	0h	**ControlHeadED** The host controller traverses the control list starting with the HcCOntrolHeadED pointer. The content is loaded from the host controller communication area during the host controller initialization.
D03:00	N/A	Not used	0	Must be written to 0.

Table 431: HcControlHeadED register

HcControlCurrentED register

Address: 9010 1024

The HcControlCurrentED register contains the physical address of the control list's current endpoint descriptor.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	R/W	CCED	0h	**ControlCurrentED** This pointer is advanced to the next endpoint descriptor after serving the present one. The host controller continues processing the list from where it left off in the last frame. When it reaches the end of the control list, the host controller checks the ControlListFilled field (see "HcCommandStatus register," beginning on page 730). If the ControlListFilled field is set, the host controller copies the content of the HcControlHeadED register to this register and clears the bit. If the ControlListFilled field is not set, the host controller does nothing. The host controller driver is allowed to modify this register only when the ControlListEnable field (see "HcControl register," beginning on page 727) is cleared. When the ControlListEnable field is set, the host controller driver only reads the instantaneous value of this register. Initially, this value is set to zero to indicate the end of the control list.
D03:00	N/A	Not used	0	Must be written to 0.

Table 432: HcControlCurrentED register

HcBulkHeadED register

Address: 9010 1028

The HcBulkHeadED register contains the physical address of the first endpoint descriptor of the bulk list.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BHED															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BHED												Not used			

Register bit assignment

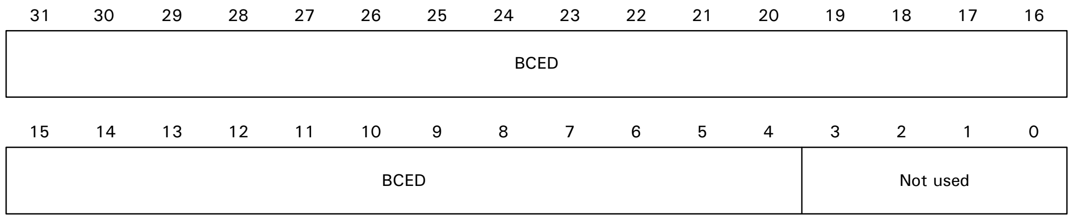
Bits	Access	Mnemonic	Reset	Description
D31:04	R/W	BHED	0h	BulkHeadED The host controller traverses the bulk list starting with the HcBulkHeadED pointer. The content is loaded from the host controller communication area during the host controller initialization.
D03:00	N/A	Not used	0	Must be written to 0.

Table 433: HcBulkHeadED register

HcBulkCurrentED register

Address: 9010 102C

The HcBulkCurrentED register contains the physical address of the bulk list's current endpoint. As the bulk list will be served in round-robin fashion, the endpoints will be ordered according to their insertion in the list.



Register bit assignment

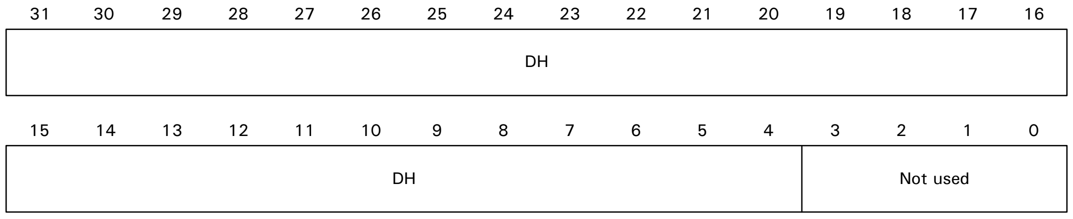
Bits	Access	Mnemonic	Reset	Description
D31:04	R/W	BCED	0h	**BulkCurrentED** BulkCurrentED is advanced to the next endpoint descriptor after the host controller has served the present endpoint descriptor. The host controller continues processing the list from where it left off in the last frame. When it reaches the end of the bulk list, the host controller checks the ControlListFilled field (see "HcControl register," beginning on page 727). If ControlListFIlled is set, the host controller copies the content of HcBulkHeadED to HcBulkCurrentED and clears the bit. If ControlListFilled is not set, the host controller does nothing. The host controller driver is allowed to modify this register only when the BulkListEnable field (see "HcControl register," beginning on page 727) is cleared. When BulkListEnable is set, the host control driver only reads the instantaneous value of this register. This value initially is set to zero to indicate the end of the bulk list.
D03:00	N/A	Not used	0	Must be written to 0.

Table 434: HcBulkCurrentED register

HcDoneHead register

Address: 9010 1030

The HcDoneHead register contains the physical address of the last completed transfer descriptor that was added to the Done queue. In normal operation, the host controller driver should not need to read this register as its content is written periodically to the host controller communication area.



Register bit assignment

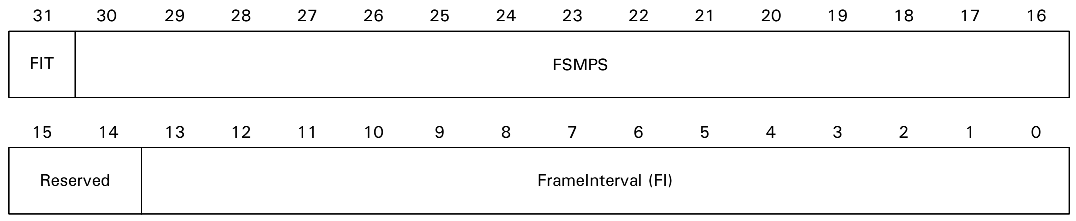
Bits	Access	Mnemonic	Reset	Description
D31:04	R	DH	0h	**DoneHead** When a TD is completed, the host controller writes the content of HcDoneHead to the NextTD field of the TD. The host controller then overwrites the content of HcDoneHead with the address of this TD. This value is written to zero whenever the host controller writes the content of this register to the host controller communications area. It also sets the WritebackDoneHead of the HcInterruptStatus register (see "HcInterruptStatus register," beginning on page 733).
D03:00	N/A	Not used	0	Must be written to 0.

Table 435: HcDoneHead register

HcFmInterval register

Address: 9010 1034

The HcFmInterval register contains the 14-bit value that indicates the bit time interval in a frame (that is, between two consecutive SOFs), and a 15-bit value indicating the full speed maximum packet size that the host controller can transmit or receive without causing a scheduling overrun. The host controller driver can perform minor adjustment on the FrameInterval by writing a new value over the present one at each SOF. This provides the programmability necessary for the host controller to synchronize with an external clocking resource and to adjust any unknown local clock offset.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	FIT	0b	**FrameIntervalToggle** The host controller driver toggles this bit whenever it loads a new value to FrameInterval.
D30:16	R/W	FSMPS	0	**FSLargestDataPacket** Specifies a value that is loaded into the largest data packet counter at the beginning of each frame. The counter value represents the largest amount of data, in bits, that can be sent to received by the host controller in a single transaction, at any given timer, without causing scheduling overrun. The field value is calculated by the host controller driver.
D15:14	N/A	Reserved	N/A	N/A

Table 436: HcFmInterval register

Bits	Access	Mnemonic	Reset	Description
D13:00	R/W	FI	2EDFh	**FrameInterval** Specifies the interval between two consecutive SOFs in bit times. The nominal value is 11,999. The host controller driver should store the current value of this field before resetting the host controller using the HostControllerReset field in the HcCommandStatus register (see "HcCommandStatus register," beginning on page 730), as this causes the host controller to reset this field to its nominal value. The host controller driver optionally can restore the stored value when the reset sequence completes.

Table 436: HcFmInterval register

HcFmRemaining register

Address: 9010 1038

The HcFmRemaining register is a 14-bit down counter showing the bit time remaining in the current frame.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FRT	Reserved														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		FrameRemainingCounter (FR)													

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R	FRT	0b	**FrameRemainingToggle** Loaded from the FrameIntervalToggle field of the HcFmInterval register (see "HcFmInterval register" on page 747) when FrameRemaining (D13:00 in this register) reaches 0. This bit is used by HCD for synchronization between FrameInterval and FrameRemaining.
D30:14	N/A	Reserved	N/A	N/A
D13:00	R	FR	0h	**FrameRemaining counter** Decremented at each bit time. When the counter reaches zero, it is reset by loading the FrameInterval value specified in the HcFrameInterval register (see "HcFmInterval register" on page 747) at the next bit time boundary.

Table 437: HcFmRemaining register

HcFmNumber register

Address: 9010 103C

The HcFmNumber register is a 16-bit counter that provides a timing reference among events happening in the host controller driver. The host controller driver can use the 16-bit value specified in this register and generate a 32-bit frame number without requiring frequent access to the register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FrameNumber (FN															

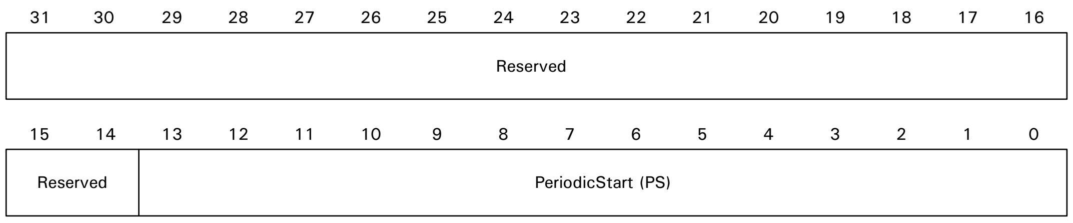
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A
D15:00	R	FN	0h	**FrameNumber** Incremented when the HcFmRemaining register (see "HcFmRemaining register" on page 748) is reloaded. The frame number will be rolled over to 0h after ffffh. When entering the USB operational state, FrameNumber is incremented automatically. The content is written to HCCA after the host controller has incremented FrameNumber at each frame boundary and sent a SOF but before HC reads the first endpoint descriptor in that frame. After writing to HCCa, the host controller sets the StartofFrame field in the HcInterruptStatus register (see "HcInterruptStatus register" on page 733).

Table 438: HcFmNumber register

HcPeriodicStart register

Address: 9010 1040



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:14	N/A	Reserved	N/A	N/A

Table 439: HcPeriodicStart register

Bits	Access	Mnemonic	Reset	Description
D13:00	R/W	PS	0h	**PeriodicStart** Determines when is the earliest time the host controller should start processing the periodic list. After a hardware reset, the PS field is cleared. The field is then set by the host controller driver during the host controller initialization. The value is calculated as approximately 10% off from the FrameInterval. A typical value is 3E67h. When FrameRemaining reaches the value specified, processing the periodic lists will have priority over control/bulk processing. The host controller starts processing the interrupt list after completing the current control or bulk transaction in progress.

Table 439: HcPeriodicStart register

HcLsThreshold register

Address: 9010 1044

The HcLSThreshold register contains a value used by the host controller to determine whether to commit to the transfer of a maximum-of-8-byte LS packet before EOF. Neither the host controller driver not the host controller is allowed to change this value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				LSThreshold(LST)											

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:12	N/A	Reserved	N/A	N/A
D11:00	R/W	LST	0628h	**LSThreshold** Contains a value that is compared to the FrameRemaining field before initiating a low speed (LS) transaction. The transaction is started only if FrameRemaining is greater than or equal to LSThreshold. The value is calculated by the host controller driver, with transmission and setup overhead considerations.

Table 440: HcLsThreshold register

Root hub partition registers

The remaining USB host block registers are dedicated to the USB root hub, which is an integral part of the host controller although it is a functionally separate entity. The host controller driver emulates USBD accesses to the root hub through a register interface. The host controller driver maintains many USB-defined hub features that are not required to be supported in hardware; for example, the hub's device configuration, interface, and endpoint descriptors are maintained only in the host controller driver as well as some status fields of the class descriptor. The host controller driver also maintains and decodes the root hub's device address as well as performs other operations that are better suited to software than hardware.

The root hub register interface maintains similarity of bit organization and operation to typical hubs that are found in the system. The four registers defined in this section are each read and written as 32 bits. These registers are written only during initialization to correspond with system implementation.

- The HcRhDescriptor A and HcRhDescriptorB registers should be implemented to be writeable, regardless of the host controller USB state.
- The HcRhStatus register and HcRhPortStatus register must be writeable during the USB operational state.

Note: *IS* denotes an *implementation-specific* reset value for the related field.

HcRhDescriptorA register

Address: 9010 1048

The HcRhDescriptorA register is the first of two registers describing the characteristics of the *root hub*. The root hub is the logical hub built into a USB host. Reset values are implementation-specific.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POTPGT								Reserved							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NOCP	OCPM	DT	PSM	NPS	NDP							

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:24	R/W	POTPGT	IS	**PowerOnToPowerGoodTime** Specifies the length of time the host controller driver has to wait before accessing a powered-on port of the root hub. The unit of time is 2 ms. The duration is calculated as POTPGT*2 ms.
D23:13	N/A	Reserved	N/A	N/A
D12	R/W	NOCP	IS	**NoOverCurrentProtection** Describes how the overcurrent status for the root hub ports is reported. When the NOCP bit is cleared, the OverCurrentProtectionMode field (D11 in this register) specifies global or per-port reporting: 0 Overcurrent status is reported collectively for all downstream ports. 1 No overcurrent protection supported.

Table 441: HcRhDescriptorA register

Bits	Access	Mnemonic	Reset	Description
D11	R/W	OCPM	IS	**OverCurrentProtectionMode** Describes how the overcurrent status for the root hub ports is reported. At reset, this field should reflect the same mode as PowerSwitchingMode (D08 in this register). The OCPM field is valid only if the NoOverCurrentProtection field (D12 in this register) is cleared. 0 Overcurrent status is reported collectively for all downstream ports. 1 Overcurrent status is reported on a per-port basis.
D10	R	DT	0b	**Device type** Specifies that the root hub is not a compound device. The root hub is not allowed to be a compound device. This field should always read 0.
D09	R/W	PSM	IS	**PowerSwitchingMode** Specifies how the power switching of the root hubs is controlled. This field is valid only if the NoPowerSwitching field (D08 in this register) is cleared. 0 All ports are powered at the same time. 1 Each port is powered individually. This mode allows port power to be controlled by either the global switch or per-port switching. If the PortPowerControlMask bit (see "HcRhDescriptorB register" on page 755) is set, the port responds only to port power commands (Set/ClearPortPower). If the port mask is cleared, the port is controlled only by the global power switch (Set/ClearGlobalPower).
D08	R/W	NPS	IS	**NoPowerSwitching** Specifies whether power switching is supported or ports are always powered. When this bit is cleared, the PowerSwitchingMode (D09 in this register) specifies global or per-port switching. 0 Ports are power switched. 1 Ports are always powered on when the host controller is powered on.
D07:00	R	NDP	IS	**NumberDownstreamPorts** Specifies the number of downstream ports supported by the root hub. The minimum number of ports is 1; the maximum number of ports supported is 15.

Table 441: HcRhDescriptorA register

HcRhDescriptorB register

Address: 9010 104C

The HcRhDescriptorB register is the second of two registers describing the characteristics of the root hub. These fields are written during initialization to correspond with the system implementation. Reset values are implementation-specific.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PortPowerControlMask (PPCM)															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DeviceRemovable (DR)															

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	R/W	PPCM	IS	**PortPowerControlMask** Each bit indicates whether a port is affected by a global control command when PowerSwitchingMode (see "HcRhDescriptorA register" on page 753) is set. When set, the port's power state is affected only by per-port power control (Set/ClearPortPower). When cleared, the port is controlled by the global power switch (Set/ClearGlobalPower). If the device is configured to global switching mode (PowerSwitchingMode=0), this field is not valid. Bit 0: Reserved Bit 1: Ganged-power mask on port #1 Bit 2: Ganged-power mask on port #2 ... Bit 15: Ganged-power mask on port #15

Table 442: HcRhDesdcriptorB register

Bits	Access	Mnemonic	Reset	Description
D15:00	R/W	DR	IS	**DeviceRemovable** Each bit is dedicated to a root hub port. ■ When cleared, the attached device is removable. ■ When set, the attached device is not removable. Bit 0: Reserved Bit 1: Device attached to port #1 Bit 2: Device attached to port #2 ... Bit 15: device attached to port #15

Table 442: HcRhDesdcriptorB register

HcRhStatus register

Address: 9010 1050

The HcRhStatus register has two parts:

- The lower word of a Dword represents the *hub status* field.
- The upper word of the Dword represents the *hub status change* field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRWE	Not used													CCIC	LPSC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRWE	Not used													OCI	LPS

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	W	CRWE	N/A	**ClearRemoteWakeupEnable** 0 Has no effect 1 Clears DeviceRemoveWakeupEnable
D30:18	N/A	Not used	N/A	Always write to 0.

Table 443: HcRhStatus register

Bits	Access	Mnemonic	Reset	Description
D17	R/W	CCIC	0b	**OverCurrentIndicatorChange** Set by hardware when a change has occurred to the OCI field (bit 01 in this register). The host controller driver clears this bit by writing 1. Writing 0 to this bit has no effect.
D16	R/W	LPSC	0b	**LocalPowerStatusChange (LPSC)** Not supported; always read as 0. **SetGlobalPower (write)** In global power mode (PowerSwitchingMode=0), this bit is written to 1 to turn on power to all ports (clear PowerPortStatus). ■ In per-port mode, this bit sets PortPowerStatus only on ports whose PortPowerControlMask bit is not set. ■ Writing 0 to this bit has no effect
D15	R/W	DRWE	0b	**DeviceRemoteWakeupEnable (read)** Enables a ConnectStatusChange bit as a resume event, causing a USBSUSPEND or USBRESUME state transition and setting the ResumeDetected interrupt. 0 ConnectStatusChange is not a remote wakeup event 1 ConnectStatusChange is a remote wakeup event. **SetRemoteWakeupEnable (write)** 0 Has no effect 1 Sets DeviceRemoteWakeupEnable
D14:02	N/A	Not used	N/A	Always write to 0.
D01	R	OCI	0b	**OverCurrentIndicator** Reports overcurrent conditions when global reporting is implemented. ■ When set, an overcurrent condition exists. ■ When cleared, all power operations are normal. If per-port overcurrent is implemented, this bit is always 0.

Table 443: HcRhStatus register

Bits	Access	Mnemonic	Reset	Description
D00	R/W	LPS	0b	**LocalPowerStatus (read)** Not supported; always read as 0. **ClearGlobalPower (write)** In global power mode (PowerSwitchingMode=0), this bit is always written to 1 to turn off power to all ports (clear PowerPortStatus). In per-port power mode, this bit clears PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing 0 to this bit has no effect.

Table 443: HcRhStatus register

HcRhPortStatus[1] register

Address: 9010 1054

The HcRhPortStatus register controls and reports port events on a per-port basis. The lower word reflects port status; the upper word reflects the status change bits. If a transaction (token through handshake) is in progress when a write to change port status occurs, the resulting port status change must be postponed until the transaction completes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											PRSC	OCIC	PSSC	PESC	CSC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used						LSDA	PPS	Not used			PRS	POCI	PSS	PES	CCS

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:21	N/A	Not used	N/A	Always write to 0.
D20	R/W	PRSC	0b	**PortResetStatusChange** 0 Port reset is not complete 1 Port reset is complete Set at the end of the 10-ms port reset signal. The host controller driver writes a 1 to clear this bit. Writing 0 has no effect.
D19	R/W	OCIC	0b	**PortOverCurrentIndicatorChange** 0 No change in PortOverCurrentIndicator 1 PortOverCurrentIndicator has changed Valid only if overcurrent conditions are reported on a per-port basis. This bit is set when root hub changes the PortOverCurrentIndicator bit. The host controller driver writes a 1 to clear this bit. Writing 0 has no effect.

Table 444: HcRhPortStatus[1] register

Bits	Access	Mnemonic	Reset	Description
D18	R/W	PSSC	0b	**PortSuspendStatusChange** 0 Resume is not completed 1 Resume completed Set when the full resume sequence has been completed. This sequence includes the 20-s resume pulse, LS EOP, and 3-ms resynchronization delay. The host controller driver writes a 1 to clear this bit. Writing 0 has no effect. This bit also is cleared when ResetStatusChange is set.
D17	R/W	PESC	0b	**PortEnableStatusChange** 0 No change in PortEnableStatus 1 Change in PortEnableStatus Set when hardware events cause the PortEnableStatus bit to be cleared. Changes from host controller driver writes do not set this bit. The host controller driver writes a 1 to clear this bit. Writing 0 has no effect.
D16	R/W	CSC	0b	**ConnectStatusChange** 0 No change in CurrentConnectStatus 1 Change in CurrentConnectStatus Set when a connect or disconnect event occurs. The host controller driver writes a 1 to clear this bit. Writing 0 has no effect. If CurrentConnectStatus is cleared when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, this bit is set to force the driver to re-evaluate the connection status, as these writes should not occur if the port is disconnected. **Note:** If the DeviceRemovable[NDP] bit is set, the CSC bit is set only after a root hub reset, to tell the system that the device is attached.
D15:10	N/A	Not used	N/A	Always write to 0.

Table 444: HcRhPortStatus[1] register

Bits	Access	Mnemonic	Reset	Description
D09	R/W	LSDA	Xb	**LowSpeedDeviceAttached (read)** 0 Full speed device attached 1 Low speed device attached Indicates the speed of the device attached to this port. When set, the low speed device is attached to this port. When clear, a full speed device is attached to this port. This field is valid only when CurrentConnectStatus is set. **ClearPortPower (write)** The host controller driver clears the PortPowerStatus bit by writing a 1 to this bit. Writing 0 has no effect.
D08	R/W	PPS	0b	**PortPowerStatus (read)** 0 Port power is off 1 Port power is on Reflects the port's power status, regardless of the type of power switching implemented. This bit is cleared is an overcurrent condition is found. The host controller driver sets this bit by writing SetPortPower or SetGlobalPower. The host controller driver clears this bit by writing ClearPortPower or ClearGlobalPower. PowerSwitchingMode and PortPowerControlMask determine which switches are enabled. ■ In global switching mode, (PowerSwitchingMode=0), only Set/ClearGlobalPower controls this bit. ■ In per-port power switching (PowerSwitchingMode=1), if the PortPowerControlMask bit is set, only Set/ClearPortPower commands are enabled. If the mask is not set, only Set/ClearGlobalPower commands are enabled. When port power is disabled, CurrentConnectStatus, PortEnableStatus, PortSuspendStatus, and PortResetStatus should be reset.
D08 *(cont)*	R/W	PPS	0b	**SetPortPower (write)** The host controller driver writes a 1 to set the PortPowerStatus bit. Writing a 0 has no effect. **Note:** This bit always reads 1b if power switching is not supported.

Table 444: HcRhPortStatus[1] register

Bits	Access	Mnemonic	Reset	Description
D07:05	N/A	Not used	N/A	Always write to 0.
D04	R/W	PRS	0b	**PortResetStatus (read)** 0 Port reset signal is not active 1 Port reset signal is active When this bit is set by a write to SetPortReset, port reset signalling is asserted. When reset is completed, this bit is cleared when PortResetStatusChange is set. This bit cannot be set if CurrentConnectStatus is cleared. **SetPortReset (write)** The HCD sets the port reset signalling by writing a 1 to this bit. Writing 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortResetStatus; it sets ConnectStatusChange, which tells the driver that it tried to reset a disconnected port.
D03	R/W	POCI	0b	**PortOverCurrentIndicator (read)** 0 No overcurrent condition 1 Overcurrent condition found Valid only when the root hub is configured such that overcurrent conditions are reported on a per-port basis. If per-port overcurrent reporting is not supported, this bit is set to 0. If cleared, all power operations are normal for this port. If set, an overcurrent condition exists on this port. This bit always reflects the overcurrent input signal. **ClearSuspendedStatus (write)** The host controller driver writes a 1 to initiate a resume. Writing 0 has no effect. A resume is initiated only if PortSuspendStatus is set.

Table 444: HcRhPortStatus[1] register

Bits	Access	Mnemonic	Reset	Description
D02	R/W	PSS	0b	**PortSuspendStatus (read)** 0 Port is not suspended 1 Port is suspended Indicates that the port is suspended or in the resume sequence. This bit is set by a SetSuspendState write and cleared when PortSuspendStatusChange is set at the end of the resume interval. This bit cannot be set if CurrentConnectStatus is cleared. This bit is cleared when PortResetStatusChange is set at the end of the port reset or when the host controller is placed in the USBRESUME state. If an upstream resume is in progress, it should propagate to the host controller. **SetPortSuspend (write)** The host controller driver sets PortEnableStatus by writing a 1. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortSuspendStatus; it sets ConnectStatusChange, which tells the driver that it tried to suspend a disconnected port.

Table 444: HcRhPortStatus[1] register

Bits	Access	Mnemonic	Reset	Description
D01	R/W	PES	0b	**PortEnableStatus (read)** 0 Port is disabled 1 Port is enabled Indicates whether the port is enabled or disabled. The NS9750 can clear this bit when an overcurrent condition, disconnect event, switched-off power, or operational bus error (such as babble) is found. This change also causes PortEnableStatusChange (bit 17 in this register) to be set. The host controller driver sets this bit by writing SetPortEnable and clears the bit by writing ClearPortEnable. This bit cannot be set when CurrentConnectStatus is cleared. This bit is also set, if not already done, at the completion of a port reset when ResetStatusChange is set or port suspend when SuspendStatusChange is set. **SetPortEnable (write)** Sets PortEnableStatus by writing a 1. Writing 0 has no effect. If CurrentCOnnectStatus is cleared, this write does not set PortEnableStatus; it sets ConnectStatusChange, which tells the driver that it tried to enable a disconnected port.
D00	R/W	CCS	0b	**CurrentConnectStatus (read)** Reflects the current state of the downstream port. 0 No device connected 1 Device connected **ClearPortEnable (write)** The host controller driver writes a 1 to this bit to clear the PortStatusEnable bit. Writing a 0 has no effect. The CurrentConnectStatus is not affected by any write. **Note:** This bit is always read as 1b when the attached device is non-removable (DeviceRemovable[NDP]).

Table 444: HcRhPortStatus[1] register

USB Device Block registers

Table 445 provides the addresses of the USB Device Block registers.

Address	Register
9010 2000	Device Descriptor/Setup Command register
9010 2004	Endpoint Descriptor #0
9010 2008	Endpoint Descriptor #1
9010 200C	Endpoint Descriptor #2
9010 2010	Endpoint Descriptor #3
9010 2014	Endpoint Descriptor #4
9010 2018	Endpoint Descriptor #5
9010 201C	Endpoint Descriptor #6
9010 2020	Endpoint Descriptor #7
9010 2024	Endpoint Descriptor #8
9010 2028	Endpoint Descriptor #9
9010 202C	Endpoint Descriptor #10
9010 2030	Endpoint Descriptor #11

Table 445: USB Device Block registers address map

Device Descriptor/Setup Command register

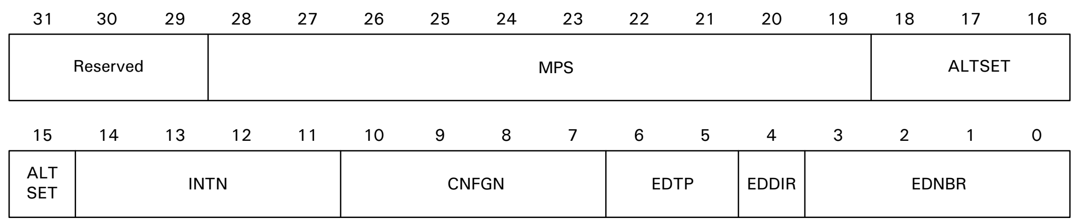
Address: 9010 2000

The Device Descriptor/Setup Command register is a legacy register. This register must be written to the value 0x0000_0100.

Endpoint Descriptor #0–#11 registers

Address: 9010 2004 / 2008 / 200C / 2010 / 2014 / 2018 / 201C / 2020 / 2024 / 2028 / 202C / 2030

The Endpoint Descriptor registers store the endpoint information. There are 12 registers, one for each endpoint descriptor. Each register contains the same information for the endpoint descriptor.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:29	N/A	Reserved	N/A	N/A
D28:19	R/W	MPS	10’h0	Maximum packet size.
D18:15	R/W	ALTSET	4’h0	Alternate setting to which this endpoint belongs.
D14:11	R/W	INTN	4’h0	Interface number to which this endpoint belongs.
D10:07	R/W	CNFGN	4’h0	Configuration number to which this endpoint belongs. **Note:** Value at 0x0 is not supported if dynamic programming is enabled (CSRPRG set to 1 in the Device IP Programming Control and Status register).
D06:05	R/W	EDTP	2’h0	**Endpoint type** 00 Control 01 Isochronous 10 Bulk 11 Interrupt
D04	R/W	EDDIR	1h’0	**Endpoint direction** 0 Out 1 In

Table 446: Endpoint Descriptor register (for endpoint descriptors 0–11)

Bits	Access	Mnemonic	Reset	Description
D03:00	R/W	EDNBR	4’h0	Endpoint number

Table 446: Endpoint Descriptor register (for endpoint descriptors 0–11)

USB Device Endpoint FIFO Control and Data registers

Table 447 provides the addresses for the endpoint registers found in the application logic that interfaces to the USB device block.

Address	Register
9010 3000	FIFO Interrupt Status 0
9010 3004	FIFO Interrupt Enable 0
9010 3010	FIFO Interrupt Status 1
9010 3014	FIFO Interrupt Enable 1
9010 3020	FIFO Interrupt Status 2
9010 3024	FIFO Interrupt Enable 2
9010 3030	FIFO Interrupt Status 3
9010 3034	FIFO Interrupt Enable 3
9010 3080	FIFO Packet Control #1
9010 3084	FIFO Packet Control #2
9010 3088	FIFO Packet Control #3
9010 308C	FIFO Packet Control #4
9010 3090	FIFO Packet Control #5
9010 3094	FIFO Packet Control #6
9010 3098	FIFO Packet Control #7
9010 309C	FIFO Packet Control #8
9010 30A0	FIFO Packet Control #9

Table 447: USB Device Endpoint FIFO Control registers address map

Address	Register
9010 30A4	FIFO Packet Control #10
9010 30A8	FIFO Packet Control #11
9010 30AC	FIFO Packet Control #12
9010 30B0	FIFO Packet Control #13
9010 3100	FIFO Status and Control #1
9010 3108	FIFO Status and Control #2
9010 3110	FIFO Status and Control #3
9010 3118	FIFO Status and Control #4
9010 3120	FIFO Status and Control #5
9010 3128	FIFO Status and Control #6
9010 3130	FIFO Status and Control #7
9010 3138	FIFO Status and Control #8
9010 3140	FIFO Status and Control #9
9010 3148	FIFO Status and Control #10
9010 3150	FIFO Status and Control #11
9010 3158	FIFO Status and Control #12
9010 3160	FIFO Status and Control #13

Table 447: USB Device Endpoint FIFO Control registers address map

Table 448 describes the *fixed* relationship from endpoint to interface FIFO to DMA channel for all registers described in the remainder of the chapter.

DMA channel	FIFO	EP number
1	1	0 (CTRL-Out)
2	2	0 (CTRL-In)
3	3	1
4	4	2
5	5	3

Table 448: FIFO to DMA channel to endpoint map

DMA channel	FIFO	EP number
6	6	4
7	7	5
8	8	6
9	9	7
10	10	8
11	11	9
12	12	10
13	13	11

Table 448: FIFO to DMA channel to endpoint map

FIFO Interrupt Status registers

The FIFO Interrupt Status registers contain interrupt status information for the device block FIFOs. All status bits are active high (1) and all interrupts are cleared by writing a 1 to the appropriate field.

Note: For diagnostic purposes, each of the interrupt status bits can be set by writing a 1 when the bits are at 0.

All FIFO status bits operate in a *catch and hold* mode, which means that once a status bit is set, it can be cleared only by writing a 1 to the corresponding bit position. If the status generating condition is present after writing a 1, the appropriate status bit is reasserted immediately.

Note: The NS9750 FIFO Interrupt Status registers pertain to DMA mode only; direct, or processor-controlled, mode is not supported at this time. The following control signals are considered “don’t care” signals, as they are valid only in direct mode: FULL, EMPTY, HALF, OFLOW, and UFLOW.

Device endpoint status

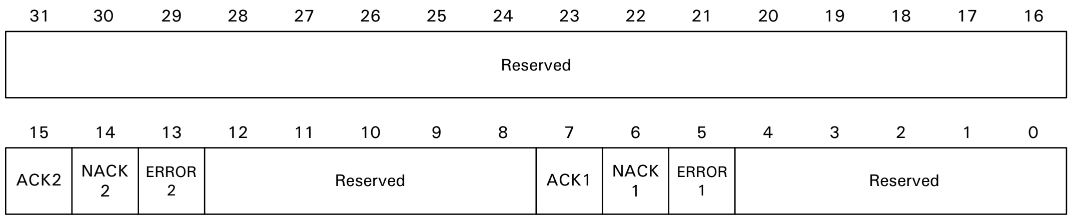
Table 449 defines the device endpoint status provided for each endpoint FIFO. Firmware uses this information to know which endpoints are active and whether there were any transmission errors.

Status	Direction	Description
ACK	In	Set when an ACK packet is received from the host in response to the previous data packet. For isochronous pipes, the ACK field is asserted automatically for each packet sent to the host.
NACK	In	Set when a NACK packet is sent to the host in response to the previous setup packet received from the host.
ERROR	In	Set when an ACK packet is not received from the host in response to the previous data packet sent to the host.
ACK	Out	Set when an ACK packet is sent to the host in response to the previous error-free packet received from the host. For isochronous pipes, this field is asserted automatically for each packet received from the host if the data is error-free.
NACK	Out	Set when a NACK packet is sent to the host in response to the previous setup packet received from the host.
ERROR	Out	Set when a packet is received from the host and a transmission error was found.

Table 449: USB device endpoint status

FIFO Interrupt Status 0 register

Address: 9010 3000



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Not used	0x0000	Always read as 0x0000.
D15	RW1TC	ACK2	0	Endpoint 0 (CTRL-In) acknowledge status. See Table 449, “USB device endpoint status,” on page 770.
D14	RW1TC	NACK2	0	Endpoint 0 (CTRL-In) negative acknowledge status. See Table 449, “USB device endpoint status,” on page 770.
D13	RW1TC	ERROR2	0	Endpoint 0 (CTRL-In) error status. See Table 449, “USB device endpoint status,” on page 770.
D12:08	N/A	Reserved	N/A	Not valid in DMA mode.
D07	RW1TC	ACK1	0	Endpoint 0 (CTRL-Out) acknowledge status. See Table 449, “USB device endpoint status,” on page 770
D06	RW1TC	NACK1	0	Endpoint 0 (CTRL-Out) negative acknowledge status. See Table 449, “USB device endpoint status,” on page 770.
D05	RW1TC	ERROR1	0	Endpoint 0 (CTRL-Out) error status. See Table 449, “USB device endpoint status,” on page 770.
D04:00	N/A	Reserved	N/A	Not valid in DMA mode.

Table 450: FIFO Interrupt Status 0 register

FIFO Interrupt Status 1 register

Address: 9010 3010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACK6	NACK 6	ERROR 6	Reserved					ACK5	NACK 5	ERROR 5	Reserved				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACK4	NACK 4	ERROR 4	Reserved					ACK3	NACK 3	ERROR 3	Reserved				

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	RW1TC	ACK6	0	Endpoint 4 acknowledge status. See Table 449, “USB device endpoint status,” on page 770.
D30	RW1TC	NACK6	0	Endpoint 4 negative acknowledge status. See Table 449, “USB device endpoint status,” on page 770.
D29	RW1TC	ERROR6	0	Endpoint 4 error status. See Table 449, “USB device endpoint status,” on page 770.
D28:24	N/A	Reserved	N/A	Not valid in DMA mode.
D23	RW1TC	ACK5	0	Endpoint 3 acknowledge status. See Table 449, “USB device endpoint status,” on page 770.
D22	RW1TC	NACK5	0	Endpoint 3 negative acknowledge status. See Table 449, “USB device endpoint status,” on page 770.
D21	RW1TC	ERROR5	0	Endpoint 3 error status. See Table 449, “USB device endpoint status,” on page 770.
D20:16	N/A	Reserved	N/A	Not valid in DMA mode.
D15	RW1TC	ACK4	0	Endpoint 2 acknowledge status. See Table 449, “USB device endpoint status,” on page 770.
D14	RW1TC	NACK4	0	Endpoint 2 negative acknowledge status. See Table 449, “USB device endpoint status,” on page 770.
D13	RW1TC	ERROR4	0	Endpoint 2 error status. See Table 449, “USB device endpoint status,” on page 770.
D12:08	N/A	Reserved	N/A	Not valid in DMA mode.
D07	RW1TC	ACK3	0	Endpoint 1 acknowledge status. See Table 449, “USB device endpoint status,” on page 770.

Table 451: FIFO Interrupt Status 1 register

Bits	Access	Mnemonic	Reset	Description
D06	RW1TC	NACK3	0	Endpoint 1 negative acknowledge status. See Table 449, “USB device endpoint status,” on page 770.
D05	RW1TC	ERROR3	0	Endpoint 1 error status. See Table 449, “USB device endpoint status,” on page 770.
D04:00	N/A	Reserved	N/A	Not valid in DMA mode.

Table 451: FIFO Interrupt Status 1 register

FIFO Interrupt Status 2 register

Address: 9010 3020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACK 10	NACK 10	ERROR 10	Reserved					ACK9	NACK 9	ERROR 9	Reserved				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACK8	NACK 8	ERROR 8	Reserved					ACK7	NACK 7	ERROR 7	Reserved				

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	RW1TC	ACK10	0	Endpoint 8 acknowledges status. See Table 449, “USB device endpoint status,” on page 770.
D30	RW1TC	NACK10	0	Endpoint 8 negative acknowledgement status. See Table 449, “USB device endpoint status,” on page 770.
D29	RW1TC	ERROR10	0	Endpoint 8 error status. See Table 449, “USB device endpoint status,” on page 770.
D28:24	N/A	Reserved	N/A	Not valid in DMA mode.
D23	RW1TC	ACK9	0	Endpoint 7 acknowledge status. See Table 449, “USB device endpoint status,” on page 770.
D22	RW1TC	NACK9	0	Endpoint 7 negative acknowledgement status. See Table 449, “USB device endpoint status,” on page 770.
D21	RW1TC	ERROR9	0	Endpoint 7 error status. See Table 449, “USB device endpoint status,” on page 770.

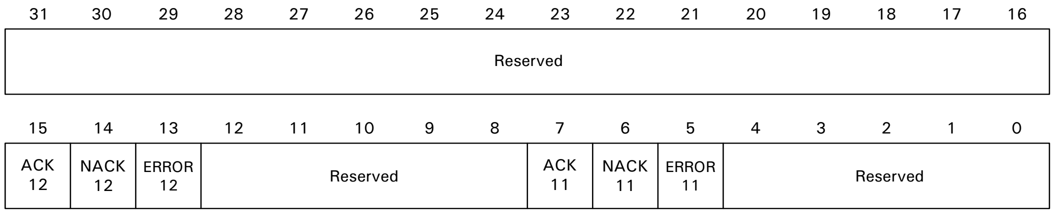
Table 452: FIFO Interrupt Status 2 register

Bits	Access	Mnemonic	Reset	Description
D20:16	N/A	Reserved	N/A	Not valid in DMA mode.
D15	RW1TC	ACK8	0	Endpoint 6 acknowledge status. See Table 449, “USB device endpoint status,” on page 770.
D14	RW1TC	NACK8	0	Endpoint 6 negative acknowledge status. See Table 449: "USB device endpoint status" on page 770.
D13	RW1TC	ERROR8	0	Endpoint 6 error status. See Table 449: "USB device endpoint status" on page 770.
D12:08	N/A	Reserved	N/A	Not valid in DMA mode.
D07	RW1TC	ACK7	0	Endpoint 5 acknowledge status. See Table 449: "USB device endpoint status" on page 770.
D06	RW1TC	NACK7	0	Endpoint 5 negative acknowledge status. See Table 449: "USB device endpoint status" on page 770.
D05	RW1TC	ERROR7	0	Endpoint 5 error status. See Table 449: "USB device endpoint status" on page 770.
D04:00	N/A	Reserved	N/A	Not valid in DMA mode.

Table 452: FIFO Interrupt Status 2 register

FIFO Interrupt Status 3 register

Address: 9010 3030



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A
D15	RW1TC	ACK12	0	Endpoint 10 acknowledge status. See Table 449: "USB device endpoint status" on page 770.
D14	RW1TC	NACK12	0	Endpoint 10 negative acknowledge status. See Table 449: "USB device endpoint status" on page 770.
D13	RW1TC	ERROR12	0	Endpoint 10 error status. See Table 449: "USB device endpoint status" on page 770.
D12:08	N/A	Reserved	N/A	Not valid in DMA mode.
D07	RW1TC	ACK11	0	Endpoint 9 acknowledge status. See Table 449: "USB device endpoint status" on page 770.
D06	RW1TC	NACK11	0	Endpoint 9 negative acknowledge status. See Table 449: "USB device endpoint status" on page 770.
D05	RW1TC	ERROR11	0	Endpoint 9 error status. See Table 449: "USB device endpoint status" on page 770.
D04:00	N/A	Reserved	N/A	Not valid in DMA mode.

Table 453: FIFO Interrupt Status 3 register

FIFO Interrupt Enable registers

The FIFO Interrupt Enable registers contain the interrupt enable information for the device block FIFOs. All interrupts are enabled by writing a 1 and are disabled by writing a 0. The endpoint to register field mapping is identical to the FIFO Interrupt Status registers.

FIFO Interrupt Enable 0 register

Address: 9010 3004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACK2	NACK 2	ERROR 2	Reserved					ACK1	NACK 1	ERROR 1	Reserved				

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Not used	0x0000	Always read as 0x0000.
D15	R/W	ACK2	0	Generate an interrupt when ACK2 in FIFO Interrupt Status 0 register is asserted.
D14	R/W	NACK2	0	Generate an interrupt when NACK2 in FIFO Interrupt Status 0 register is asserted.
D13	R/W	ERROR2	0	Generate an interrupt when ERROR2 in FIFO Interrupt Status 0 register is asserted.
D12:08	N/A	Reserved	N/A	Not valid in DMA mode.
D07	R/W	ACK1	0	Generate an interrupt when ACK1 in FIFO Interrupt Status 0 register is asserted.
D06	R/W	NACK1	0	Generate an interrupt when NACK1 in FIFO Interrupt Status 0 register is asserted.
D05	R/W	ERROR1	0	Generate an interrupt when ERROR1 in FIFO Interrupt Status 0 register is asserted.
D04:00	N/A	Reserved	N/A	Not valid in DMA mode.

Table 454: FIFO Interrupt Enable 0 register

FIFO Interrupt Enable 1 register

Address: 9010 3014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACK6	NACK 6	ERROR 6	Reserved					ACK5	NACK 5	ERROR 5	Reserved				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACK4	NACK 4	ERROR 4	Reserved					ACK3	NACK 3	ERROR 3	Reserved				

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	ACK6	0	Generate an interrupt when ACK6 in FIFO Interrupt Status 1 register is asserted.
D30	R/W	NACK6	0	Generate an interrupt when NACK6 in FIFO Interrupt Status 1 register is asserted.
D29	R/W	ERROR6	0	Generate an interrupt when ERROR6 in FIFO Interrupt Status 1 register is asserted.
D28:24	N/A	Reserved	N/A	Not valid in DMA mode.
D23	R/W	ACK5	0	Generate an interrupt when ACK5 in FIFO Interrupt Status 1 register is asserted.
D22	R/W	NACK5	0	Generate an interrupt when NACK5 in FIFO Interrupt Status 1 register is asserted.
D21	R/W	ERROR5	0	Generate an interrupt when ERROR5 in FIFO Interrupt Status 1 register is asserted.
D20:16	N/A	Reserved	N/A	Not valid in DMA mode.
D15	R/W	ACK4	0	Generate an interrupt when ACK4 in FIFO Interrupt Status 1 register is asserted.
D14	R/W	NACK4	0	Generate an interrupt when NACK4 in FIFO Interrupt Status 1 register is asserted.
D13	R/W	ERROR4	0	Generate an interrupt when ERROR4 in FIFO Interrupt Status 1 register is asserted.
D12:08	N/A	Reserved	N/A	Not valid in DMA mode.

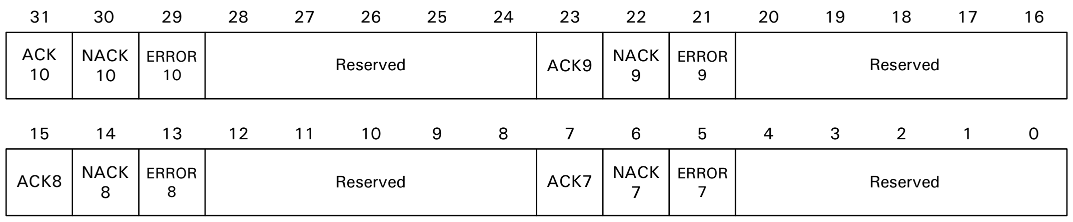
Table 455: FIFO Interrupt Enable 1 register

Bits	Access	Mnemonic	Reset	Description
D07	R/W	ACK3	0	Generate an interrupt when ACK3 in FIFO Interrupt Status 1 register is asserted.
D06	R/W	NACK3	0	Generate an interrupt when NACK3 in FIFO Interrupt Status 1 register is asserted.
D05	R/W	ERROR3	0	Generate an interrupts when ERROR3 in FIFO Interrupt Status 1 register is asserted.
D04:00	N/A	Reserved	N/A	Not valid in DMA mode.

Table 455: FIFO Interrupt Enable 1 register

FIFO Interrupt Enable 2 register

Address: 9010 3024



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	ACK10	0	Generate an interrupt when ACK10 in FIFO Interrupt Status 2 register is asserted,
D30	R/W	NACK10	0	Generate an interrupt when NACK10 in FIFO Interrupt Status 2 register is asserted.
D29	R/W	ERROR10	0	Generate an interrupt when ERROR10 in FIFO Interrupt Status 2 register is asserted.
D28:24	N/A	Reserved	N/A	Not valid in DMA mode.
D23	R/W	ACK9	0	Generate an interrupt when ACK9 in FIFO Interrupt Status 2 register is asserted.
D22	R/W	NACK9	0	Generate an interrupt when NACK9 in FIFO Interrupt Status 2 register is asserted.

Table 456: FIFO Interrupt Enable 2 register

Bits	Access	Mnemonic	Reset	Description
D21	R/W	ERROR9	0	Generate an interrupt when ERROR9 in FIFO Interrupt Status 2 register is asserted.
D20:16	N/A	Reserved	N/A	Not valid in DMA mode.
D15	R/W	ACK8	0	Generate an interrupt when ACK8 in FIFO Interrupt Status 2 register is asserted.
D14	R/W	NACK8	0	Generate an interrupt when NACK8 in FIFO Interrupt Status 2 register is asserted.
D13	R/W	ERROR8	0	Generate an interrupt when ERROR8 in FIFO Interrupt Status 2 register is asserted.
D12:08	N/A	Reserved	N/A	Not valid in DMA mode.
D07	R/W	ACK7	0	Generate an interrupt when ACK7 in FIFO Interrupt Status 2 register is asserted.
D06	R/W	NACK7	0	Generate an interrupt when NACK7 in FIFO Interrupt Status 2 register is asserted.
D05	R/W	ERROR7	0	Generate an interrupt when ERROR7 in FIFO Interrupt Status 2 register is asserted.
D04:00	N/A	Reserved	N/A	Not valid in DMA mode.

Table 456: FIFO Interrupt Enable 2 register

FIFO Interrupt Enable 3 register

Address: 9010 3034

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used								ACK 13	NACK 13	ERROR 13	Reserved				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACK 12	NACK 12	ERROR 12	Reserved					ACK 11	NACK 11	ERROR 11	Reserved				

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:24	N/A	Not used	0x00	Always read as 0x00.
D23	R/W	ACK13	0	Generate an interrupt when ACK13 in FIFO Interrupt Status 3 register is asserted.
D22	R/W	NACK13	0	Generate an interrupt when NACK13 in FIFO Interrupt Status 3 register is asserted.
D21	R/W	ERROR13	0	Generate an interrupt when ERROR13 in FIFO Interrupt Status 3 register is asserted.
D20:16	N/A	Reserved	N/A	Not valid in DMA mode.
D15	R/W	ACK12	0	Generate an interrupt when ACK12 in FIFO Interrupt Status 3 register is asserted.
D14	R/W	NACK12	0	Generate an interrupt when NACK12 in FIFO Interrupt Status 3 register is asserted.
D13	R/W	ERROR12	0	Generate an interrupt when ERROR12 in FIFO Interrupt Status 3 register is asserted.
D12:08	N/A	Reserved	N/A	Not valid in DMA mode.
D07	R/W	ACK11	0	Generate an interrupt when ACK11 in FIFO Interrupt Status 3 register is asserted,
D06	R/W	NACK11	0	Generate an interrupt when NACK11 in FIFO Interrupt Status 3 register is asserted.
D05	R/W	ERROR11	0	Generate an interrupt when ERROR11 in FIFO Interrupt Status 3 register is asserted.
D04:00	N/A	Reserved	N/A	Not valid in DMA mode.

Table 457: FIFO Interrupt Enable 3 register

FIFO Packet Control registers

Address: 9010 3080 / 3084 / 3088 / 308C / 3090 / 3094 / 3098 / 309C / 30A0 / 30A4 / 30A8 / 30AC / 30B0

The FIFO Packet Control registers contain packet information for the device block FIFOs. There are 13 of these registers in the USB module, one for each non-control endpoint and the two required for the bidirectional control endpoint.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used		MAX										Not used			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT															

Register bit assignment

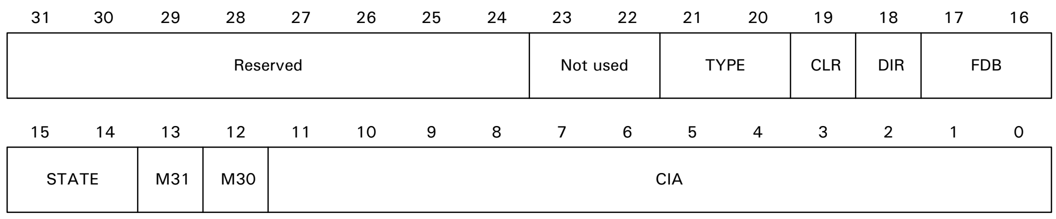
Bits	Access	Mnemonic	Reset	Description
D31:30	N/A	Not used	0x0	Always read as 0x0.
D29:20	R/W	MAX	0x040	Indicates the maximum packet size supported by the associated USB device endpoint. This value should be set to the same value as the maximum packet size (see page 766) in the Endpoint Descriptor register. **Note:** This field does not apply for FIFO #2 as that FIFO is dedicated to the IN direction of the control endpoint.
D19:16	N/A	Not used	0x00	Always read as 0x00.
D15:00	R	COUNT	0x0000	Indicates the number of error-free packets sent by the USB device module (USB-IN transactions) with the current DMA buffer descriptor for the associated FIFO. **Note:** This field does not apply for FIFO #1 as that FIFO is dedicated to the OUT direction of the control endpoint.

Table 458: FIFO Packet Control registers

FIFO Status and Control registers

Address: 9010 3100 / 3108 / 3110 / 3118 / 3120 / 3128 / 3130 / 3138 / 3140 / 3148 / 3150 / 3158 / 3160

The FIFO Status and Control registers contain additional status and control information for the device block FIFOs. There are 13 of these registers in the USB module, one for each non-control endpoint and the two required for the bidirectional endpoint.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:24	N/A	Reserved	N/A	N/A
D23:22	R/W	Not used	0x0	Always write as 00.
D21:20	R/W	TYPE	0x0	**Type field** Defines the endpoint type associated with the FIFO. 00 Control 01 Isochronous 10 Bulk 11 Interrupt
D19	R/W	CLR	1	**Clear field** Writing a 1 places the FIFO into the reset state, disabling the associated endpoint. Any data currently in the FIFO is flushed. When the endpoint is configured for the IN direction and the type is either bulk or interrupt, this bit is also set to 1 by the hardware when a packet error is found by the host.
D18	R/W	DIR	0	**Direction field** 0 Defines the OUT direction 1 Defines the IN direction Defines the FIFO (endpoint) direction.
D17:16	R	FDB	0x0	**Valid data bytes** Indicates the valid number of bytes in the FIFO when receiving; this field is always 0 when transmitting. **Note:** This field is not used when using the DMA controller.

Table 459: FIFO Status and Control registers

Bits	Access	Mnemonic	Reset	Description
D15:14	R	STATE	0x0	**State field** Defines the state of the endpoint after the most recent communication with the USB device module. 00 Undefined 01 Data phase transaction 10 Status phase transaction 11 No-data status phase transaction This field is used primarily for diagnostic purposes.
D13	R	M31	0	**Successful transfer status bit** 0 Unsuccessful transfer 1 Successful transfer **For successful transfers** ■ For IN packets: If the host sends an ACK handshake (when the command involves a handshake), this bit is set, indicating that the data is transferred successfully to the host. ■ For OUT and SETUP packets: When set, this bit also indicates that an ACK handshake is being sent to the host. **For unsuccessful transfers** ■ For OUT packets with errors, this bit is cleared. ■ For IN packets, this bit is cleared if an ACK handshake was not sent.
D12	R	M30	0	**Setup command status bit** 0 Current transaction is not a setup command 1 Current transaction is a setup command
D11:00	R	CIA	0x000	**Configuration Interface Alternate** Value depends on setting of the M30 field (D12 in this register). ■ If M30 = 1, this field contains the value 0x100. ■ If the M30 = 0, this field contains the following: [11:08] — configuration [07:04] — interface [03:00] — alternate

Table 459: FIFO Status and Control registers

Timing

CHAPTER 17

This chapter provides the electrical specifications, or timing, integral to the operation of the NS9750. Timing includes information about DC and AC characteristics, output rise and fall timing, and crystal oscillator specifications.

Electrical characteristics

The NS9750 operates at a 1.5V core, with 3.3V I/O ring voltages.

Absolute maximum ratings

Permanent device damage can occur if the absolute maximum ratings are exceeded for even an instant.

Parameter	Symbol†	Rating	Unit
DC supply voltage	V_{DDA}	-0.3 to +3.9	V
DC input voltage	V_{INA}	-0.3 to V_{DDA}+0.3	V
DC output voltage	V_{OUTA}	-0.3 to V_{DDA}+0.3	V
DC input current	I_{IN}	±10	mA
Storage temperature	T_{STG}	-40 to +125	^{o}C
† V_{DDA}, V_{INA}, V_{OUTA}: Ratings of I/O cells for 3.3V interface			

Table 460: Absolute maximum ratings

Recommended operating conditions

Recommended operating conditions specify voltage and temperature ranges over which a circuit's correct logic function is guaranteed. The specified DC electrical characteristics are satisfied over these ranges.

Parameter	Symbol†	Rating	Unit
DC supply voltage	V_{DDA}	3.0 to 3.6	V
	V_{DDC} (core)	1.4 to 1.6	V
	V_{DDC} (PLL)	1.425 to 1.575	V
Maximum junction temperature	T_J	125	^{o}C
† V_{DDA}: Ratings of I/O cells for 3.3V interface V_{DDC}: Ratings of internal cells			

Table 461: Recommended operating conditions

Maximum power dissipation

Table 462 shows the maximum power dissipation, including sleep mode information, for I/O and core.

Operation				Sleep mode with wake up on			
CPU clock	**Full**	**No PCI**	**No PCI, LCD**	**All ports**	**BBus ports**	**AHB bus ports**	**No wake up ports**
Total@ 200 MHZ	1.7W	1.55 W	1.5 W	350 mW	285 mW	240 mW	180 mW
Core	1.05 W	1 W	1 W	260 mW	210 mW	220 mW	170 mW
I/O	0.65 W	0.55 W	0.5 W	90 mW	75 mW	20 mW	10 mW
Total @ 162 MHz	1.4 W	1.25 W	1.2 W	285 mW	235 mW	200 mW	145 mW
Core	0.9 W	0.8 W	0.8 W	210 mW	170 mW	180 mW	140 mW
I/O	0.5 W	0.45 W	0.4 W	75 mW	65 mW	20 mW	5 mW
Total @ 125 MHz	1.05 W	1 W	950 mW	220 mW	180 mW	150 mW	110 mW
Core	0.65 W	0.65 W	640 mW	210 mW	130 mW	140 mW	105 mW
I/O	0.4 W	0.35 W	310 mW	75 mW	50 mW	10 mW	5 mW

Table 462: NS9750 power dissipation

Typical power dissipation

The next table shows typical power dissipation for I/O and core.

Operation				
CPU clock		**Full**	**No PCI**	**No PCI, LCD**
Total@ 200 MHZ		952 mW	886 mW	809 mW
	Core	419 mW	353 mW	287 mW
	I/O	533 mW	533 mW	522 mW

DC electrical characteristics

DC characteristics specify the worst-case DC electrical performance of the I/O buffers that are guaranteed over the specified temperature range.

Inputs

All electrical inputs are 3.3V interface.

Note: V_{SS} = 0V (GND)

Sym	Parameter	Condition	Value		Unit
V_{IH}	High-level input voltage:		Min		
	LVTTL level			2.0	V
	PCI level			$0.5V_{DDA}$	V
V_{IL}	Low-level input voltage:		Max		
	LVTTL level			0.8	V
	PCI level			$0.3V_{DDA}$	V
I_{IH}	High level input current (no pulldown)	$V_{INA}=V_{DDA}$	Min/Max	-10/10	µA
	Input buffer with pulldown		Min/Max	10/200	µA
I_{IL}	Low-level input current (no pullup	$V_{INA}=V_{SS}$	Min/Max	-10/10	µA
	Input buffer with pullup		Min/Max	10/200	µA
I_{OZ}	High-impedance leakage current	$V_{OUTA}=V_{DDA}$ or V_{SS}	Min/Max	-10/10	µA
I_{DDS}	Quiescent supply current	$V_{INA}=V_{DDA}$ or V_{SS}	Max	TBD	

Table 463: DC electrical inputs

USB DC electrical inputs

Symbol	Parameter	Min	Max	Units	Notes
V_{IH}	Input high level (driven)	2.0	V_{DDA}-0.6	V	
V_{IZ}	Input high level (floating)	2.7	3.6	V	

Table 464: USB DC electrical inputs

Symbol	Parameter	Min	Max	Units	Notes
V_{IL}	Input low level		0.8	V	
V_{DI}	Differential input sensitivity	0.2		V	1
V_{CM}	Differential common mode range	0.8	2.5	V	2

Table 464: USB DC electrical inputs

Notes:

1 |(usb_dp) – (usb_dm)|

2 Includes V_{DI} range.

Outputs

All electrical outputs are 3.3V interface.

Sym	Parameter	Value		Unit
V_{OH}	High-level output voltage (LVTTL)	Min	V_{DDA}-0.6	V
V_{OL}	Low-level output voltage (LVTTL)	Max	0.4	V
V_{OH}	PCI high-level output voltage	Min	$0.9V_{DDA}$	V
V_{OL}	PCI low-level output voltage	Max	$0.1V_{DDA}$	V

Table 465: DC electrical outputs

USB DC electrical outputs

Symbol	Parameter	Min	Max	Units	Notes
V_{OL}	Output low level	0.0	0.3	V	1
V_{OH}	Output high level	2.8	3.6	V	2
V_{CRS}	Output signal crossover voltage	1.3	2.0	V	3

Table 466: USB DC electrical outputs

Notes:

1 Measured with R_L of 1.425k ohm to 3.6V.

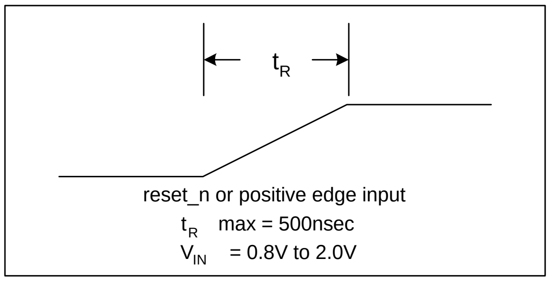
2 Measured with R_L of 14.25k ohm to GND.

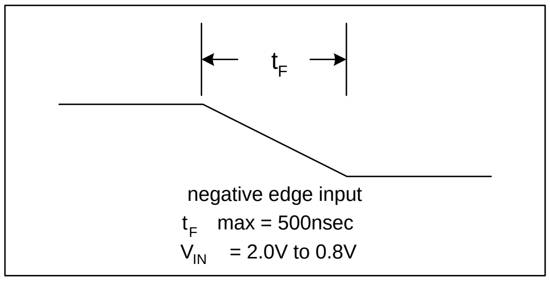
3 Excluding the first transition from the idle state.

Reset and edge sensitive input timing requirements

The critical timing requirement is the rise and fall time of the input. If the rise time is too slow for the reset input, the hardware strapping options may be registered incorrectly. If the rise time of a positive-edge-triggered external interrupt is too slow, then an interrupt may be detected on both the rising and falling edge of the input signal.

A maximum rise and fall time must be met to ensure that reset and edge sensitive inputs are handled correctly. With Digi processors, the maximum is 500 nanoseconds as shown:



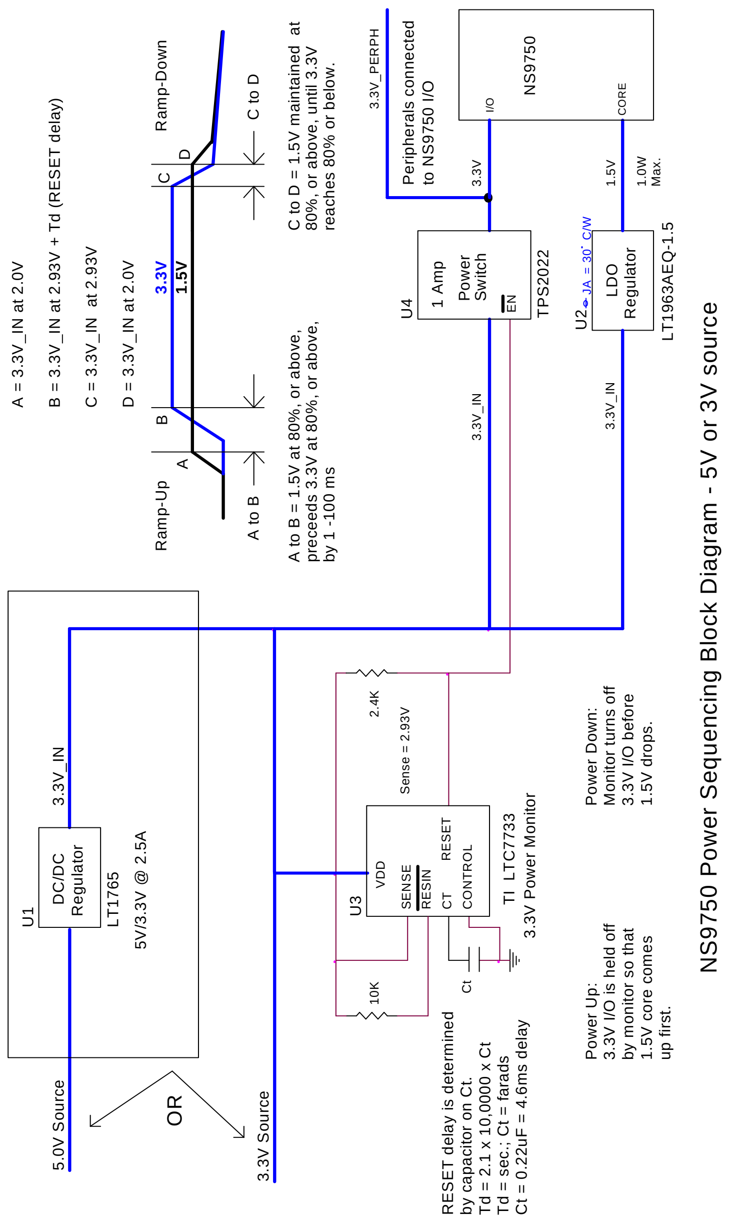


If an external device driving the reset or edge sensitive input on a Digi processor cannot meet the 500ns maximum rise and fall time requirement, the signal must be buffered with a Schmitt trigger device. Here are sample Schmitt trigger device part numbers:

Manufacturer	Part number	Description
Fairchild	NC7SP17	Single Schmitt trigger buffer, available in 5-lead SC70 and 6-lead MicroPak packages
Philips	74LVC1G17GW	Single Schmitt trigger buffer, available in 5-lead SC70 and SOT 353 packages
TI	SN74LVC1G17DCK	Single Schmitt trigger buffer, available in 5-lead SC70 and SOT 353 packages
ON Semi	NL17SZ17DFT2	Single Schmitt trigger buffer, available in 5-lead SC70 and SOT 353 packages.

Power sequencing

Use these requirements for power sequencing:


A = 3.3V_IN at 2.0V
B = 3.3V_IN at 2.93V + Td (RESET delay)
C = 3.3V_IN at 2.93V
D = 3.3V_IN at 2.0V
Ramp-Up
Ramp-Down
3.3V
1.5V
A
B
C
D
A to B
C to D
A to B = 1.5V at 80%, or above, preceeds 3.3V at 80%, or above, by 1 -100 ms
C to D = 1.5V maintained at 80%, or above, until 3.3V reaches 80% or below.
3.3V_PERPH
Peripherals connected to NS9750 I/O
NS9750
I/O
CORE
3.3V
1.5V
1.0W Max.
U4
1 Amp Power Switch
EN
TPS2022
U2
θJA = 30° C/W
LDO Regulator
LT1963AEQ-1.5
3.3V_IN
3.3V_IN
U1
DC/DC Regulator
LT1765
5V/3.3V @ 2.5A
3.3V_IN
5.0V Source
OR
3.3V Source
2.4K
Sense = 2.93V
U3
VDD
SENSE
RESIN
CT
CONTROL
RESET
TI LTC7733
3.3V Power Monitor
10K
Ct
RESET delay is determined by capacitor on Ct.
Td = 2.1 x 10,0000 x Ct
Td = sec.; Ct = farads
Ct = 0.22uF = 4.6ms delay
Power Up:
3.3V I/O is held off by monitor so that 1.5V core comes up first.
Power Down:
Monitor turns off 3.3V I/O before 1.5V drops.
NS9750 Power Sequencing Block Diagram - 5V or 3V source

Memory timing

Note: All AC characteristics are measured with 35pF, unless otherwise noted.

Memory timing contains parameters and diagrams for both SDRAM and SRAM timing.

Table 467 describes the values shown in the SDRAM timing diagrams (Figure 104 through Figure 112).

Parm	Description	Min	Max	Unit	Notes
M1	data input setup time to rising	1.6		ns	
M2	data input hold time to rising	3.3		ns	
M3	clk_out high to clk_en high		6.1	ns	
M4	clk_out high to address valid		6.1	ns	
M5	clk_out high to data_mask		6.1	ns	1, 2
M6	clk_out high to dy_cs_n low		6.1	ns	3, 4
M7	clk_out high to ras_n low		6.1	ns	
M8	clk_out high to cas_n low		6.1	ns	
M9	clk_out high to we_n low		6.1	ns	
M10	clk_out high to data out		6.2	ns	
M11	address hold time	3.5			
M12	data out hold time	3.8			
M13	clk_en high to sdram access	2	2	clock	
M14	end sdram access to clk_en low	2	2	clock	

Table 467: SDRAM timing parameters

Notes:

1 All four data_mask signals are used for all transfers.

2 All four data_mask signals will go low during a read cycle, for both 16-bit and 32-bit transfers.

3 Only one of the four clk_out signals is used.

4 Only one of the four dy_cs_n signals is used.

SDRAM burst read (16-bit)

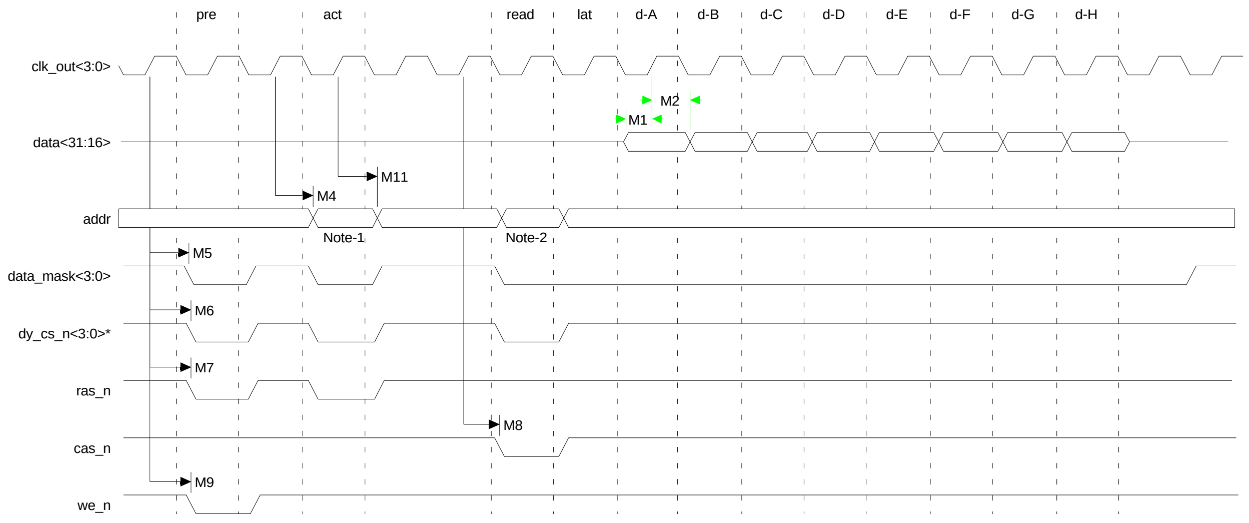


Figure 104: SDRAM burst read (16-bit) timing

Notes:

1 This is the bank and RAS address.

2 This is the CAS address.

SDRAM burst read (16-bit), CAS latency = 3

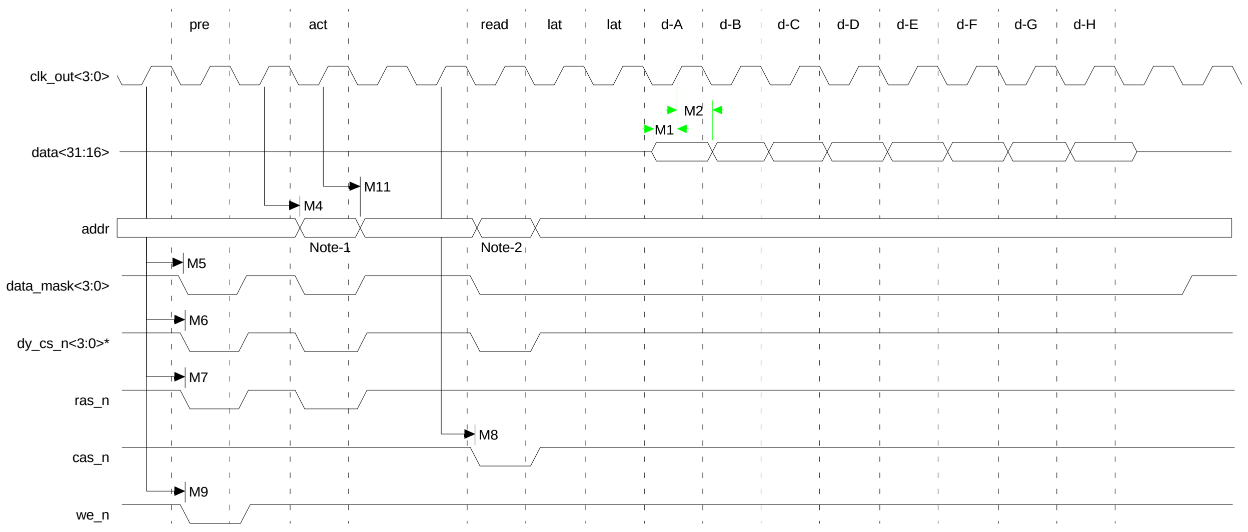


Figure 105: SDRAM burst read (16-bit), CAS latency = 3 timing

Notes:

1 This is the bank and RAS address.

2 This is the CAS address.

SDRAM burst write (16-bit)

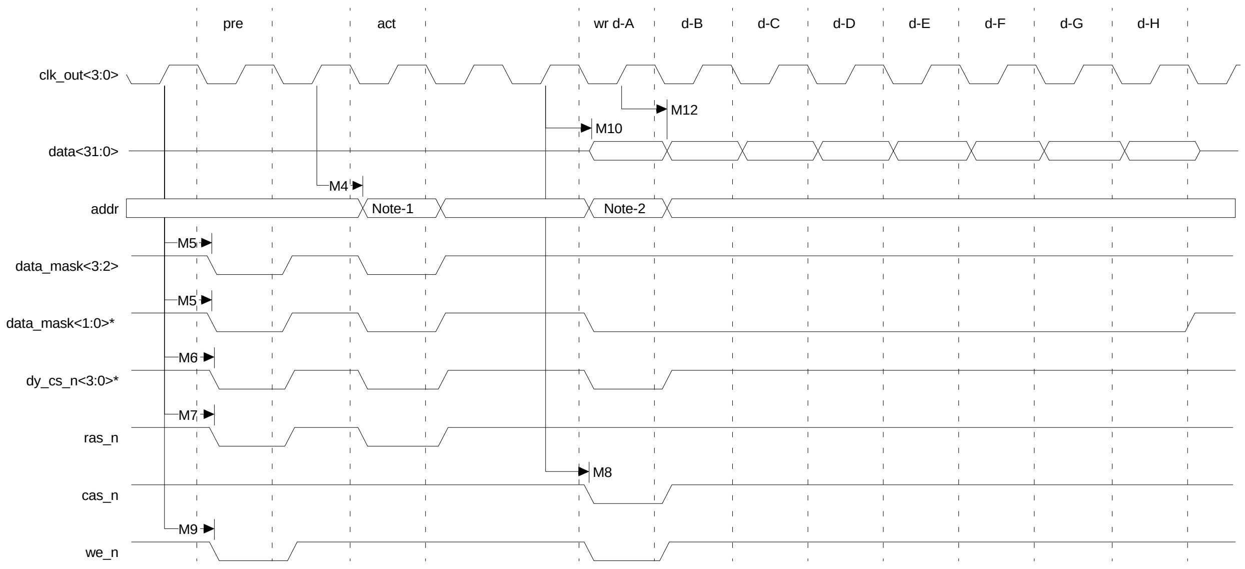


Figure 106: SDRAM burst write (16-bit) timing

Notes:

1 This is the bank and RAS address.

2 This is the CAS address.

SDRAM burst read (32-bit)

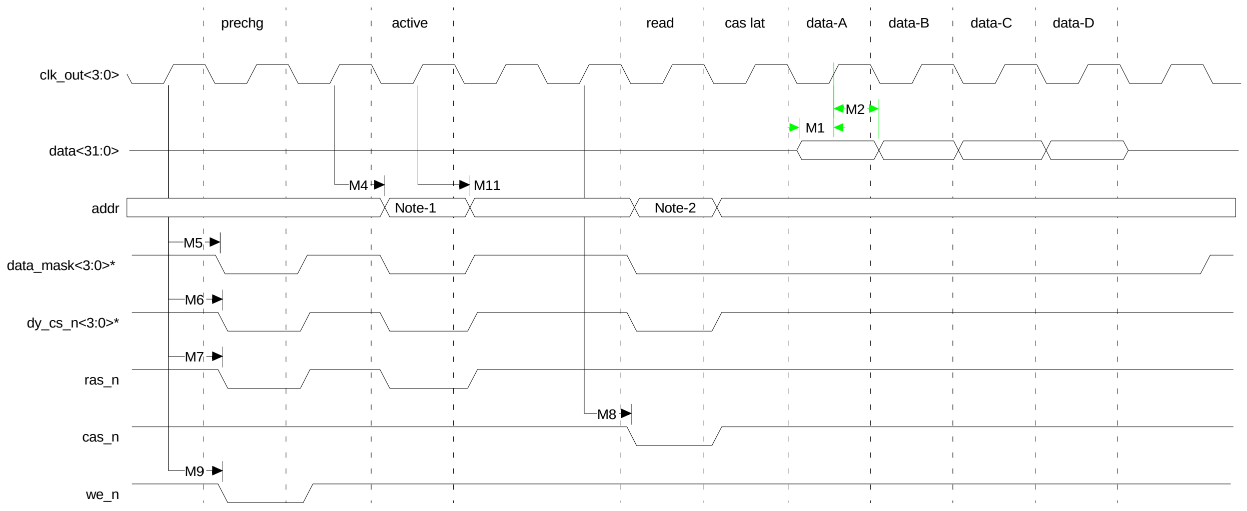


Figure 107: SDRAM burst read (32-bit) timing

Notes:

1 This is the bank and RAS address.

2 This is the CAS address.

SDRAM burst read (32-bit), CAS latency = 3

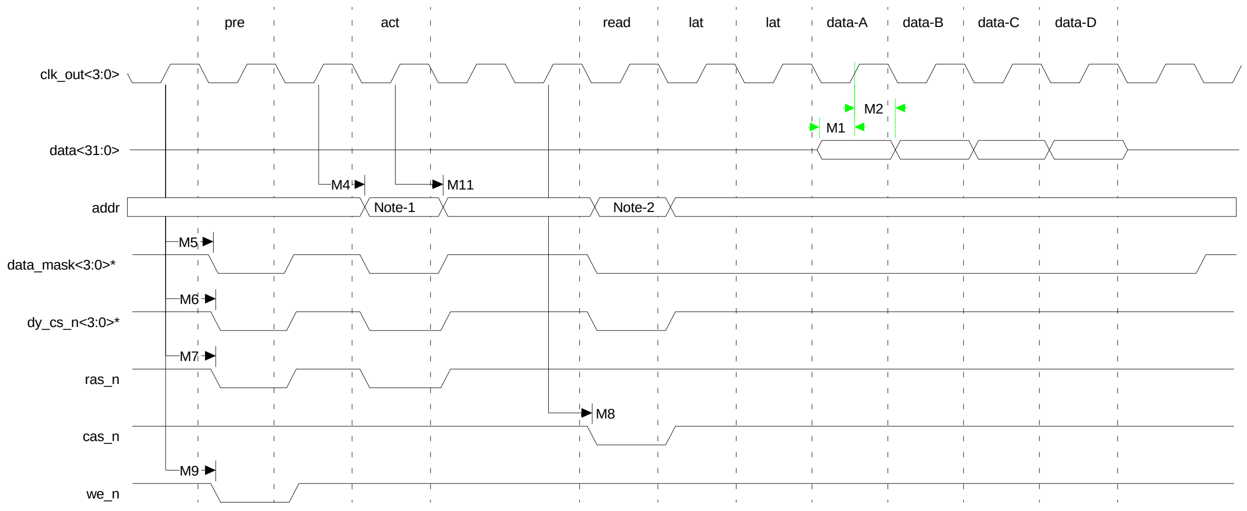


Figure 108: SDRAM burst read (32-bit), CAS latency = 3 timing

Notes:

1 This is the bank and RAS address.

2 This is the CAS address.

SDRAM burst write (32-bit)

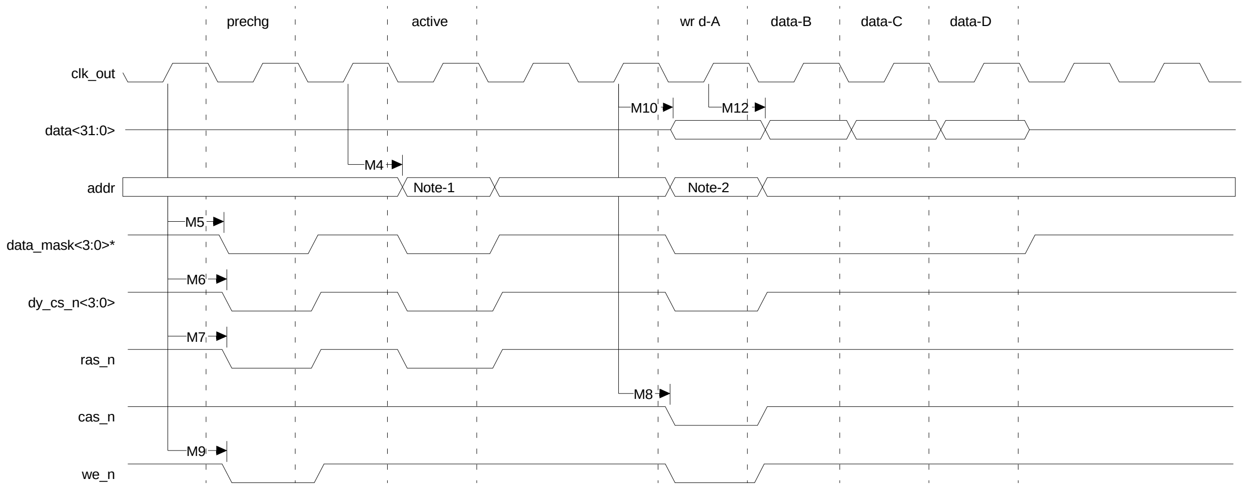


Figure 109: SDRAM burst write (32-bit) timing

Notes:

1 This is the bank and RAS address.

2 This is the CAS address.

SDRAM load mode

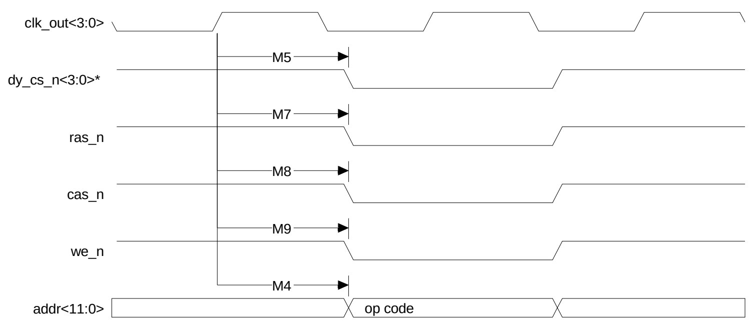


Figure 110: SDRAM load mode timing

SDRAM refresh mode

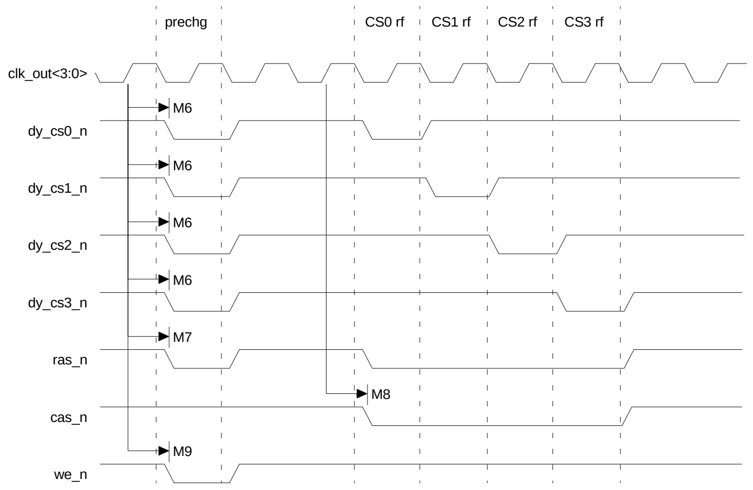


Figure 111: SDRAM refresh mode timing

Clock enable timing

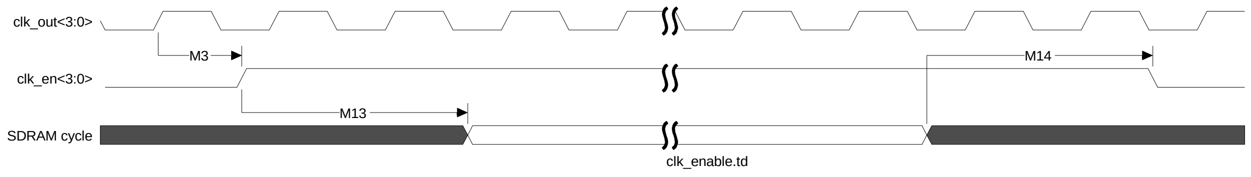


Figure 112: Clock enable timing

Table 468 describes the values shown in the SRAM timing diagrams (Figure 113 through Figure 118).

Parm	Description	Min	Max	Unit	Notes
M15	clock high to data out valid	-2	+2	ns	
M16	data out hold time from clock high	-2	+2	ns	
M17	clock high to address valid	-2	+2	ns	
M18	address hold time from clock high	-2	+2	ns	
M19	clock high to st_cs_n low	-2	+2	ns	2
M20	clock high to st_cs_n high	-2	+2	ns	2
M21	clock high to we_n low	-2	+2	ns	
M22	clock high to we_n high	-2	+2	ns	
M23	clock high to byte_lanes low	-2	+2	ns	
M24	clock high to byte_lanes high	-2	+2	ns	
M25	data input setup time to rising clk	10		ns	
M26	data input hold time to rising clk	0		ns	
M27	clock high to oe_n low	-2	+2	ns	
M28	clock high to oe_n high	-2	+2	ns	

Table 468: SRAM timing parameters

Notes:

1 The (CPU clock out / 2) signal is for reference only.

2 Only one of the four dy_cs_n signals is used. The diagrams show the active low configuration, which can be reversed (active high) with the PC field.

3 Use this formula to calculate the length of the st_cs_n signal:

Tacc + board delay + (optional buffer delays, both address out and data in) + 10ns

Static RAM read cycles with 0 wait states

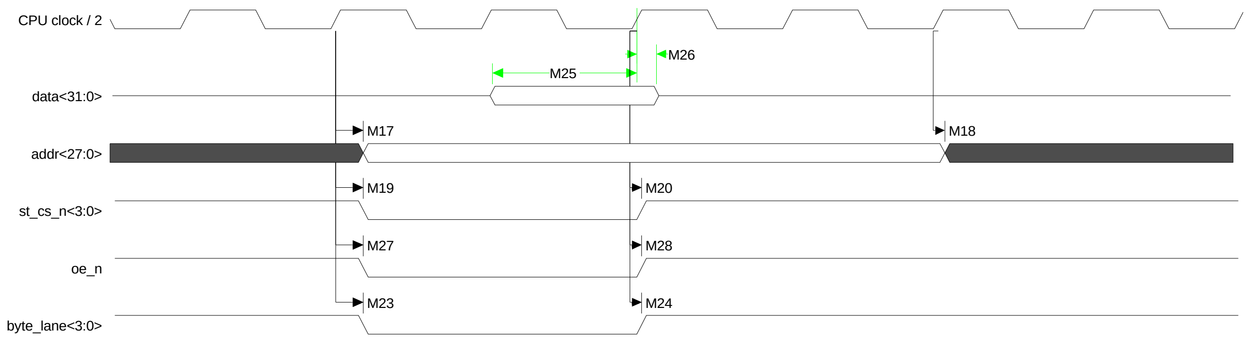


Figure 113: Static RAM read cycles with 0 wait states timing

- WTRD = 1
  WOEN = 1
- If the PB field is set to 1, all four byte_lane signals will go low for 32-bit, 16-bit, and 8-bit read cycles.
- If the PB field is set to 0, the byte_lane signal will always be high.

Static RAM asynchronous page mode read, WTPG = 1

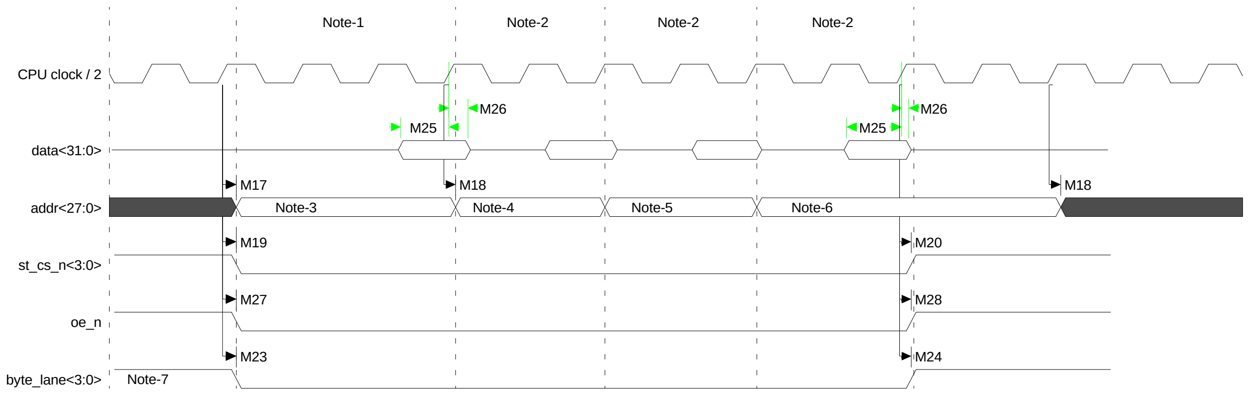


Figure 114: Static RAM asynchronous page mode read, WTPG = 1 timing

- WTPG = 1
  WTRD = 2
- If the PB field is set to 1, all four byte_lane signals will go low for 32-bit, 16-bit, and 8-bit read cycles.
- The asynchronous page mode will read 16 bytes in a page cycle. A 32-bit bus will do four 32-bit reads, as shown (3-2-2-2). A 16-bit bus will do eight 16-bit reads (3-2-2-2-3-2-2-2) per page cycle, and an 8-bit bus will do sixteen 8-bit reads (3-2-2-2-3-2-2-2-3-2-2-2-3-2-2-2) per page cycle. 3-2-2-2 is the example used here, but the WTRD and WTPG fields can set them differently.

Notes:

1 The length of the first cycle in the page is determined by the WTRD field.

2 The length of the 2nd, 3rd, and 4th cycles is determined by the WTPG field.

3 This is the starting address. The least significant two bits will always be ‘00.’

4 The least significant two bits in the second cycle will always be ‘01.’

5 The least significant two bits in the third cycle will always be ‘10.’

6 The least significant two bits in the fourth cycle will always be ‘11.’

7 If the PB field is set to 0, the byte_lane signal will always be high during a read cycle.

Static RAM read cycle with configurable wait states

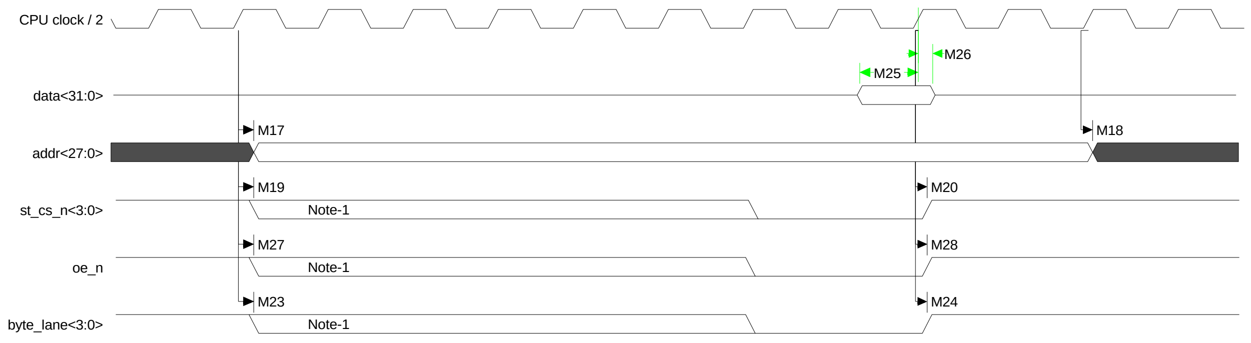


Figure 115: Static RAM read cycle with configurable wait states

- WTRD = from 1 to 15
  WOEN = from 0 to 15
- If the PB field is set to 1, all four byte_lane signals will go low for 32-bit, 16-bit, and 8-bit read cycles.
- If the PB field is set to 0, the byte_lane signal will always be high.

Static RAM sequential write cycles

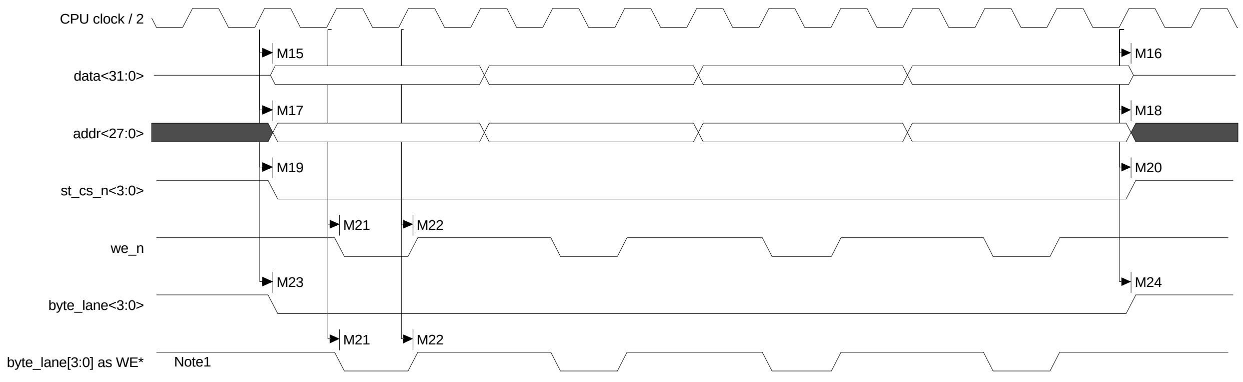


Figure 116: Static RAM sequential write cycles

- WTWR = 0
  WWEN = 0
- During a 32-bit transfer, all four byte_lane signals will go low.
- During a 16-bit transfer, two byte_lane signals will go low.
- During an 8-bit transfer, only one byte_lane signal will go low.

Note:

1 If the PB field is set to 0, the byte_lane signals will function as write enable signals and the we_n signal will always be high.

Static RAM write cycle

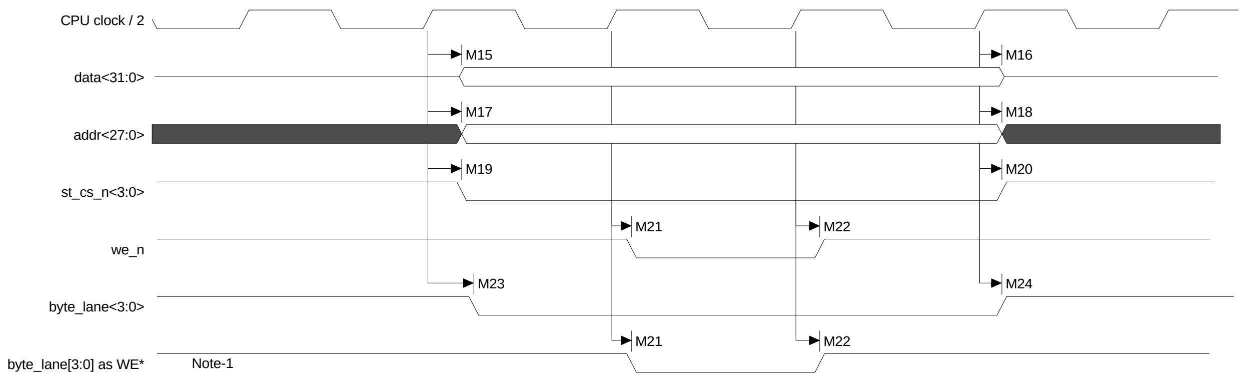


Figure 117: Static RAM write cycle

- WTWR = 0
  WWEN = 0
- During a 32-bit transfer, all four byte_lane signals will go low.
- During a 16-bit transfer, two byte_lane signals will go low.
- During an 8-bit transfer, only one byte_lane signal will go low.

Note:

1 If the PB field is set to 0, the byte_lane signals will function as write enable signals and the we_n signal will always be high.

Static write cycle with configurable wait states

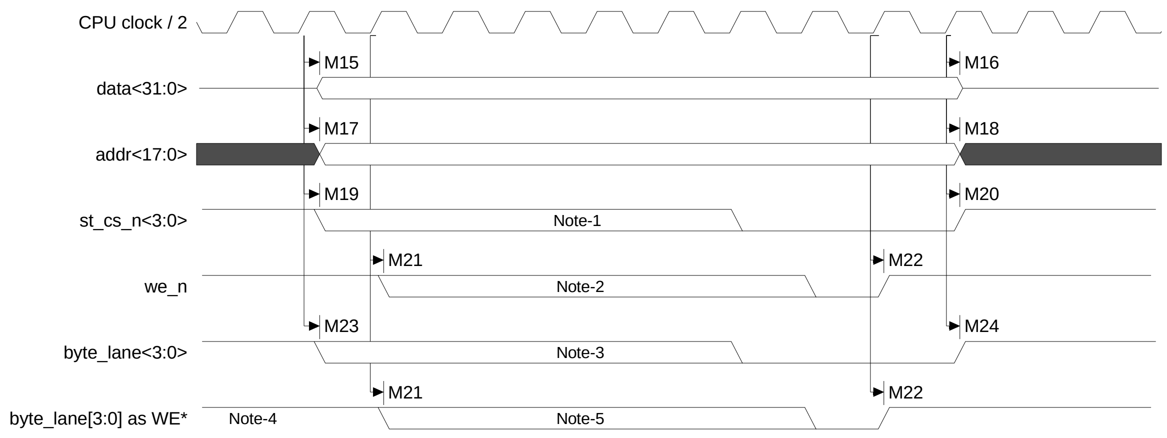


Figure 118: Static write cycle with configurable wait states

- WTWR = from 0 to 15
  WWEN = from 0 to 15
- The WTWR field determines the length on the write cycle.
- During a 32-bit transfer, all four byte_lane signals will go low.
- During a 16-bit transfer, two byte_lane signals will go low.
- During an 8-bit transfer, only one byte_lane signal will go low.

Notes:

1 Timing of the st_cs_n signal is determined with a combination of the WTWR and WWEN fields. The st_cs_n signal will always go low at least one clock before we_n goes low, and will go high one clock after we_n goes high.

2 Timing of the we_n signal is determined with a combination of the WTWR and WWEN fields.

3 Timing of the byte_lane signals is determined with a combination of the WTWR and WWEN fields. The byte_lane signals will always go low one clock before we_n goes low, and will go one clock high after we_n goes high.

4 If the PB field is set to 0, the byte_lane signals will function as the write enable signals and the we_n signal will always be high.

5 If the PB field is set to 0, the timing for the byte_lane signals is set with the WTWR and WWEN fields.

Slow peripheral acknowledge timing

This table describes the values shown in the slow peripheral acknowledge timing diagrams.

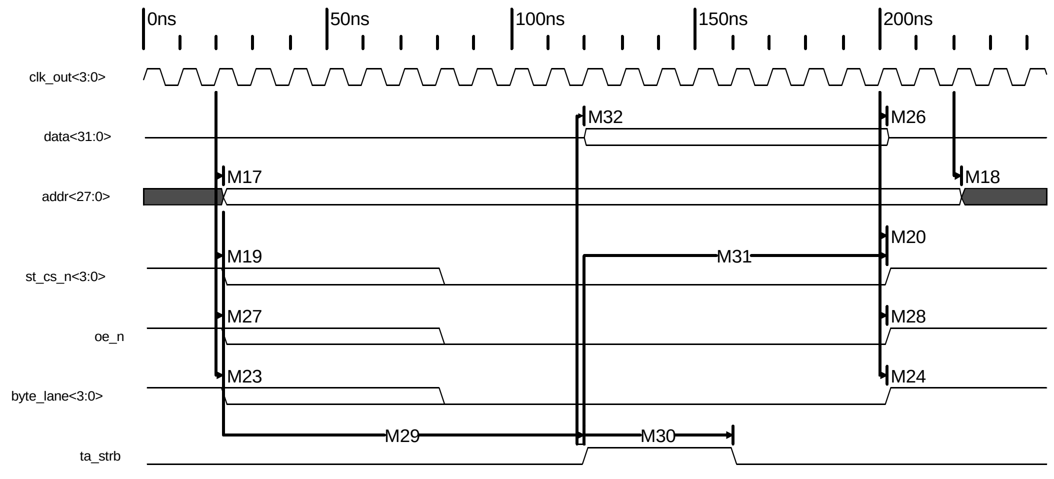
Parameter	Description	Min	Max	Unit	Notes
M15	clock high to data out valid		+2	ns	
M16	data out hold time from clock high	-2		ns	
M17	clock high to address valid		+2	ns	
M18	address hold time from clock high	-2		ns	
M19	clock high to st_cs_n low		+2	ns	1
M20	clock high to st_cs_n high		+2	ns	1
M21	clock high to we_n low		+2	ns	
M22	clock high to we_n high		+2	ns	
M23	clock high to byte_lanes low		+2	ns	
M24	clock high to byte_lanes high		+2	ns	
M26	data input hold time to rising clk	4.5		ns	
M27	clock high to oe_n low		+2	ns	
M28	clock high to oe_n high		+2	ns	
M29	address/chip select valid to ta_strb high	2		CPU cycles	
M30	ta_strb pulse width	4	8	CPU cycles	
M31	ta_strb rising to chip select/address change	4	10	CPU cycles	
M32	data setup to ta_strb rising	0		ns	

Table 469: Slow peripheral acknowledge timing parameters

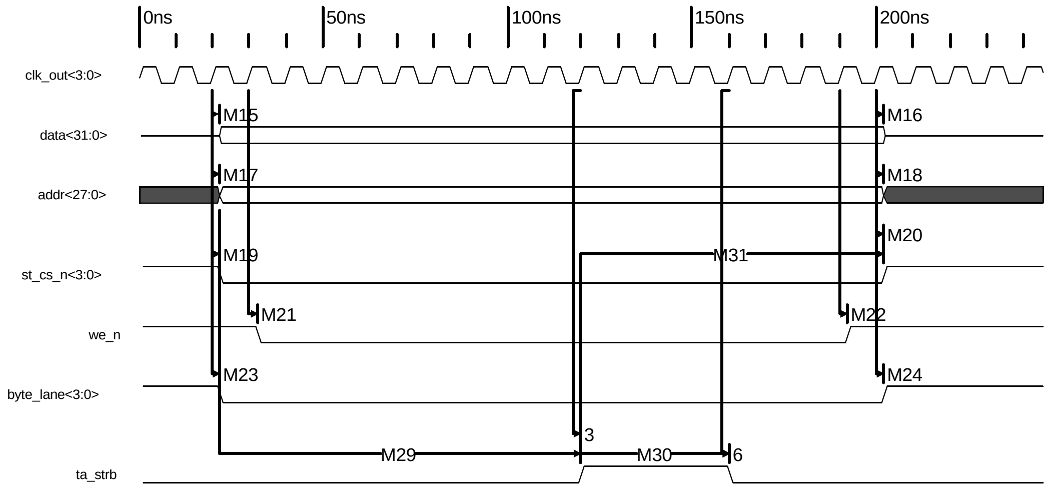
Note:

1 Only one of the four st_cs_n signals is used. The diagrams show the active low configuration, which can be reversed (active high) with the PC field.

Slow peripheral acknowledge read



Slow peripheral acknowledge write



Ethernet timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

Table 470 describes the values shown in the Ethernet timing diagrams (Figure 119 and Figure 120).

Parameter	Description	Min	Max	Unit	Notes
E1	MII tx_clk to txd, tx_en, tx_er	3	11	ns	2
E2	MII rxd, rx_en, rx_er setup to rx_clk rising	3		ns	
E3	MII rxd, rx_en, rx_er hold from rx_clk rising	1		ns	
E4	mdio (input) setup to mdc rising	10		ns	
E5	mdio (input) hold from mdc rising	0		ns	
E6	mdc to mdio (output)	18	38	ns	1,2
E7	mdc period	80		ns	
E8	RMII ref_clk to txd, tx_en	3	12	ns	2
E9	RMII rxd, crs, rx_er setup to ref_clk rising	3		ns	
E10	RMII rxd, crs, rx_er hold from ref_clk rising	1		ns	
E11	MII rx_clk to cam_req	3	10	ns	
E12	MII cam_reject setup to rx_clk rising	N/A		ns	3
E13	MII cam_reject hold from rx_clk rising	N/A		ns	3

Table 470: Ethernet timing characteristics

Notes:

1 Minimum specification is for fastest AHB bus clock of 100 MHz. Maximum specification is for slowest AHB bus clock of 50 MHz.

2 C_{load} = 10pf for all outputs and bidirects.

3 No setup and hold requirements for cam_reject because it is an asynchronous input. This is also true for RMII PHY applications.

Ethernet MII timing

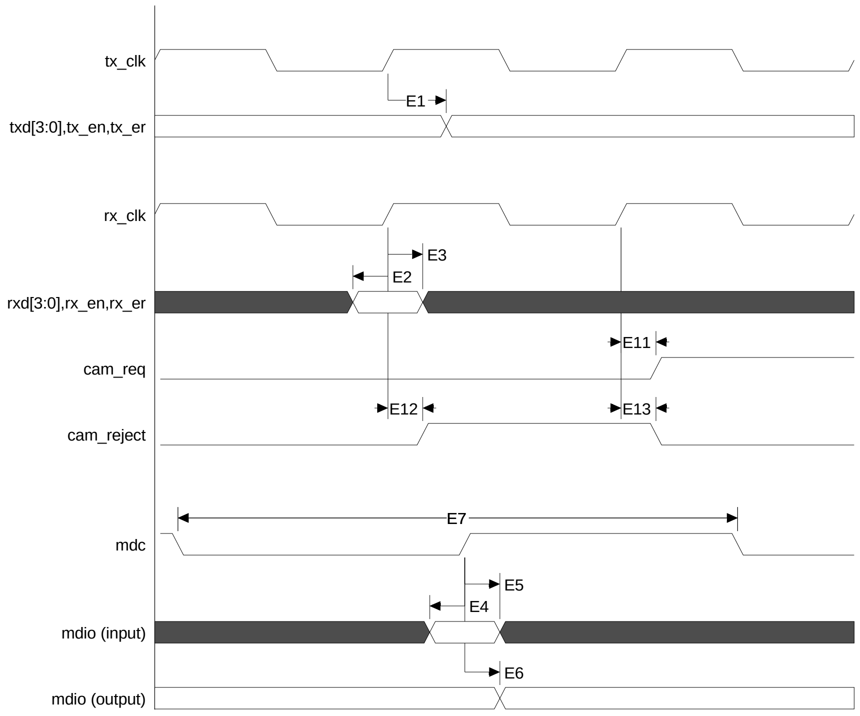


Figure 119: Ethernet MII timing

Ethernet RMII timing

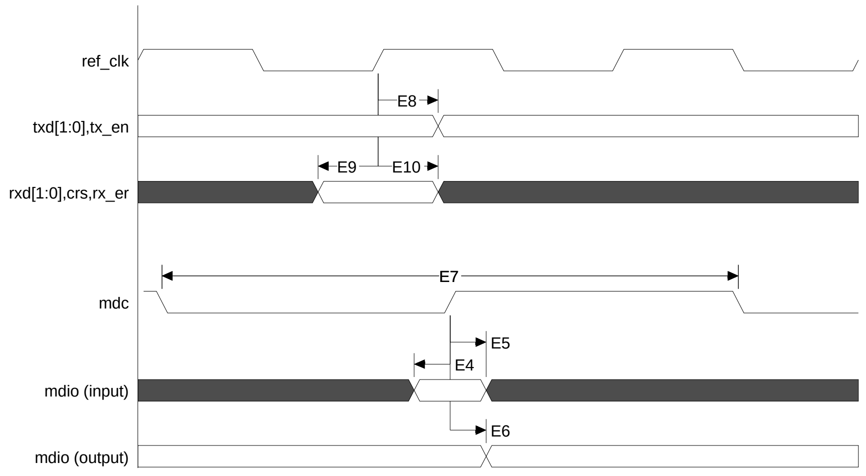


Figure 120: Ethernet RMII timing

PCI timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

Table 471 and Table 472 describe the values shown in the PCI timing diagrams (Figure 121 through Figure 127).

Parameter	Description	Min	Max	Units	Notes
P1	pci_clk_in to signal valid delay	2	9	ns	1,2
P2	Input setup to pci_clk_in	5		ns	1
P3	Input hold from pci_clk_in	0		ns	
P4	pci_clk_in to signal active	2		ns	2
P5	pci_clk_in to signal float		28	ns	2
P6	pci_clk_out high time	50%-1	50%+1	ns	3
P7	pci_clk_out low time	50%-1	50%+1	ns	3
P8	pci_clk_in cycle time	30		ns	
P9	pci_clk_in high time	11		ns	
P10	pci_clk_in low time	11		ns	

Table 471: PCI timing characteristics

Notes:

1 Parameters same for bussed and point-to-point signals.

2 CLOAD = 10pf on all outputs

3 pci_clk_out high and low times specified as 50% of the clock period +/-1 ns.

Parameter	Description	Min	Max	Units	Notes
P1	pci_clk_in to signal valid delay	2	10	ns	1
P2	Input setup to pci_clk_in	5		ns	1
P3	Input hold from pci_clk_in	0		ns	
P4	pci_clk_in to signal active	2		ns	1
P5	pci_clk_in to signal float		28	ns	1
P6	pci_clk_out high time	50%-1	50%+1	ns	2
P7	pci_clk_out low time	50%-1	50%+1	ns	2
P8	pci_clk_in cycle time	30		ns	
P9	pci_clk_in high time	11		ns	
P10	pci_clk_in low time	11		ns	

Table 472: CardBus timing characteristics

Notes:

1 Minimum times are specified with 0pf and maximum times are specified with 30pf.

2 pci_clk_out high and low times specified as 50% of the clock period +/-1 ns.

Internal PCI arbiter timing

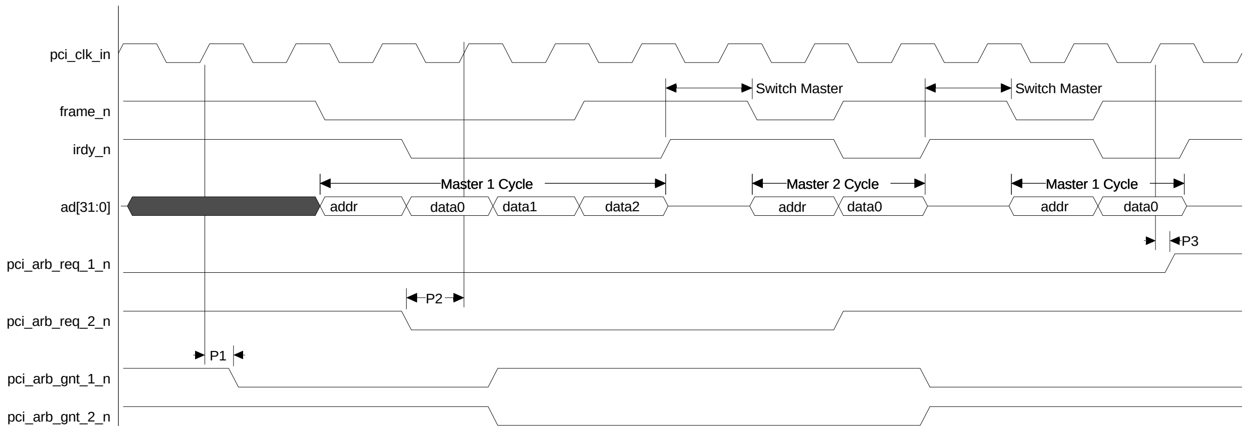


Figure 121: Internal PCI arbiter timing

PCI burst write from NS9750 timing

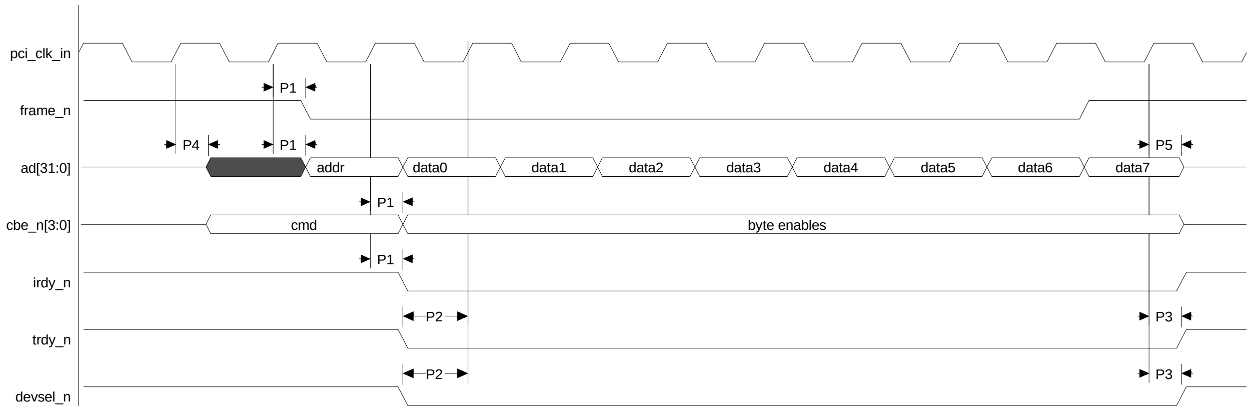


Figure 122: PCI burst write from NS9750 timing

Note: The functional timing for trdy_n and devsel_n shows the fastest possible response from the target.

PCI burst read from NS9750 timing

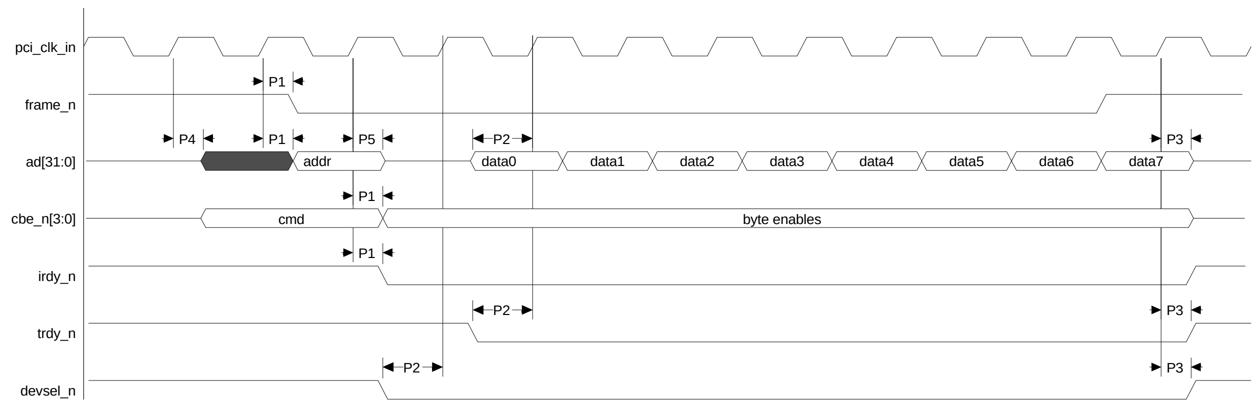


Figure 123: PCI burst read from NS9750 timing

Note: The functional timing for trdy_n, devsel_n, and the read data on ad[31:0] shows the fastest possible response from the target.

PCI burst write to NS9750 timing

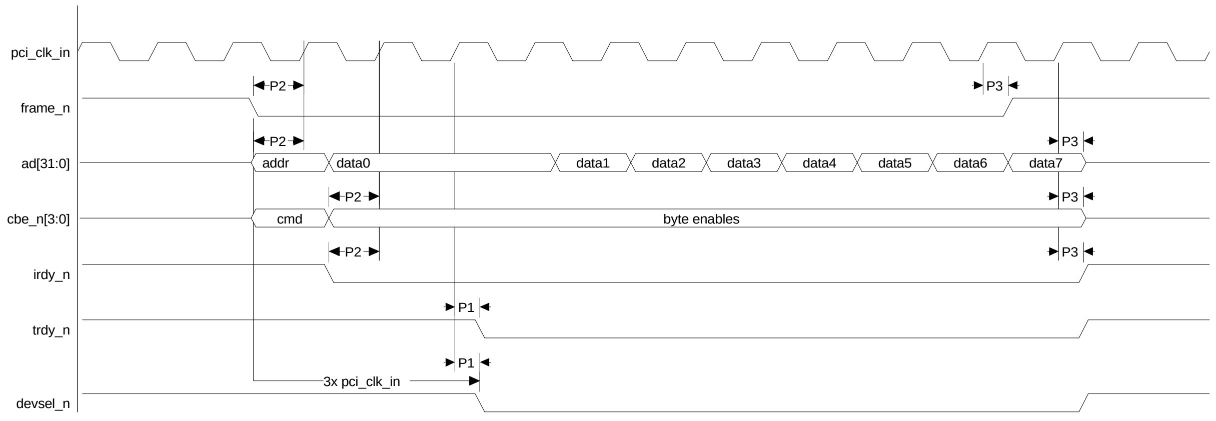


Figure 124: PCI burst write to NS9750 timing

PCI burst read to NS9750 timing

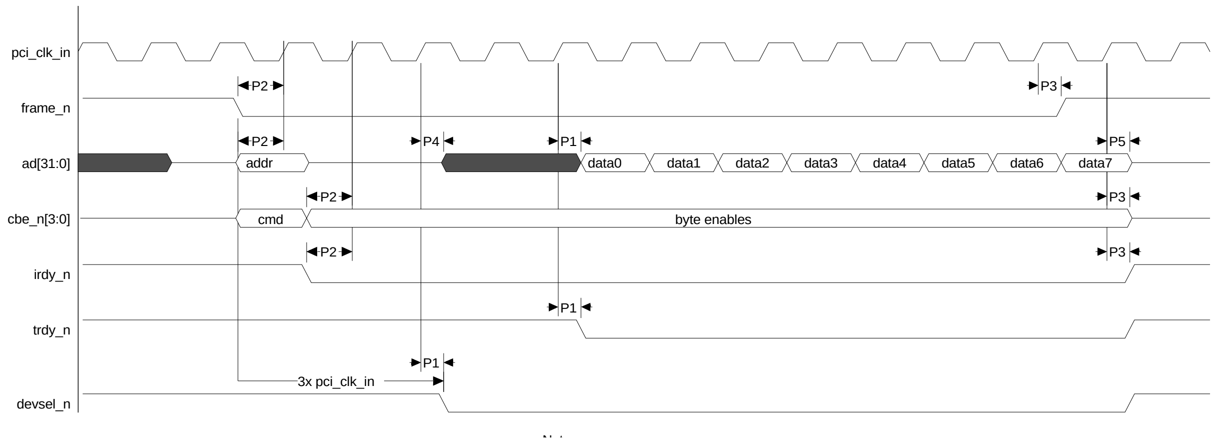


Figure 125: PCI burst read to NS9750 timing

Note: The functional timing for valid read data on ad[31:0] is just an example. The actual response time will depend on when the PCI bridge gets access to the AHB bus internal to NS9750.

PCI clock timing

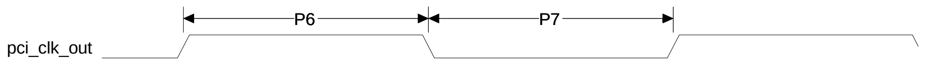


Figure 126: pci_clock_out timing

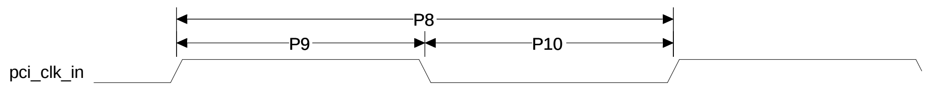


Figure 127: pci_clk_in timing

I^{2}C timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

Table 473 describes the values shown in the I^{2}C timing diagram (Figure 128).

		Standard mode		Fast mode		
Parm	**Description**	**Min**	**Max**	**Min**	**Max**	**Unit**
C1	iic_sda to iic_scl START hold time	4.0		0.6		µs
C2	iic_scl low period	4.7		1.3		µs
C3	iic_scl high period	4.7		1.3		µs
C4	iic_scl to iic_sda DATA hold time	0		0		µs
C5	iic_sda to iic_scl DATA setup time	250		100		ns
C6	iic_scl to iic_sda START setup time	4.7		0.6		µs
C7	iic_scl to iic_sda STOP setup time	4.0		0.6		µs

Table 473: I^{2}C timing parameters

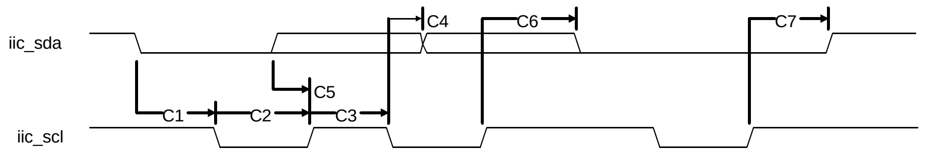


Figure 128: I^{2}C timing

LCD timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

Table 474 describes the values shown in the LCD timing diagrams (Figure 129 through Figure 135).

Parm	Description	Register	Value	Units
L1	Horizontal front porch blanking	LCDTiming0	HFP+1	CLCP periods
L2	Horizontal sync width	LCDTiming0	HSW+1	CLCP periods
L3	Horizontal period	N/A	L1+L2+L15+L4	CLCP periods
L4	Horizontal backporch	LCDTiming0	HBP+1	CLCP periods
L5	TFT active line	LCDTiming0	16*(PPL+1) (see note 3)	CLCP periods
L6	LCD panel clock frequency	LCDTiming1	**For BCD=0:** CLCDCLK/(PCD+2) **For BCD=1:** CLCDCLK (see note 1)	MHz
L7	TFT vertical sync width	LCDTiming1	VSW+1	H lines
L8	TFT vertical lines/frame	N/A	L7+L9+L10+L11	H lines
L9	TFT vertical back porch	LCDTiming1	VBP	H lines
L10	TFT vertical front porch	LCDTiming1	VFP	H lines
L11	Active lines/frame	LCDTiming1	LPP+1	H lines
L12	STN HSYNC inactive to VSYNC active	LCDTiming0	HBP+1	CLCP periods
L13	STN vertical sync width	N/A	1	H lines
L14	STN vertical lines/frame	N/A	L11+L16	H lines
L15	STN active line	LCDTiming2	CPL+1 (see note 4)	CLCP periods
L16	STN vertical blanking	LCDTiming1	VSW+VFP+VBP+1	H lines
L17	STN CLCP inactive to HSYNC active	LCDTiming0	HFP+1.5	CLCP periods

Table 474: LCD timing parameters

Parm	Description	Register	Value	Units
L18	CLCP to data/control (see notes 7 and 8)		-1.0 (min) +1.5 (max)	ns
L19	CLCP high (see notes 8, 9)		50%±0.5ns	ns
L20	CLCP low (see notes 8, 9)		50%±0.5ns	ns
L21	TFT VSYNC active to HSYNC active (see note 8)		-0.1ns (min) +0.1ns (max)	ns
L22	TFT VSYNC active to HSYNC inactive	LCDTiming0	HSW	CLCP periods
L23	STN VSYNC active to HSYNC inactive	LCDTiming0	**STN color:** 14+HSW+HFP **STN Mono8:** 6+HSW+HFP **STN Mono4:** 10+HSW+HFP	CLCP periods
L24	STN HSYNC inactive to VSYNC inactive	LCDTiming0	HBP+1	CLCP periods
L25	STN VSYNC inactive to HSYNC active	LCDTiming0	**STN color:** HFP+13 **STN Mono8:** HFP+15 **STN Mono4:** HFP+9	CLCP periods
L26	CLCP period		12.5ns (min)	ns

Table 474: LCD timing parameters

Notes:

1 CLCDCLK is selected from 5 possible sources:

— lcdclk/2 (lcdclk is an external oscillator)

— AHB clock

— AHB clock/2

— AHB clock/4

— AHB clock/8

See the LCD Controller chapter for acceptable clock frequencies for the different display configurations.

2 The polarity of CLLP, CLFP, CLCP, and CLAC can be inverted using control fields in the LCDTiming1 register.

3 The CPL field in the LCDTiming2 register must also be programmed to T5-1.

4 The PPL field in the LCDTiming0 register must also be programmed correctly.

5 These data widths are supported:

— 4-bit mono STN single panel

— 8-bit mono STN single panel

— 8-bit color STN single panel

— 4-bit mono STN dual panel (8 bits to LCD panel)

— 8-bit mono STN dual panel (16 bits to LCD panel)

— 8-bit color STN dual panel (16 bits to LCD panel)

— 24-bit TFT

— 18-bit TFT

6 See "LCDTiming0," beginning on page 580, and "LCDTiming1," beginning on page 582, for definitions of the bit fields referred to in this table.

7 Note that data is sampled by the LCD panel on the falling edge of the CLCP in Figure 135, "LCD output timing," on page 826). If the polarity of CLCP is inverted, this parameter is relative to CLCP falling, instead.

8 C_{load} = 10pf on all outputs.

9 CLCP high and low times specified as 50% of the clock period +/- 0.5ns.

10 Maximum allowable LCD panel clock frequency is 80 MHz.

Horizontal timing for STN displays

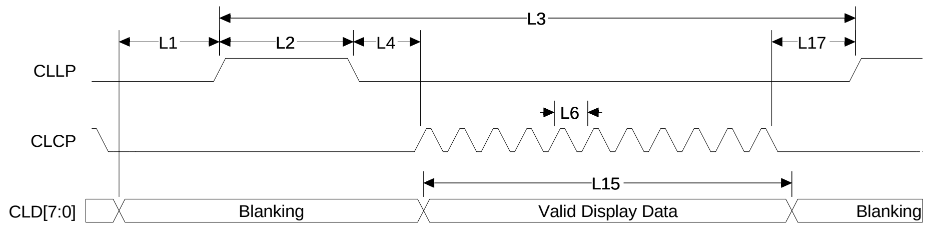


Figure 129: Horizontal timing for STN displays

Vertical timing for STN displays

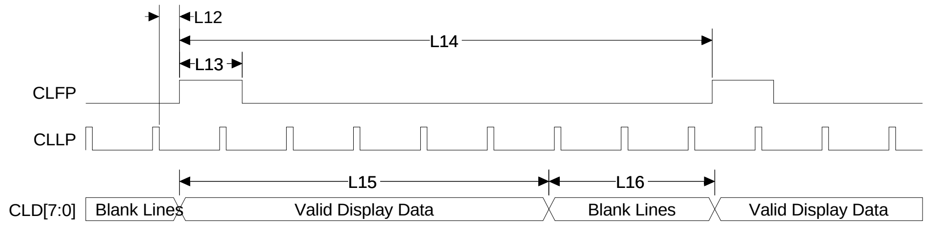


Figure 130: Vertical timing parameters for STN displays

Horizontal timing for TFT displays

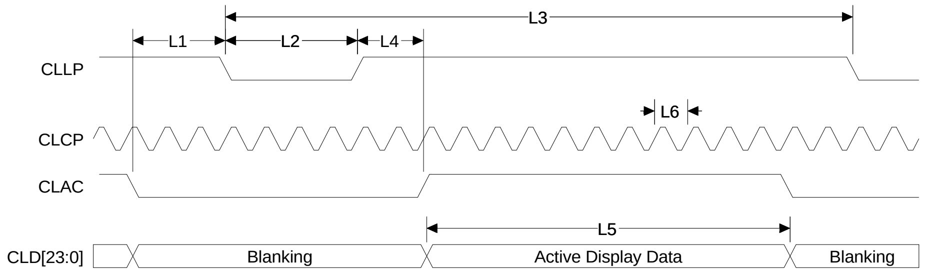


Figure 131: Horizontal timing parameters for TFT displays

Vertical timing for TFT displays

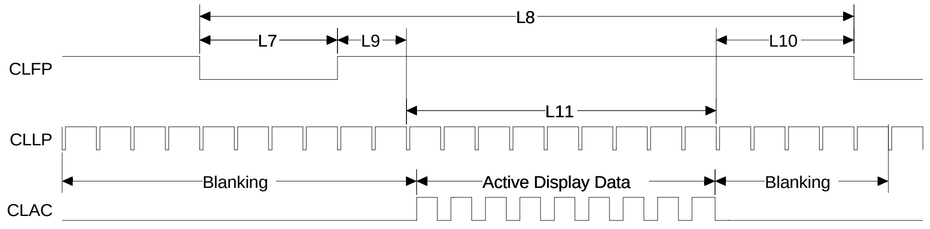


Figure 132: Vertical timing parameters for TFT displays

HSYNC vs VSYNC timing for STN displays

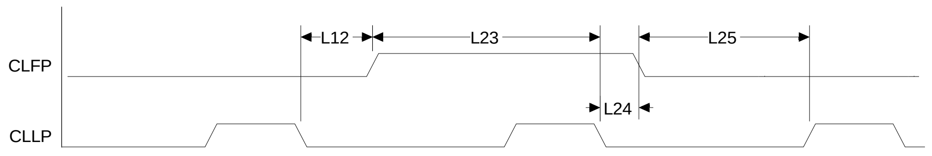


Figure 133: HSYNC vs VSYNC timing for STN displays

HSYNC vs VSYNC timing for TFT displays

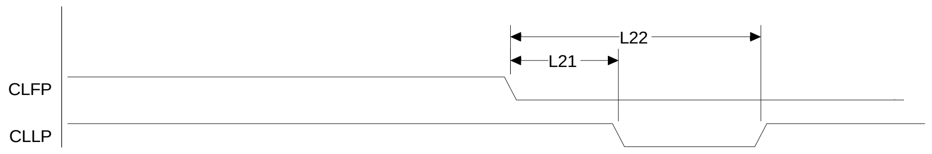


Figure 134: HSYNC vs VSYNC timing for TFT displays

LCD output timing

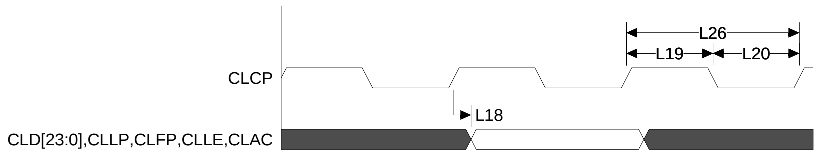


Figure 135: LCD output timing

SPI timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

Table 475 describes the values shown in the SPI timing diagrams (Figure 136 through Figure 139).

Parm	Description	Min	Max	Units	Modes	Notes
SPI master parameters						
SP0	SPI enable low setup to first SPI CLK out rising	3*T_{BCLK}-10		ns	0, 3	1, 3
SP1	SPI enable low setup to first SPI CLK out falling	3*T_{BCLK}-10		ns	1, 2	1, 3
SP3	SPI data in setup to SPI CLK out rising	30		ns	0, 3	
SP4	SPI data in hold from SPI CLK out rising	0		ns	0, 3	
SP5	SPI data in setup to SPI CLK out falling	30		ns	1, 2	
SP6	SPI data in hold from SPI CLK out falling	0		ns	1, 2	
SP7	SPI CLK out falling to SPI data out valid		10	ns	0, 3	6
SP8	SPI CLK out rising to SPI data out valid		10	ns	1, 2	6
SP9	SPI enable low hold from last SPI CLK out falling	3*T_{BCLK}-10		ns	0, 3	1, 3
SP10	SPI enable low hold from last SPI CLK out rising	3*T_{BCLK}-10		ns	1, 2	1, 3
SP11	SPI CLK out high time	SP13*45%	SPI3*55%	ns	0, 1, 2, 3	4
SP12	SPI CLK out low time	SP13*45%	SPI3*55%	ns	0, 1, 2, 3	4
SP13	SPI CLK out period	T_{BCLK}*6		ns	0, 1, 2, 3	3
SPI slave parameters						

Table 475: SPI timing parameters

Parm	Description	Min	Max	Units	Modes	Notes
SP14	SPI enable low setup to first SPI CLK in rising	30		ns	0, 3	1
SP15	SPI enable low setup to first SPI CLK in falling	30		ns	1, 2	1
SP16	SPI data in setup to SPI CLK in rising	0		ns	0, 3	
SP17	SPI data in hold from SPI CLK in rising	60		ns	0, 3	
SP18	SPI data in setup to SPI CLK in falling	0		ns	1, 2	
SP19	SPI data in hold from SPI CLK in falling	60		ns	1, 2	
SP20	SPI CLK in falling to SPI data out valid	20	70	ns	0, 3	6
SP21	SPI CLK in rising to SPI data out valid	20	70	ns	1, 2	6
SP22	SPI enable low hold from last SPI CLK in falling	15		ns	0, 3	1
SP23	SPI enable low hold from last SPI CLK in rising	15		ns	1, 2	1
SP24	SPI CLK in high time	SP26*40%	SP26*60%	ns	0, 1, 2, 3	5
SP25	SPI CLK in low time	SP26*40%	SP26*60%	ns	0,1, 2, 3	5
SP26	SPI CLK in period	T_{BCLK}*10		ns	0, 1, 2, 3	

Table 475: SPI timing parameters

Notes:

1 Active level of SPI enable is inverted (that is, 1) if the CSPOL bit in Serial Channel B/A/C/D Control Register B is set to a 1. Note that in SPI slave mode, only a value of 0 (low enable) is valid; the SPI slave is fixed to an active low chip select.

2 SPI data order is reversed (that is, LSB last and MSB first) if the BITORDR bit in Serial Channel B/A/C/D Control Register B is set to a 0.

3 T_{BCLK} is period of BBus clock.

4 ±5% duty cycle skew.

5 ±10% duty cycle skew.

6 C_{load} = 10pf for all outputs.

7 SPI data order can be reversed such that LSB is first. Use the BITORDR bit in Serial Channel B/A/C/D Control Register A.

SPI master mode 0 and 1: 2-byte transfer

(see note 7)

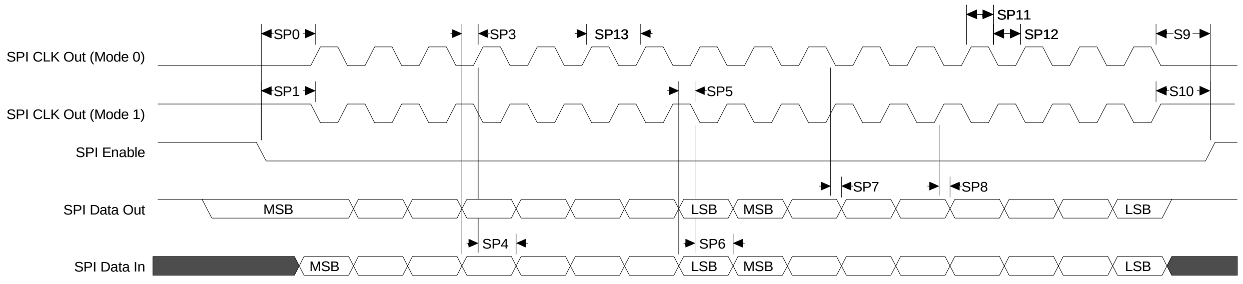


Figure 136: SPI master mode 0 and 1 (2-byte transfer)

SPI master mode 2 and 3: 2-byte transfer

(see note 7)

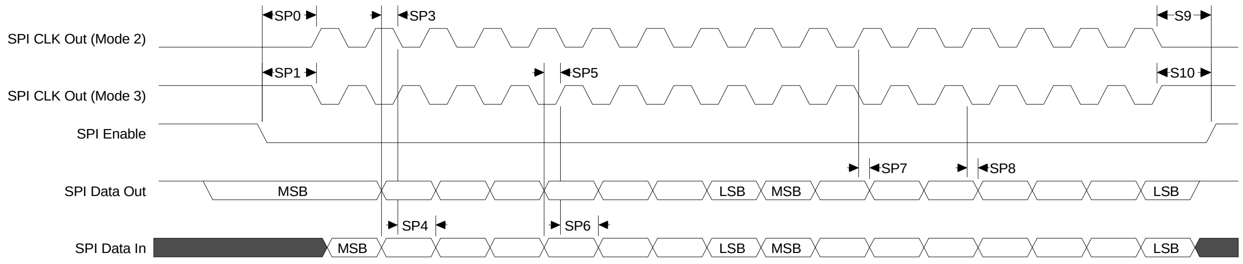


Figure 137: SPI master mode 2 and 3 (2-byte transfer)

SPI slave mode 0 and 1: 2-byte transfer

(see note 7)

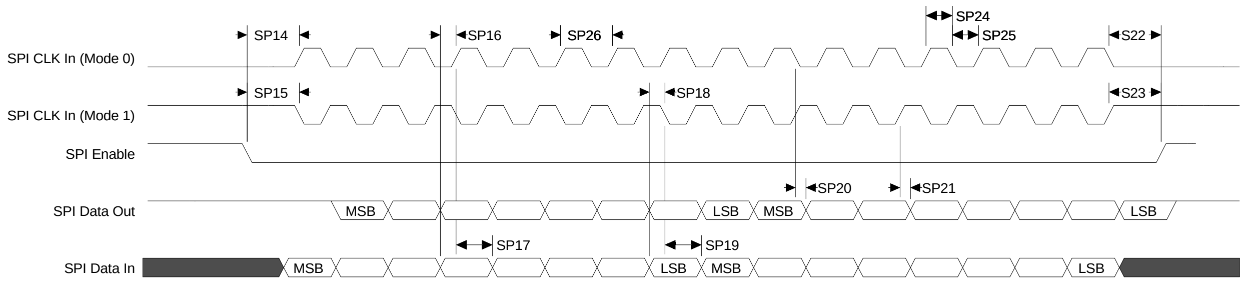


Figure 138: SPI slave mode 0 and 1 (2-byte transfer)

SPI slave mode 2 and 3: 2-byte transfer

(see note 7)

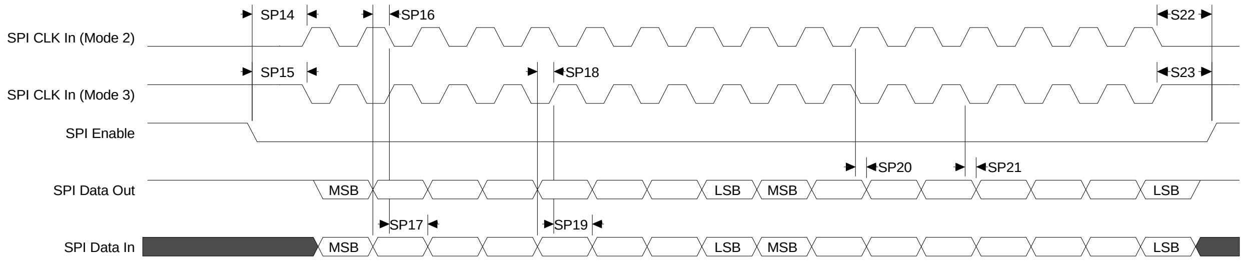


Figure 139: SPI slave mode 2 and 3 (2-byte transfer)

IEEE 1284 timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

Table 476 describes the values shown in the IEEE 1284 timing diagram (Figure 140).

Parm	Description	Min	Max	Unit	Note
IE1	Busy-while-Strobe	0	500	ns	1
IE2	Busy high to nAck low	0		ns	
IE3	Busy high		1000	ns	2
IE4	nAck low		500	ns	3
IE5	nAck high to Busy low		500	ns	3

Table 476: IEEE 1284 timing parameters

Notes:

1 The range is 0ns up to one time unit.

2 Two time units.

3 Three time units.

IEEE 1284 timing example

The IEEE 1284 timing is determined by the BBus clock and the Granularity Count register (GCR) setting. In this example, the BBus clock is 50 MHz and the Granularity Count register is set to 25. The basic time unit is 1/50 MHz x 25, which is 500ns.

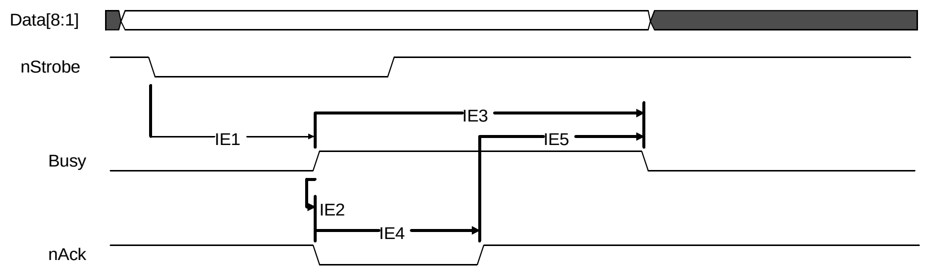


Figure 140: IEEE 1284 timing with BBus clock at 50 MHZ and GCR set to 25

USB timing

Table 477 and Table 478 describe the values shown in the USB timing diagrams (Figure 141 through Figure 143).

Parm	Description	Min	Max	Unit	Notes
U1	Rise time (10% – 90%)	4	20	ns	1
U2	Fall time (10% – 90%)	4	20	ns	1
U3	Differential rise and fall time matching	90	111.11	%	2, 5
U4	Driver output resistance	28	44	ohms	3

Table 477: USB full speed timing parameters

Parm	Description	Min	Max	Unit	Notes
U1	Rise time (10% – 90%)	75	300	ns	4
U2	Fall time (10% – 90%)	75	300	ns	4
U3	Differential rise and fall time matching	80	125	%	2, 5

Table 478: USB low speed timing parameters

Notes:

1 Load shown in Figure 142, "USB full speed load," on page 833.

2 U1/U2.

3 Includes resistance of 27 ohm ±2 ohm external series resistor.

4 Load shown in Figure 143, "USB low speed load," on page 834.

5 Excluding the first transition from the idle state.

USB differential data timing

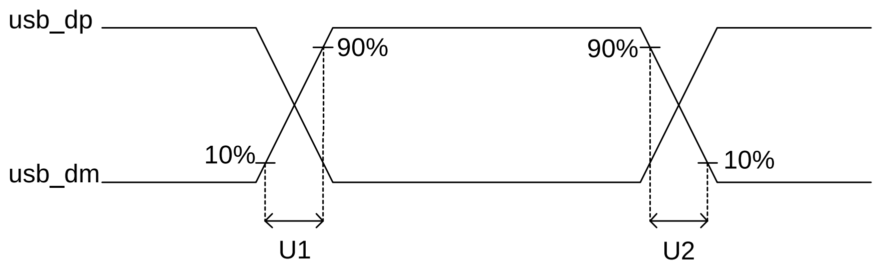


Figure 141: USB differential data

USB full speed load timing

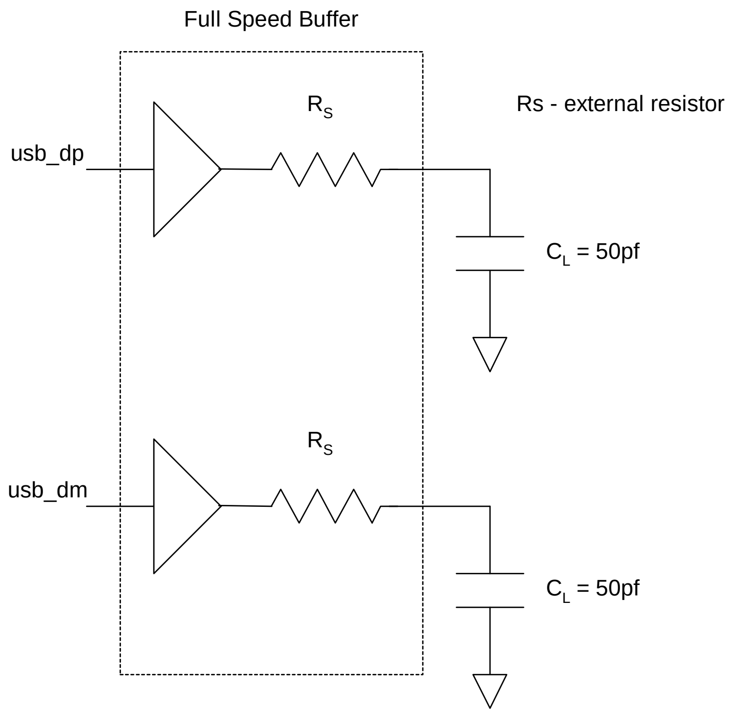


Figure 142: USB full speed load

USB low speed load

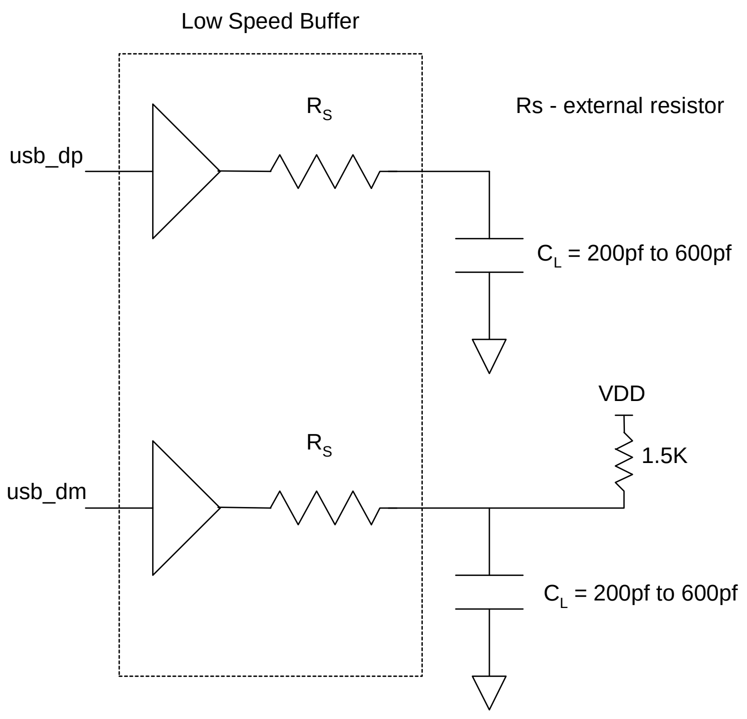


Figure 143: USB low speed load

Reset and hardware strapping timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

Table 479 describes the values shown in the reset and hardware strapping timing diagram (Figure 144).

Parm	Description	Min	Max	Unit	Notes
R1	reset_n minimum time	10		x1_sys_osc clock cycles	1
R2	reset_n to reset_done		4	ms	

Table 479: Reset and hardware strapping timing parameters

Note:

1 The hardware strapping pins are latch 5 clock cycles after reset_n is deasserted (goes high).

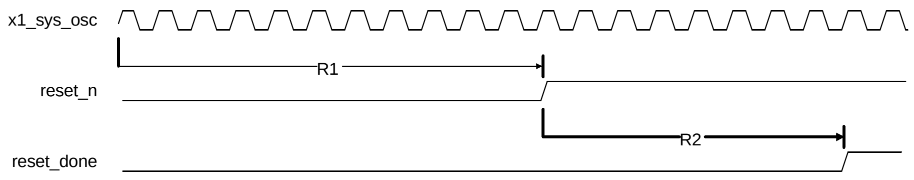


Figure 144: Reset and hardware strapping timing

- **R1:** reset_n must be held low for a minimum of 10 x1_sys_osc clock cycles after powerup.
- **R2:** reset_done is asserted 4ms after reset_n is driven high.
- The hardware strapping pins are latched when reset_done is asserted.

JTAG timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

Table 480 describes the values shown in the JTAG timing diagram (Figure 145).

Parm	Description	Min	Max	Unit
J1	tms (input) setup to tck rising	5		ns
J2	tms (input) hold to tck rising	2		ns
J3	tdi (input) setup to tck rising	5		ns
J4	tdi (input) hold to tck rising	2		ns
J5	tdo (output) to tck falling	2.5	10	ns

Table 480: JTAG timing parameters

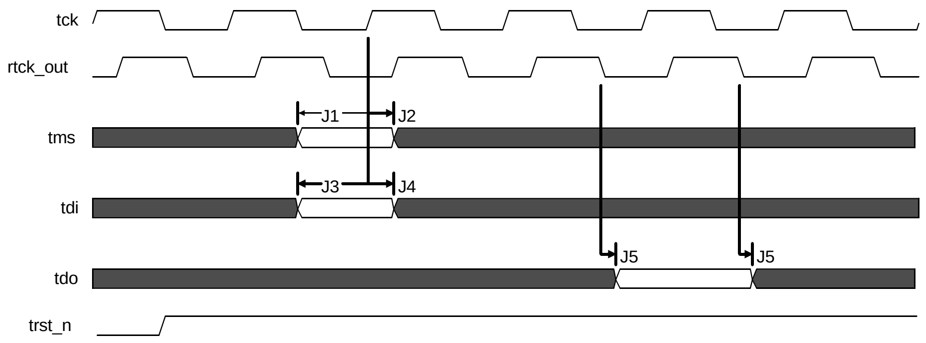


Figure 145: JTAG timing

Notes:

1 Maximum tck rate is 10 MHz.

2 rtck_out is an asynchronous output, driven off of the CPU clock.

3 trst_n is an asynchronous input.

Clock timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

The next three timing diagrams pertain to clock timing.

USB crystal/external oscillator timing

Table 481 describes the values shown in the USB crystal/external oscillator timing diagram (Figure 146).

Parm	Description	Min	Max	Unit	Notes
UC1	x1_usb_osc cycle time	20.831	20.835	ns	1
UC2	x1_usb_osc high time	(UC1/2) x 0.4	(UC1/2) x 0.6	ns	
UC3	x1_usb_osc low time	(UC1/2) x 0.4	(UC1/2) x 0.6	ns	

Table 481: USB crystal/external oscillator timing parameters

Note:

1 If using a crystal, the tolerance must be ±100 ppm or better.

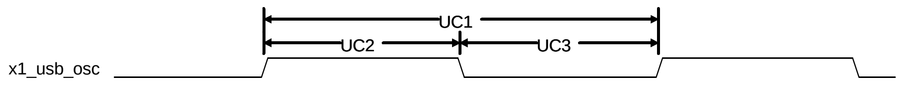


Figure 146: USB crystal/external oscillator timing

LCD input clock timing

Table 482 describes the values shown in the LCD input clock timing diagram (Figure 147).

Parm	Description	Min	Max	Unit	Notes
LC1	lcdclk cycle time	6.25		ns	1
LC2	lcdclk high time	(LC1/2) x 0.4	(LC1/2) x 0.6	ns	
LC3	lcdclk low time	(LC1/2) x 0.4	(LC1/2) x 0.6	ns	

Table 482: LCD input clock timing parameters

Note:

1 The clock rate supplied on lcdclk is twice the actual LCD clock rate.

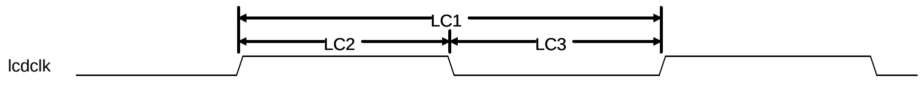


Figure 147: LCD input clock timing

System PLL bypass mode timing

Table 483 describes the values shown in the system PLL bypass mode timing diagram (Figure 148).

Parm	Description	Min	Max	Unit	Notes
SC1	x1_sys_osc cycle time	2.5	5	ns	1
SC2	x1_sys_osc high time	(SC1/2) x 0.45	(SC1/2) x 0.55	ns	
SC3	x1_sys_osc low time	(SC1/2) x 0.45	(SC1/2) x 0.55	ns	

Table 483: System PLL bypass mode timing parameters

Note:

1 The system PLL can be bypassed. In this mode, the CPU clock speed is 1/2 of x1_sys_osc.

2 The minimum high/low time on system clock input pin when used with an external oscillator is 1.5nS.

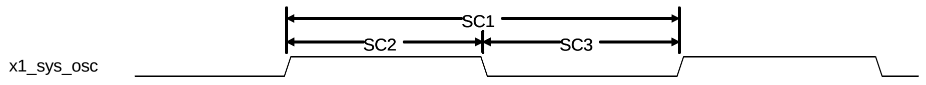


Figure 148: System PLL bypass mode timing

Packaging

C H A P T E R 18

The NS9750 is a complete system-on-chip processor, and includes Ethernet, display support, and a robust peripheral set.

NS9750 dimensions and pinout are shown on the following pages.

Figure 149 displays the top view and dimensions of the NS9750. Figure 150 displays the side and bottom views and dimensions.

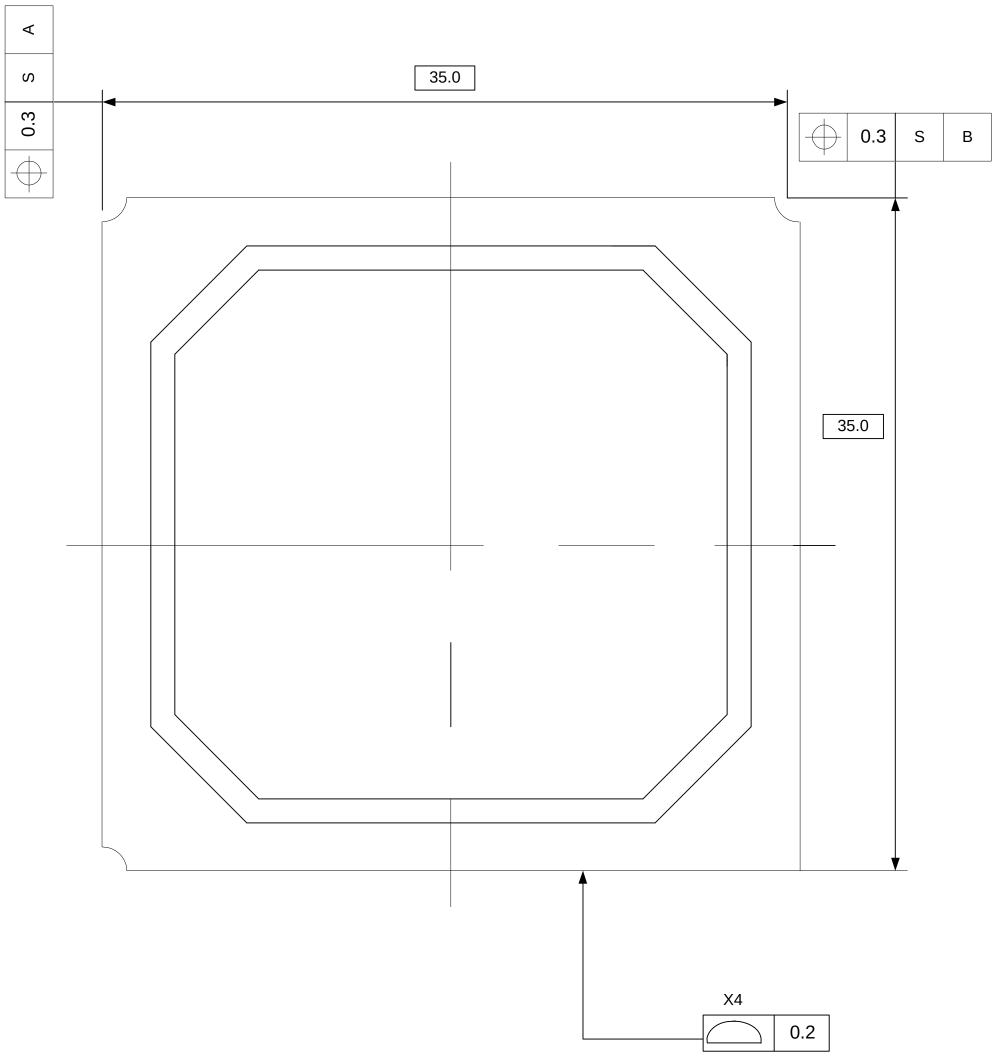


Figure 149: NS9750 top view

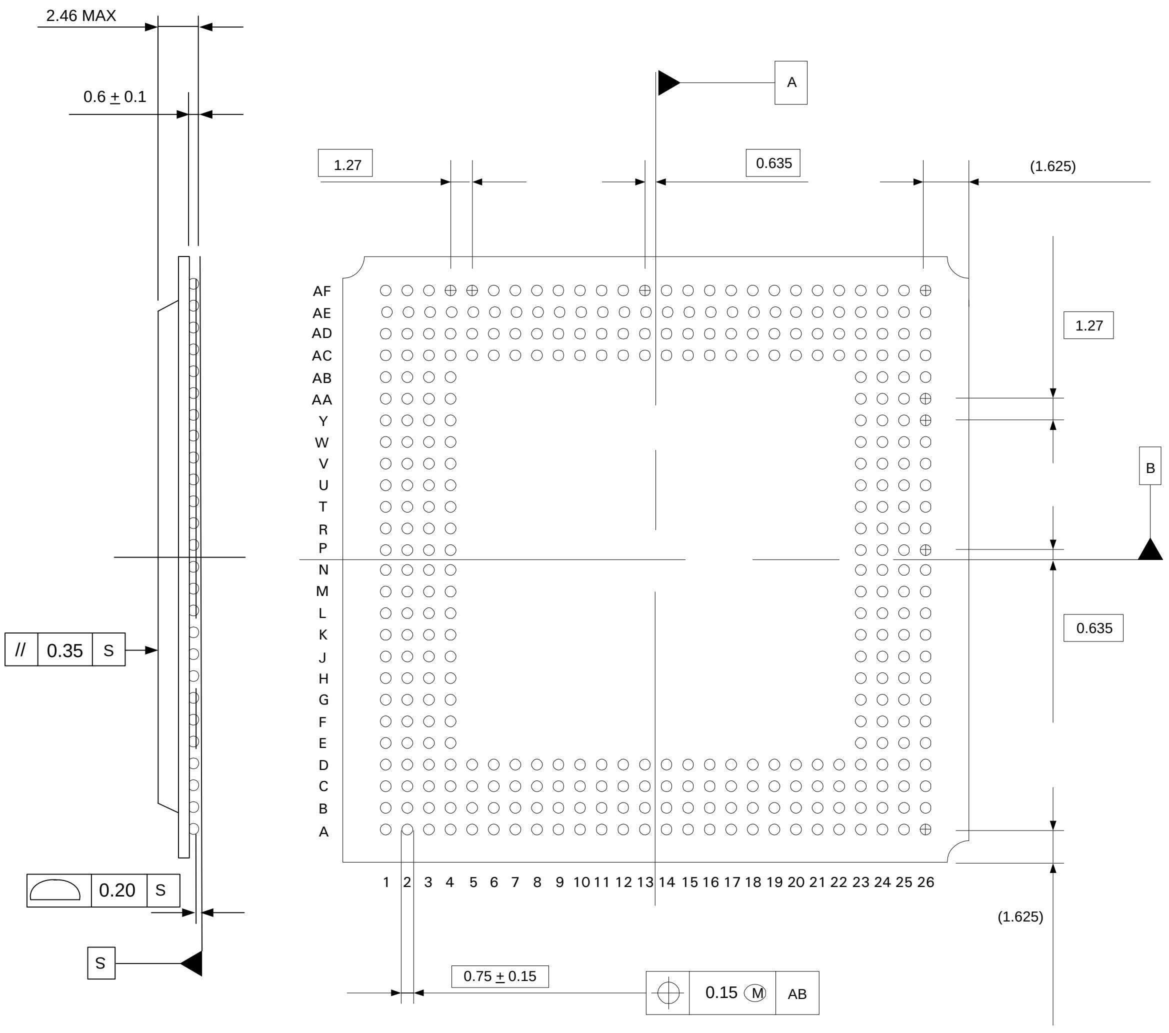


Figure 150: NS9750 side and bottom views

Figure 151 shows the layout of the NS9750, for use in setting up the board.

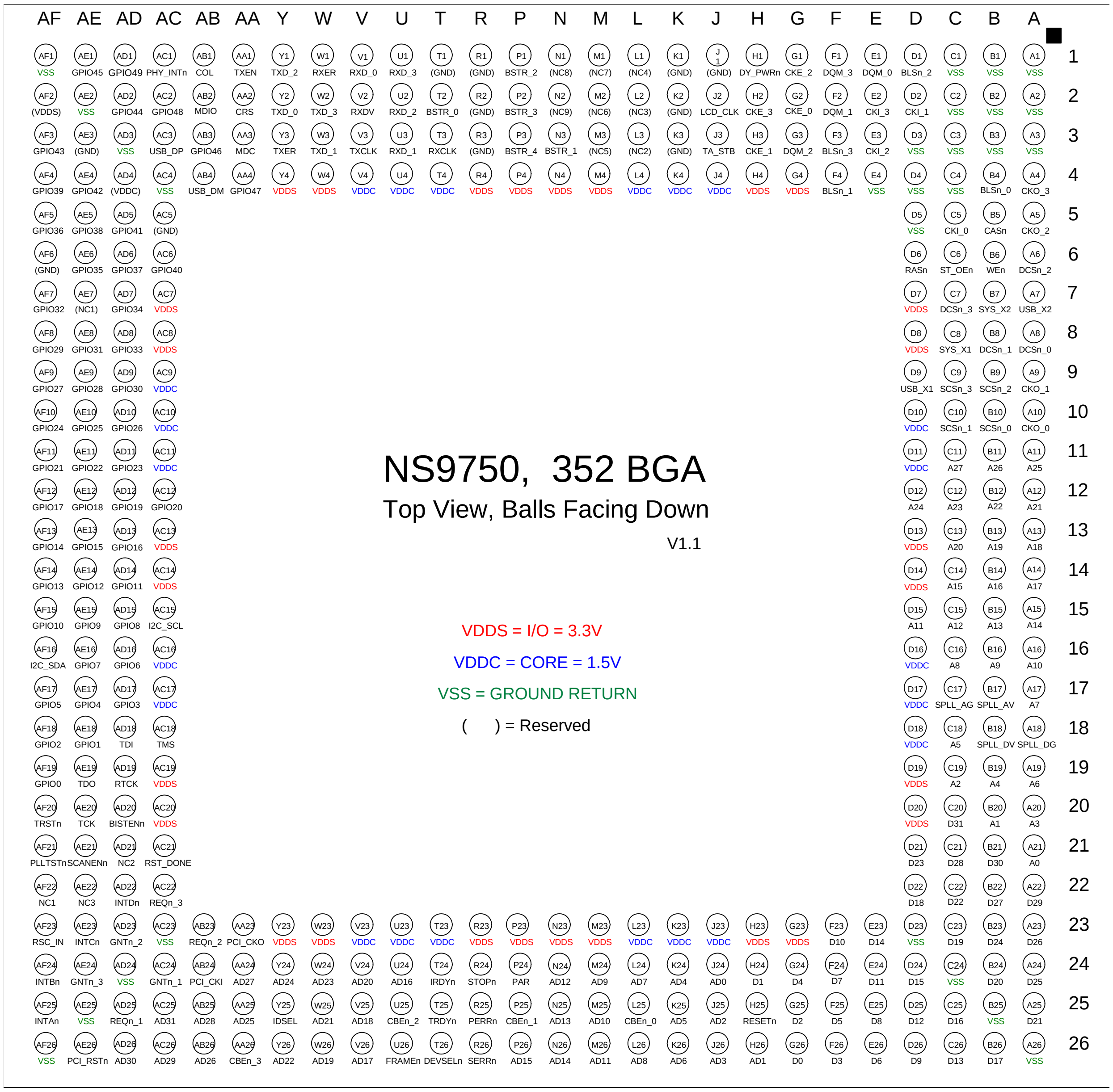

AF AE AD AC AB AA Y W V U T R P N M L K J H G F E D C B A
1: AF1 VSS, AE1 GPIO45, AD1 GPIO49, AC1 PHY_INTn, AB1 COL, AA1 TXEN, Y1 TXD_2, W1 RXER, V1 RXD_0, U1 RXD_3, T1 (GND), R1 (GND), P1 BSTR_2, N1 (NC8), M1 (NC7), L1 (NC4), K1 (GND), J1 (GND), H1 DY_PWRn, G1 CKE_2, F1 DQM_3, E1 DQM_0, D1 BLSn_2, C1 VSS, B1 VSS, A1 VSS
2: AF2 (VDDS), AE2 VSS, AD2 GPIO44, AC2 GPIO48, AB2 MDIO, AA2 CRS, Y2 TXD_0, W2 TXD_3, V2 RXDV, U2 RXD_2, T2 BSTR_0, R2 (GND), P2 BSTR_3, N2 (NC9), M2 (NC6), L2 (NC3), K2 (GND), J2 LCD_CLK, H2 CKE_3, G2 CKE_0, F2 DQM_1, E2 CKI_3, D2 CKI_1, C2 VSS, B2 VSS, A2 VSS
3: AF3 GPIO43, AE3 (GND), AD3 VSS, AC3 USB_DP, AB3 GPIO46, AA3 MDC, Y3 TXER, W3 TXD_1, V3 TXCLK, U3 RXD_1, T3 RXCLK, R3 (GND), P3 BSTR_4, N3 BSTR_1, M3 (NC5), L3 (NC2), K3 (GND), J3 TA_STB, H3 CKE_1, G3 DQM_2, F3 BLSn_3, E3 CKI_2, D3 VSS, C3 VSS, B3 VSS, A3 VSS
4: AF4 GPIO39, AE4 GPIO42, AD4 (VDDC), AC4 VSS, AB4 USB_DM, AA4 GPIO47, Y4 VDDS, W4 VDDS, V4 VDDC, U4 VDDC, T4 VDDC, R4 VDDS, P4 VDDS, N4 VDDS, M4 VDDS, L4 VDDC, K4 VDDC, J4 VDDC, H4 VDDS, G4 VDDS, F4 BLSn_1, E4 VSS, D4 VSS, C4 VSS, B4 BLSn_0, A4 CKO_3
5: AF5 GPIO36, AE5 GPIO38, AD5 GPIO41, AC5 (GND), D5 VSS, C5 CKI_0, B5 CASn, A5 CKO_2
6: AF6 (GND), AE6 GPIO35, AD6 GPIO37, AC6 GPIO40, D6 RASn, C6 ST_OEn, B6 WEn, A6 DCSn_2
7: AF7 GPIO32, AE7 (NC1), AD7 GPIO34, AC7 VDDS, D7 VDDS, C7 DCSn_3, B7 SYS_X2, A7 USB_X2
8: AF8 GPIO29, AE8 GPIO31, AD8 GPIO33, AC8 VDDS, D8 VDDS, C8 SYS_X1, B8 DCSn_1, A8 DCSn_0
9: AF9 GPIO27, AE9 GPIO28, AD9 GPIO30, AC9 VDDC, D9 USB_X1, C9 SCSn_3, B9 SCSn_2, A9 CKO_1
10: AF10 GPIO24, AE10 GPIO25, AD10 GPIO26, AC10 VDDC, D10 VDDC, C10 SCSn_1, B10 SCSn_0, A10 CKO_0
11: AF11 GPIO21, AE11 GPIO22, AD11 GPIO23, AC11 VDDC, D11 VDDC, C11 A27, B11 A26, A11 A25
12: AF12 GPIO17, AE12 GPIO18, AD12 GPIO19, AC12 GPIO20, D12 A24, C12 A23, B12 A22, A12 A21
13: AF13 GPIO14, AE13 GPIO15, AD13 GPIO16, AC13 VDDS, D13 VDDS, C13 A20, B13 A19, A13 A18
14: AF14 GPIO13, AE14 GPIO12, AD14 GPIO11, AC14 VDDS, D14 VDDS, C14 A15, B14 A16, A14 A17
15: AF15 GPIO10, AE15 GPIO9, AD15 GPIO8, AC15 I2C_SCL, D15 A11, C15 A12, B15 A13, A15 A14
16: AF16 I2C_SDA, AE16 GPIO7, AD16 GPIO6, AC16 VDDC, D16 VDDC, C16 A8, B16 A9, A16 A10
17: AF17 GPIO5, AE17 GPIO4, AD17 GPIO3, AC17 VDDC, D17 VDDC, C17 SPLL_AG, B17 SPLL_AV, A17 A7
18: AF18 GPIO2, AE18 GPIO1, AD18 TDI, AC18 TMS, D18 VDDC, C18 A5, B18 SPLL_DV, A18 SPLL_DG
19: AF19 GPIO0, AE19 TDO, AD19 RTCK, AC19 VDDS, D19 VDDS, C19 A2, B19 A4, A19 A6
20: AF20 TRSTn, AE20 TCK, AD20 BISTENn, AC20 VDDS, D20 VDDS, C20 D31, B20 A1, A20 A3
21: AF21 PLLTSTn, AE21 SCANENn, AD21 NC2, AC21 RST_DONE, D21 D23, C21 D28, B21 D30, A21 A0
22: AF22 NC1, AE22 NC3, AD22 INTDn, AC22 REQn_3, D22 D18, C22 D22, B22 D27, A22 D29
23: AF23 RSC_IN, AE23 INTCn, AD23 GNTn_2, AC23 VSS, AB23 REQn_2, AA23 PCI_CKO, Y23 VDDS, W23 VDDS, V23 VDDC, U23 VDDC, T23 VDDC, R23 VDDS, P23 VDDS, N23 VDDS, M23 VDDS, L23 VDDC, K23 VDDC, J23 VDDC, H23 VDDS, G23 VDDS, F23 D10, E23 D14, D23 VSS, C23 D19, B23 D24, A23 D26
24: AF24 INTBn, AE24 GNTn_3, AD24 VSS, AC24 GNTn_1, AB24 PCI_CKI, AA24 AD27, Y24 AD24, W24 AD23, V24 AD20, U24 AD16, T24 IRDYn, R24 STOPn, P24 PAR, N24 AD12, M24 AD9, L24 AD7, K24 AD4, J24 AD0, H24 D1, G24 D4, F24 D7, E24 D11, D24 D15, C24 VSS, B24 D20, A24 D25
25: AF25 INTAn, AE25 VSS, AD25 REQn_1, AC25 AD31, AB25 AD28, AA25 AD25, Y25 IDSEL, W25 AD21, V25 AD18, U25 CBEn_2, T25 TRDYn, R25 PERRn, P25 CBEn_1, N25 AD13, M25 AD10, L25 CBEn_0, K25 AD5, J25 AD2, H25 RESETn, G25 D2, F25 D5, E25 D8, D25 D12, C25 D16, B25 VSS, A25 D21
26: AF26 VSS, AE26 PCI_RSTn, AD26 AD30, AC26 AD29, AB26 AD26, AA26 CBEn_3, Y26 AD22, W26 AD19, V26 AD17, U26 FRAMEn, T26 DEVSELn, R26 SERRn, P26 AD15, N26 AD14, M26 AD11, L26 AD8, K26 AD6, J26 AD3, H26 AD1, G26 D0, F26 D3, E26 D6, D26 D9, C26 D13, B26 D17, A26 VSS
NS9750, 352 BGA
Top View, Balls Facing Down
V1.1
VDDS = I/O = 3.3V
VDDC = CORE = 1.5V
VSS = GROUND RETURN
( ) = Reserved

Figure 151: NS9750 BGA layout

Product specifications

These tables provide additional information about the NS9750.

ROHS substance	PPM level
Lead	0
Mercury	0
Cadmium	0
Hexavalent Chromium	0
Polybrominated biphenyls	0
Polybrominated diphenyl ethers	0

Table 484: NS9750 ROHS specifications

Component	Weight [mg]	Material		Weight [mg]	Weight [%]
		CAS no.	**Name**		
Chip	27.037	7440-21-3	Si	27.0370	0.61
Frame	1841.616	223769-10-6	Epoxy resin	865.5600	19.71
		7440-50-8	Cu	736.6500	16.77
		7440-02-0	Ni	9.2100	0.21
		7440-57-5	Au	1.8400	0.04
			Other	228.3560	5.20
Bonding wire	6.990	7440-57-5	Au	6.9900	0.16
Ag paste	3.400	7440-22-4	Ag	2.6200	0.06
			Epoxy, other	0.7800	0.02
Epoxy resin	1920.177	60676-86-0	Silica (SiO2)	1747.3570	39.79
			Epoxy, other	86.4100	1.97
			Phenol Resin	86.4100	1.97

Table 485: NS9750 materials sheet

Component	Weight [mg]	Material		Weight [mg]	Weight [%]
		CAS no.	**Name**		
Solder ball	592.400	7440-31-5	Sn	571.6700	13.02
		7440-22-4	Ag	17.7700	0.40
		7440-50-8	Cu	2.9600	0.07
Total weight	4391.620				

Table 485: NS9750 materials sheet

Index

Numerics

10/100 Ethernet MAC 3
1284 parallel peripheral port 5
16-bit byte gap counter 689, 690
16-bit maximum buffer counter 689, 690
16-bit Thumb instruction set 48, 50
32-bit ARM instruction set 48, 50
8-bit Java instruction set 50

A

absolute maximum ratings 788
access sequencing and memory width, dynamic memory controller 162
access sequencing and memory width, static memory controller 123
Active Interrupt Level Status register 301
address connectivity 150
address manipulation, ARM926EJ-S 51
address mapping 163–201
address translation, MMU 81–95
AHB address decoding and translation 408
AHB Arbiter Gen Configuration register 282
AHB behavior with noncacheable instruction fetches 112
AHB bus arbiter timeout 257
AHB bus error interrupt 409
AHB input clock, LCD 565
AHB interface, LCD 568–579
  dual DMA FIFOs amd associated control logic 568
  external pad interface signals 575
  generating interrupts 575
  grayscaler 574
  LCD panel signal multiplexing details 575
  master and slave interfaces 568
  panel clock generator 574
  pixel serializer 569
  RAM palette 573
  timing controller 574
  upper and lower panel formatters 574
AHB master interface 405
AHB slave/target interface 406
alignment faults 101
ambient temperatures 7
arbitration and multiplexing, BBus bridge 468
ARM boot configuration (SPI-EEPROM) 486
ARM debug architecture 48
ARM Wake-up register 541
ARM926EJ-S processor 47

about 48
cache format 110
DSP 78
Jazelle (Java) 77
Memory Management Unit. See *MMU.*
system register addresses 51
ARM926EJ-S RISC processor 2
attributes, system control module 271–276

B

Back-to-Back Inter-Packet_Gap register 354
bandwidth requirements 483
BBus
master and slave modules 472
BBus bridge 467–500
arbitration (masters and slaves) 472
arbitration and multiplexing 468
bandwidth requirements 483
BBus Bridge Interrupt Enable register 499
BBus Bridge Interrupt Status register 498
Buffer Descriptor Pointer register 491
Control and Status registers 490–500
control logic 469, 472–473
descriptor list processing 476
DMA accesses 471
DMA buffer descriptor 474
DMA Channel 1/2 Control register 491
DMA memory-to-peripheral transfers 471
DMA Peripheral Chip Select register 496
DMA peripheral-to-memory transfers 471
DMA Status and Interrupt Enable register 494
functions 468
interrupt aggregation 483
main function 467
peripheral address map 473
register addresses 490
SDRAM boot algorithm 488–489
SPI-EEPROM boot logic 484–489
SPI-EEPROM boot logic, memory controller configuration 486
system boot engine 468
two-channel DMA controller 468, 474–483
BBus bridge arbitration
cycles 473
masters and slaves 472
BBus Bridge Control and Status registers 490–500
BBus Bridge Interrupt Enable register 499
BBus Bridge Interrupt Status register 498
BBus DMA controller 501–519
about the controllers 502
controller block 503
DMA buffer descriptor 504–508
DMA Buffer Descriptor Pointer register 512
DMA channel assignments 509–510
DMA context memory 503
DMA Control register 514
DMA Status/Interrupt Enable register 516
DMA transfer status 506–508
fly-by DMA transfers 502
fly-by mode 502
register addresses 511
BBus DMA Interrupt Enable register 537
BBus DMA Interrupt Status register 536
BBus Monitor register 535

BBus peripheral address map 473
BBus slave and DMA interface module 677–705
  CPU mode 677
  DMA mode 677
  register map 677
BBus subsystem 501
BBus utility 521–542
  ARM Wake-up register 541
  BBus DMA Interrupt Enable register 537
  BBus DMA Interrupt Status register 536
  BBus Monitor register 535
  Endian Configuration register 539
  GPIO Configuration register options 528
  GPIO Configuration registers 524–529
  GPIO Control registers 529–532
  GPIO Status registers 532–535
  Master Reset register 523
  register addresses 522
  USB Configuration register 538
BGA layout 844
bist_en_n truth/termination table 28
bit-rate examples 628–629
bit-rate generator 603, 645
  clock sources 604, 645
bootstrap initialization 272–276
  configuration pins 273
BRC channel assignment 283
BRC0, BRC1, BRC2, BRC3 registers 283
BRF field 256
Bridge Configuration registers 413–418
Buffer Descriptor Pointer register 491
Buffer Full Status register 703
buffer length 475, 505
buffer length bit, Ethernet 327
buffer length, Ethernet 328
buffer pointer, Ethernet 326, 328
bus arbitration
  I2C 547
bus interconnection, system control module 254
bus turnaround 143–148
byte lane control 149
byte lane control and databus steering 162
  little and big endian, address not right-justified 161
  little and big endian, address right-justified 154
  byte mode 672
  data transfer cycle 672

C

caches and write buffer 105–111
  cache features 105
  cache MVA and set/way formats 109
  enabling the caches 107
  write buffer 106
CAM controller 334
CardBus adapter requirements 464
CardBus Miscellaneous Support register 442
CardBus port 3
CardBus signals 463
CardBus Socket Control register 454
CardBus Socket Event register 446
CardBus Socket Force Event register 451
CardBus Socket Mask register 447
CardBus Socket Present State register 448
CardBus support 461–465
Carry Register 1 378
Carry Register 1 Mask register 380
Carry Register 2 379

Carry Register 2 Mask register 382
Centronics mode. See *compatibility mode.</Emphasis>*
chip select 1 memory configuration 119
CLD signal 42
Clock Configuration register 293
clock frequency settings 13
clock generation/system pins 26
clock generator 7
clock timing 837–839
clocks, LCD 565
coarse page tables 84, 87, 90, 92
Collision Window/Retry register 355
combined transmit and receive statistics counters 368
Command Transmit Data register 548
compatibility mode 671
  data transfer cycle 671
Configuration Address Data Port register 411
Configuration Address Port register 411
Configuration register 552
Configuration register, memory 207
configuring the NS9750 for CardBus support 463
context (static) RAM 502
control in packets 711
control logic, BBus bridge 469, 472–473
Control register, memory 205
conventions, documentation xxi
Core Phase (IEEE 1284) register 704
CPU 47–114
  about the ARM926EJ-S processor 48
  accessing the CP15 registers 52
  caches and write buffer 105–111
  core 47
  DSP 78
  Jazelle (Java) 77
MMU 78–105
  access permissions and domains 79
  address translation 81–95
  coarse page table descriptor 87
  disabling 104
  enabling 103
  external aborts 102
  fault checking sequence 99
  features 78
  fine page table descriptor 88
  first-level descriptor 84
  first-level fetch 84
  MMU faults and aborts 95–98
  program accessible registers 80
  second-level descriptor 89
  section descriptor 86
  subpages 95
  TLB structure 104
  translated entries 79
  translating parge page references 91
  translating section references 89
  translating small page references 93
  translating tiny page references 94
  translation table base 82
noncacheable instruction fetches 111
R0, ID code and cache type status registers 55–57
R1, Control register 58–60
R10, TLB Lockdown register 73
R11 register 74
R12 register 74
R13, Process ID register 75–77
R14 register 77
R15, Test and debug register 77
R2, Translation Table Base register 61

R3, Domain Access Control register 61, 98
R4 register 62
R5, Fault Status registers 62
R6, Fault Address register 64, 97
R7, Cache Operations register 64–68
R8, TLB Operations register 68
R9, Cache Lockdown register 69–73
system control processor registers 51–77
addresses 51
summary 53
terms and abbreviations 52
CPU arbiter, high speed 255
CPU bus arbitration 472
CPU subsystem 255
cross-bridge transaction error handling 407

D

data and command FIFOs 675
data structures in LCD panels 570–572
DC electrical characteristics 790–791
inputs 790
outputs 791
DC electrical inputs, USB 790
DC electrical outputs, USB 791
DCache, data cache 56, 67, 68, 105, 106, 107, 108
write buffer 106
debug mode (GPTC) 264
decoding. See *BBus peripheral address map.*
destination address 475, 505
device block, USB 708, 709, 710–712
architecture 710
control and status 710
logical and physical endpoints 712
packet and data flow 711
Device Control and Status register 718
Device IP Programming Control/Status register 724
device selection for configuration, PCI 458
DMA
BBus controller. *See BBus DMA controller.*
descriptor list processing 476
DMA accesses, BBus bridge 471
memory-to-peripheral transfers 471
peripheral-to-memory transfers 471
DMA buffer descriptor 474, 504–508
buffer length 505
definition 505
destination address 505
DMA transfer status 506–508
FULL bit 506
illustration 505
interrupt bit 506
LAST bit 506
source address 505
status field 506
WRAP bit 505
DMA Buffer Descriptor Pointer register 512
DMA Channel 1/2 Control register 491
DMA channel assignments 509–510
DMA context memory 503
DMA Control register 514
DMA controller
two-channel 468, 474–483
initiating a DMA transfer 474
DMA memory-to-peripheral transfers 471
DMA Peripheral Chip Select register 496
DMA peripheral-to-memory transfers 471

DMA read. See also *DMA memory-to-peripheral transfers.* 471
DMA Status and Interrupt Enable register 494
DMA Status/Interrupt Enable register 516
DMA system 5
DMA transfer
 executing 474
 two-channel transfer
  external-peripheral-initiated 474
  processor-initated 474
DMA transfer status 506–508
DMA write. See also *DMA peripheral-to-memory transfers.* 471
documentation
 conventions xxi
domain faults 101
downstream port (USB) 708
downstream transactions 405
DSP 78
dummy default master 257
dynamc memory controller
 address mapping 163–201
Dynamic Memory Active Bank A to Active Bank B Time register 222
Dynamic Memory Active to Active Command Period register 219
Dynamic Memory Active to Precharge Command Period register 214
Dynamic Memory Auto refresh Period register 220
Dynamic Memory Configuration 0-3 registers 225
 address mapping 226
Dynamic Memory Control register 208
dynamic memory controller 162–201
 write protection 162
dynamic memory controller, access sequencing and memory width 162
Dynamic Memory Data-in to Active Command Time register 217
Dynamic Memory Exit Self-refresh register 221
Dynamic Memory Last Data Out to Active Time register 216
Dynamic Memory Load Mode register to Active Command Time register 223
Dynamic Memory Precharge Command Period register 213
Dynamic Memory RAS and CAS Delay 0-3 registers 229
Dynamic Memory Read Configuration register 212
Dynamic Memory Refresh Timer register 210
Dynamic Memory Self-refresh Exit Time register 215
Dynamic Memory Write Recovery Time register 218

E

ECP mode 673–674
 forward transfer cycles 673
 host processing sequence example 674
 reverse transfer cycles 674
electrical characteristics 788–789
electrical specifications. See *Timing*.
ENABLE bit, Ethernet 326
enable/disable interrupts for BBus DMA 537
Endian configuration 411
Endian Configuration register 539
Endpoint Descriptor 0-11 registers 766
Ethernet communication module 315–401
 Back-to-Back Inter-Packet-Gap

register 354
Collision Window/Retry register 355
control and status registers 337–396
definition 315
diagram 316
Ethernet front-end (EFE) module. See also *Ethernet front-end module.* 323–333
Ethernet front-end features 316
Ethernet front-end module 315
Ethernet General Control Register #1 339
Ethernet General Control Register #2 342
Ethernet General Status register 344
Ethernet Interrupt Enable register 387
Ethernet Interrupt Status register 385
Ethernet MAC 317–322
Ethernet MAC features 316
Ethernet Receive Status register 347
Ethernet timing diagrams 813–815
Ethernet Transmit Status register 344
external CAM filtering 334–336
General Statistics registers 377
  Carry Register 1 378
  Carry Register 1 Mask register 380
  Carry Register 2 379
  Carry Register 2 Mask register 382
MAC Configuration Register #1 348
MAC Configuration Register #2 351
Maximum Frame register 357
MII Management Address register 361
MII Management Command register 360
MII Management Configuration register 359
MII Management Indicators register 363
MII Management Read Data register 363
MII Management Write Data register 362
Non Back-to-Back Inter-Packet Gap register 355
PHY Support register 358
register addresses 337
register hash tables 366
RX A Buffer Descriptor Pointer Offset register 391
RX A Buffer Descriptor Pointer register 383
RX B Buffer Descriptor Offset register 392
RX B Buffer Descriptor Pointer register 383
RX C Buffer Descriptor Pointer Offset register 393
RX C Buffer Descriptor Pointer register 384
RX D Buffer Descriptor Pointer Offset register 393
RX D Buffer Descriptor Pointer register 384
RX Free Buffer register 395
sample hash table code 397–401
Station Address Filter register 366
Station Address registers 364
Statistics registers 368
  combined transmit and receive statistics counters 368
  receive statistics counters 369
  transmit statistics counters 373
Transmit Buffer Descriptor Pointer Offset register 394
Transmit Recover Buffer Descriptor Pointer register 389
TX Buffer Descriptor Pointer register 389
TX buffer descriptor RAM 396

TX Error Buffer Descriptor Pointer register 390
Ethernet front-end module
about 323
Ethernet slave interface 330
features 316
interrupts 331
power down mode 324
receive packet processor 324
resets 332
transferring a frame to system memory 325
transmit packet processor 327
transmitting a frame to the Ethernet MAC 330
Ethernet General Control Register #1 339
Ethernet General Control Register #2 342
Ethernet General Status register 344
Ethernet interface pinout 25
Ethernet Interrupt Enable register 387
Ethernet Interrupt Status register 385
Ethernet MAC 315, 317–322
about 317
external clocks 14
features 316, 318
PHY interface mappings 320
station address logic (SAL) 321
statistics module 321
Ethernet Media Access Controller. See *Ethernet MAC*.
Ethernet Receive Status register 347
Ethernet slave interface 330
Ethernet timing 813–815
Ethernet Transmit Status register 344
Extended Control register 698
extended wait transfers 122
Extensibility Byte Requested by Host retgister 698
extensibility byte values 676
external aborts 102
external CAM filtering 334–336
CAM controller 334
External Interrupt 0-3 Control register 313
external interrupts 7
external pad interface signals 575
external peripheral 5
external system bus interface 2
external-peripheral-initiated DMA transfer (AHB) 474

F

fast interrupts. See *FIQ interrupts.*
Feature Control Register A 694
Feature Control Register B 695
FIFO Interrupt Enable registers 776–780
FIFO Interrupt Enable 0 register 776
FIFO Interrupt Enable 1 register 777
FIFO Interrupt Enable 2 register 778
FIFO Interrupt Enable 3 register 779
FIFO Interrupt Status registers 769–775
FIFO Interrupt Status 0 register 771
FIFO Interrupt Status 1 register 771
FIFO Interrupt Status 2 register 773
FIFO Interrupt Status 3 register 775
FIFO management 605–608, 647–650
FIFO Packet Control registers 780
FIFO Status and Control registers 781
FIFO Status register 684
filtering options, station address logic 321
fine page tables 85, 88, 90, 93
FIQ interrupts 267
flash boot 10
fly-by DMA transfers 502
fly-by memory-to-peripheral operations 505

fly-by mode 502
fly-by peripheral to memory operations 505
Forward Address register 703
Forward Command DMA Control register 689
Forward Command FIFO Read register 686
Forward Data DMA Control register 690
Forward Data FIFO Read register 687
FULL bit 476, 506
FULL bit, Ethernet 326, 328

G

gated timer 263
GEN ID register 311
general purpose I/O. See also *GPIO*.
general purpose timers and counters 6
general purpose timers/counters 263
General Statistics registers (Ethernet) 377
  Carry Register 1 378
  Carry Register 1 Mask register 380
  Carry Register 2 379
  Carry Register 2 Mask register 382
generating interrupts, LCD 575
Global Control and Status register 717
Global Interrupt Enable register 720
Global Interrupt Status register 721
GPIO Configuration register options 528
GPIO Configuration registers 524–529
GPIO Control registers 529–532
GPIO MUX pinout 34
GPIO Status registers 532–535
GPTC. See *general purpose timers/counters*.
Granularity Count register 702
grayscaler 574

H

handling USB-IN packet errors 715
handling USB-OUT packet errors 715
Harvard-cached architecture 48
HcBulkCurrentED register 744
HcBulkHeadED register 743
HcCommandStatus register 730
HcControl register 727
HcControlCurrentED register 742
HcControlHeadED register 741
HcDoneHead register 746
HcFmInterval register 747
HcFmNumber register 749
HcFmRemaining register 748
HcHCCA register 739
HcInterruptDisable register 737
HcInterruptEnable register 735
HcInterruptStatus register 733
HcLsThreshold register 751
HcPeriodCurrentED register 740
HcPeriodicStart register 750
HCRevision register 726
HcRhDescriptorA register 753
HcRhDescriptorB register 755
HcRhPortStatus1 register 759
HcRhStatus register 756
high-speed bus architecture 255
high-speed peripheral subsystem 255
horizontal axis panel 580
horizontal timing restrictions, LCDTiming0 581
host block, USB 708, 710, 712–714
  control and status 712
  packet data flow 713
host controller communication area 739
hsel_x signal 261

I

I bit, Ethernet 326, 328
I2C
    bus arbitration 547
    command interface 545
    Command Transmit Data register 548
    Configuration register 552
    external addresses 545
    locked interrupt driven mode 546
    Master Address register 550
    master module commands 546
    master module flowchart 556
    master software driver 555
    master/slave interface 543–557
    module 544
    physical bus 544
    register addresses 547
    Slave Address register 551
    slave module commands 546
    slave module flowchart 557
    Status Receive Data register 549
    timing diagram 821
I2C controller
    port features 5
I2C pinout 43
I2C timing 821
ICache, instruction cache 51, 56, 105, 107
IDLE condition 605
IEEE 1284
    BBus slave and DMA interface module 677–705
    Buffer Full Status register 703
    byte mode 672
    compatibility mode 671
    Core Phase (IEEE 1284) register 704
    data and command FIFOs 675
    ECP mode 673–674
    Extended Control register 698
    Extensibility Byte Requested by Host register 698
    extensibility byte values 676
    Feature Control Register A 694
    Feature Control Register B 695
    FIFO Status register 684
    Forward Address register 703
    Forward Command DMA Control register 689
    Forward Command FIFO Read register 686
    Forward Data DMA Control register 690
    Forward Data FIFO Read register 687
    General Configuration register 679
    Granularity Count register 702
    Interrupt Enable register 695
    Interrupt Status and Control register 681
    Interrupt Status register 699
    Interrupt Timeout Counter register 700
    Master Enable register 697
    negotiation 676
    nibble mode 672
    peripheral controller 669–705
    peripheral port control module 670
    Pin Interrupt Control register 701
    Pin Interrupt Mask register 700
    Port Control register 693
    Port Status register, host 692
    Port Status register, peripheral 694
    Printer Data Pins register 691
    register map (BBus slave and DMA interface) 677
    requirements 670
    Reverse FIFO Write register 687
    Reverse FIFO Write Register - Last 687

timing diagram 831
IEEE 1284 General Configuration register 679
IEEE 1284 timing 831
individual interrupts, LCD 566
industry-standard Ethernet interfaces
MII 315
RMII 315
initializing PLL parameters 14
initiating a DMA transfer, two channel (AHB) 474
Int Config registers (0-31) 286
register address mapping 286
interrupt aggregation 483
INTERRUPT bit 476
interrupt bit 506
interrupt codes 553
Interrupt Configuration register 267
interrupt controller 267–270
FIQ interrupts 267
interrupt sources 268, 269
IRQ interrupts 267
vector interrupt controller 270
Interrupt Enable register 695
interrupt sources 268, 269
Interrupt Status Active register 289
Interrupt Status and Control register 681
interrupt status bits for DMA 536
Interrupt Status Raw register 290
Interrupt Status register 699
Interrupt Timeout Counter register 700
Interrupt Vector Address register 267
Interrupt Vector Address Register Level 0-31 285
interrupts
CardBus 465
Ethernet 331
PCI 409, 458
interrupts, LCD 598–599
master bus error interrupt 598
next base address update interrupt 599
vertical compare interrupt 598
IRQ interrupts 267
IRSADDR register 288

J

Jazelle (Java) 77
JTAG interface
pinout for ARM core/boundary scan 43
JTAG timing 836

L

L bit, Ethernet 328
LAST bit 476, 506
LCD controller 42, 559–599
about 559
AHB interface 568–579
dual DMA FIFOS and associated control logic 568
external pad interface signals 575
generating interrupts 575
grayscaler 574
LCD panel signal multiplexing details 575
master and slave interfaces 568
panel clock generator 574
pixel serializer 569
RAM palette 573
timing controller 574
upper and lower panel formatters 574
clocks 565

color STN panels 562
features 4, 560–564
functional overview 564–567
interrupts 598–599
master bus error interrupt 598
next base update interrupt 599
vertical compare interrupt 598
LCD panel resolution 561
LCD timing diagrams 822–826
LCDControl register 590
LCDInterrupt register 594
LCDINTRENABLE register 589
LCDLPBASE register 587
LCDLPCURR register 594
LCDPalette register 595
LCDStatus register 593
LCDTiming0 register 580
LCDTiming1 register 582
LCDTiming2 register 583
LCDTiming3 register 587
LCDUPBASE register 587
LCDUPCURR register 594
mono STN panels 563
number of colors 562
panel support 561
power up/power down sequence 563
programmable parameters 560
register addresses 579
registers 579–597
signals and interrupts 566
individual interrupts 566
programmable control signals 566
single combined interrupt 566
supported display types 564
TFT panels 562
LCD input clock 565
LCD module signals 42
LCD panel resolution 561
LCD panel signal multiplexing details 575
LCD panel support 561
LCD timing 822–826
LCDControl register 590
LCDInterrupt register 594
LCDINTRENABLE register 589
LCDLPBASE register 587
LCDLPCURR register 594
LCDPalette register 595
big endian configuration 596
little endian configuration 596
LCDStatus register 593
LCDTiming0 register 580
horizontral timing restriction 581
LCDTiming1 register 582
LCDTiming2 register 583
LCDTiming3 register 587
LCDUPBASE register 587
LCDUPCURR register 594
locked bus sequence 257
locked interrupt driven mode 546
low-power operation 118
low-power SDRAM partial array refresh 118

M

MAC Configuration Register #1 348
MAC Configuration Register #2 351
main arbiter, high speed 256
Master Address register 550
master bus error interrupt (MBERRORINTR) 598
Master Enable register 697
master module commands, I2C 546
Master Reset register 523
master reset. See *reset function*.
Maximum Frame register 357

maximum power dissipation 789
Media Independent Interface. See *MII*.
memory controller 115–240
  access sequencing and memory width, dynamic memory controller 162
  access sequencing and memory width, static memory controller 123
  address connectivity 150
  address mapping, dynamic memory controller 163–201
  bootstrap signals 119
  bootstrap signals, memory controller 119
  bus turnaround 143–148
  byte lane control 149
  byte lane control and databus steering 162
  Configuration register 207
  configuration, SPI-EEPROM 486
  Control register 205
  Dynamic Memory Active Bank A to Active Bank B Time register 222
  Dynamic Memory Active to Active Command Period register 219
  Dynamic Memory Active to Precharge Command Period register 214
  Dynamic Memory Auto Refresh Period register 220
  Dynamic Memory Configuration 0-3 registers 225
    address mapping 226
  Dynamic Memory Control register 208
  dynamic memory controller 162–201
  Dynamic Memory Data-in to Active Command Time register 217
  Dynamic Memory Exit Self-refresh register 221
  Dynamic Memory Last Data Out to Active Time register 216
  Dynamic Memory Load Mode register to Active Command Time register 223
  Dynamic Memory Precharge Command Period register 213
  Dynamic Memory RAS and CAS Delay 0-3 registers 229
  Dynamic Memory Read Configuration register 212
  Dynamic Memory Refresh Timer register 210
  Dynamic Memory Self-refresh Exit Timer register 215
  Dynamic Memory Write Recovery Time register 218
  extended wait transfers 122
  features 116
  low-power operation 118
  memory map 118
  memory mapped peripherals 123
  memory timing diagrams 795–812
  register addresses 202
  registers 202–239
  Static Memory Configuration 0-3 registers 230
  static memory controller 121–162
  static memory controller configurations 121
  static memory controller initialization 123–148
  Static Memory Extended Wait register 224
  Static Memory Output Enable Delay 0-3 registers 235
  Static Memory Page Mode Read Delay 0-3 registers 237
  static memory read control 124–136
  Static Memory Read Delay 0-3 registers 236

Static Memory Turn Round Delay 0-3 registers 239
static memory write control 136–143
Static Memory Write Delay 0-3 registers 238
Static Memory Write Enable Delay 0-3 registers 234
Status register 207
system overview 117
write protection
dynamic memory controller 162
static memory controller 122
Memory Management Unit. See *MMU*.
memory map 118
memory mapped peripherals 123
memory timing 795–812
MII 315
MII Management Address register 361
MII Management Command register 360
MII Management Configuration register 359
clocks field settings 360
MII Management Indicators register 363
MII Management Read Data register 363
MII Management Write Data register 362
Miscellaneous System Configuration and Status register 296
MMU 78–105
access permissions and domains 79
address translation 81–95
coarse page table descriptor 87
disabling 104
domain access control 98
enabling 103
external aborts 102
fault checking sequence 99
alignment faults 101
domain faults 101
permission faults 102
translation faults 101
features 78
fine page table descriptor 88
first-level descriptor 84
first-level fetch 84
MMU faults and aborts 95–98
program accessible registers 80
second-level descriptor 89
section descriptor 86
subpages 95
TLB structure 104
translated entries 79
translating large page references 91
translating section references 89
translating small page references 93
translating tiny page references 94
translation table base 82

N

negotiation, IEEE 1284 676
next base address update interrupt (LNBU) 599
nibble mode 672
data transfer cycle 672
Non Back-to-Back Inter-Packet-Gap register 355
noncacheable instruction fetches 111
AHB behavior 112
NS9750 17
10/100 Ethernet MAC 3
1284 parallel peripheral port 5
clock generation/system pins pinout 26
clock generator 7
configuring for CardBus support 463
definition 1, 2

DMA 5
Ethernet interface pinout 25
external interrupts 7
external peripheral 5
external system bus interface 2
features 2–7
general purpose I/O (GPIO) 6
general purpose timers and counters 6
GPIO MUX 34
I2C pinout 43
I2C port 5
JTAG interface pinout 43
LCD controller 4
LCD module signals 42
operating grades and ambient temperatures 7
PCI CardBus port 3
PCI pinout 28
PCI/CardBus signals 31
peripheral bus 5
pinout 18–46
power ground 46
power management 6
power sequencing 794
processor 2
reserved pins 45
reset function 10–13
serial ports 4
software resets 12
system boot 3
system bus 5
system clock 13
system memory interface pinout 18
system memory interface signals 22
system timers 6
system-level interfaces 8
USB clock 15
USB interface pinout 43
USB ports 4
vector interrupt controller 6

O

operating grades 7

P

packaging 841–846
  BGA layout 844
  dimensions 842–843
PAD operation table 353
panel clock generator 574
PCI
  CardBus port 3
  interface pinout 28
  PCI/CardBus signals 31
PCI address decoding and mapping 408
PCI Arbiter Configuration register 423
PCI Arbiter Interrupt Enable register 425
PCI Arbiter Interrupt Status register 424
PCI arbiter. See *PCI bus arbiter*.
PCI Base Address registers 417
PCI BIST register 417
PCI Bridge Address Translation Control register 441
PCI Bridge AHB Error Address register 433
PCI Bridge AHB to PCI Memory Address Translate 0 register 437
PCI Bridge AHB to PCI Memory Address Translate 1 register 438
PCI Bridge AHB-to-PCI IO Address Translate register 439
PCI Bridge Configuration register 432
PCI Bridge Interrupt Enable register 435
PCI Bridge Interrupt Status register 434

PCI Bridge PCI Error Address register 433
PCI Bridge PCI to AHB Memory Address Translate 0 register 439
PCI Bridge PCI to AHB Memory Address Translate 1 register 440
PCI bus arbiter 407, 418–455
  about 418
  configuration registers 420–455
  functional description 419
  register addresses 420
  slave interface 420
PCI bus error interrupt 409
PCI Cache Size register 416
PCI CardBus CIS Pointer register 417
PCI central resource functions 458
PCI Class Code register 416
PCI Command register 414
PCI Configuration 0 register 428
PCI Configuration 1 register 429
PCI Configuration 2 register 430
PCI Configuration 3 register 431
PCI Device ID register 414
PCI Expansion ROM register 418
PCI Header register 416
PCI Interrupt Line register 418
PCI Interrupt Pin register 418
PCI Latency Timer register 416
PCI master interface 406
PCI Max Latensy register 418
PCI Min Grant register 418
PCI Miscellaneous Support register 426
PCI Revision ID register 416
PCI Status register 415
PCI Subsystem ID register 418
PCI Subsystem Vendor ID register 417
PCI system configurations 456–461
  central resource functions 458
  device selection 458
  interrupts 458
PCI target error filtering 407
PCI target interface 406
PCI timing 816–820
PCI Vendor ID register 414
PCI-to-AHB bridge 403–465
  AHB address decoding and translation 408
  AHB master interface 405
  AHB slave/target interface 406
  Bridge Configuration registers 413–418
  CardBus adapter requirements 464
  CardBus Miscellaneous Support register 442
  CardBus Socket Control register 454
  CardBus Socket Event register 446
  CardBus Socket Force Event register 451
  CardBus Socket Mask register 447
  CardBus Socket Present State register 448
  CardBus support 461–465
  central resource functions 458
  Configuration Address Data Port register 411
  Configuration Address Port register 411
  configuration and status registers 406
  configuration registers 411
  configuring the NS9750 for CardBus support 463
  definition 403
  device selection for configuration 458
  Endian configuration 411
  features 404
  functionality 405–407
  interrupts 409, 458
  PCI address decoding and mapping 408

PCI Arbiter Configuration register 423
PCI Arbiter Interrupt Enable register 425
PCI Arbiter Interrupt Status register 424
PCI Base Address registers 417
PCI BIST register 417
PCI Bridge Address Translation Control register 441
PCI Bridge AHB Error Address register 433
PCI Bridge AHB to PCI Memory Address Translate 0 register 437
PCI Bridge AHB to PCI Memory Address Translate 1 register 438
PCI Bridge AHB-to-PCI IO Address Translate register 439
PCI Bridge Configuration register 432
PCI Bridge Interrupt Enable register 435
PCI Bridge Interrupt Status register 434
PCI Bridge PCI Error Address register 433
PCI Bridge PCI to AHB Memory Address Translate 0 register 439
PCI Bridge PCI to AHB Memory Address Translate 1 register 440
PCI bus arbiter 407, 418–455
  configuration registers 420–455
  register addresses 420
  slave interface 420
PCI Cache Size register 416
PCI CardBus CIS Pointer register 417
PCI Class Code register 416
PCI Command register 414
PCI Configuration 0 register 428
PCI Configuration 1 register 429
PCI Configuration 2 register 430
  PCI Configuration 3 register 431
  PCI Device ID register 414
  PCI Expansion ROM register 418
  PCI Header register 416
  PCI Interrupt Line register 418
  PCI Interrupt Pin register 418
  PCI Latency Timer register 416
  PCI Max Latency register 418
  PCI Min Grant register 418
  PCI Miscellaneous Support register 426
  PCI Revision ID register 416
  PCI Status register 415
  PCI Subsystem register 418
  PCI Subsystem Vendor ID register 417
  PCI target error filtering 407
  PCI target interface 406
  PCI timing diagrams 816–820
  PCI Vendor ID register 414
  register addresses, PCI/bridge configuration 413
  system configurations 456–461
  transaction ordering 410
PCI-to-AHB bus
  CardBus interrupts 465
  PCI master interface 406
PCI-to-AHB-bridge
  PCI bus arbiter
    functional description 419
peripheral bus 5
peripheral DMA read access 477
peripheral DMA write access 478
peripheral REQ and DONE signalling 479
permission faults 102
PHY Support register 358
Pin Interrupt Control register 701
Pin Interrupt Mask register 700
pinout
  clock generation/system pins 26

Ethernet interface 25
GPIO MUX 34
I2C 43
JTAG interface 43
NS9750 18–46
PCI interface 28
reserved pins 45
system memory interface 18
USB interface 43
pixel serializer 569
PLL configuration 271
PLL Configuration register 299
PLL multiplier values 274
pll_test_n truth/termination table 28
Port Control register 693
Port Status register, host 692
Port Status register, peripheral 694
power down mode, Ethernet 324
power ground 46
power management 6
power sequencing 794
power up/power down sequence, LCD 563
power-on reset memory map 118
Printer Data Pins register 691
priority encoder block 268
privileged access mode 262
processor-initiated DMA transfer (AHB) 474
programmable control signals, LCD 566
programmable parameters, LCD 560
programmable timers 263–264
general purpose timers/counters 263
software watchdog timer 263

R

R0, ID code and cache type status registers 55–57
R1, Control register 58–60
R10, TLB Lockdown register 73
R11 register 74
R12 register 74
R13, Process ID register 75–77
R14 register 77
R15, Test and debug register 77
R2, Translation Table Base register 61
R3, Domain Access Control register 61, 98
R4 register 62
R5, Fault Status registers 62
R6, Fault Address register 64, 97
R7, Cache Operations register 64–68
R8, TLB Operations register 68
R9, Cache Lockdown register 69–73
RAM palette 573
raw interrupt status 593
receive alignment error counter 371
receive broadcast packet counter 370
receive buffer descriptor format
Ethernet 326
receive byte counter 369
receive carrier sense error counter 372
receive code error counter 371
receive control frame packet counter 371
receive FCS error counter 370
receive FIFO interface 606, 648
receive fragments counter 372
receive jabber counter 372
receive multicast packet counter 370
receive oversize packet counter 372
receive packet counter 370
receive packet processor 323, 324
receive PAUSE frame packet counter 371
receive statistics counters (Ethernet) 369
receive undersize packet counter 372
receive unknown OPCODE packet counter 371

recommended operating conditions 788
Reduced Media Independent Interface. See *RMII*.
register hash tables 366
regular timer 263
relinquishing the bus 257
reserved pins 45
reset and edge sensitive input timing requirements 792
Reset and hardware strapping timing 835
Reset and Sleep Control register 295
reset function
  boot sequence 11
  master reset 10
  RESET DONE as input 11
  RESET DONE as output 11
reset values 204
resets, Ethernet 332
Reverse FIFO Write register 687
Reverse FIFO Write Register - Last 687
RMII 315
root hub, USB 752
RX A Buffer Descriptor Pointer Offset register 391
RX A Buffer Descriptor Pointer register 383
RX B Buffer Descriptor Pointer Offset register 392
RX B Buffer Descriptor Pointer retgister 383
RX C Buffer Descriptor Pointer Offset register 393
RX C Buffer Descriptor Pointer register 384
RX D Buffer Descriptor Pointer Offset register 393
RX D Buffer Descriptor Pointer register 384
RX Free Buffer register 395

S

sample hash table code 397–401
scan_en_n truth/termination table 28
SDRAM boot algorithm 488–489
self-refresh mode 118
Serial Channel B/A/C/D Bit-rate register 624, 660
Serial Channel B/A/C/D Control Register A 611, 652
Serial Channel B/A/C/D Control Register B 614, 655
Serial Channel B/A/C/D FIFO Data register 629, 665
Serial Channel B/A/C/D Flow Control Force register 638
Serial Channel B/A/C/D Flow Control register 636
Serial Channel B/A/C/D Receive Buffer GAP Timer register 630
Serial Channel B/A/C/D Receive Character GAP Timer register 632
Serial Channel B/A/C/D Receive Match MASK register 635
Serial Channel B/A/C/D Receive Match register 634
Serial Channel B/A/C/D Status Register A 617, 657
serial controller
  port features 4
  UART features 4
serial controller, SPI 643–666
  bit-rate generator 645
  features 644
  FIFO management 647–650
  receive FIFO interface 648
  Serial Channel B/A/C/D Bit-rate register 660
  Serial Channel B/A/C/D Control Register A 652

Serial Channel B/A/C/D Control register B 655
Serial Channel B/A/C/D FIFO Data register 665
Serial Channel Status Register A 657
serial port control and status registers 650–666
serial port performance 650
SPI timing diagrams 827–830
structure 644
transmit FIFO interface 647
serial controller, UART 601–639
bit-rate examples 628–629
bit-rate generator 603
features 602
FIFO management 605–608
receive FIFO interface 606
transmit FIFO interface 605
framing structure 604
functions 605
Serial Channel B/A/C/D Bit-rate register 624
Serial Channel B/A/C/D Control Register A 611
Serial Channel B/A/C/D Control Register B 614
Serial Channel B/A/C/D FIFO Data register 629
Serial Channel B/A/C/D Flow Control Force register 638
Serial Channel B/A/C/D Flow Control register 636
Serial Channel B/A/C/D Receive Buffer GAP Timer register 630
Serial Channel B/A/C/D Receive Character GAP Timer register 632
Serial Channel B/A/C/D Receive Match MASK register 635
Serial Channel B/A/C/D Receive Match register 634
Serial Channel B/A/C/D Status Register A 617
serial port control and status registers 608–639
serial port performance 608
serial EEPROM boot 10
Serial Interface Engine 713
serial port control and status registers 608–639, 650–666
SPI register addresses, serial controllers B and A 650
SPI register addresses, serial controllers C and D 651
UART register addresses, serial controllers B and A 608
UART register addresses, serial controllers C and D 610
serial port maximum rates, UART 608
serial port performance 608, 650
SHOULD BE ONE, SBO (CP15 registers) 53
SHOULD BE ZERO or PRESERVED, SBZP (CP15 registers) 53
SHOULD BE ZERO, SBZ (CP15 registers) 53
signals
external pad interface 575
LCD module 42
PCI/CardBus 31
system memory interface 22
signals and interrupts, LCD 566
single combined interrupt 566
Slave Address register 551
slave module commands, I2C 546
slew rates, USB PHY 712
Software Watchdog Configuration register 291
software watchdog timer 263
Software Watchdog Timer register 293
source address 475, 505

SPI mode 646
  individual mode definition 647
  individual modes 646
  See also *serial controller, SPI.*
SPI timing 827–830
SPI-EEPROM boot logic 484–489
  ARM (memory) boot configuration 486
  example 485
  memory controller configuration 486
spike filter 552
SPLIT transfers 258
state machine 502
static (context) RAM 502
Static Memory Configuration 0-3 registers 230
static memory controller 121–162
  configurations 121
  extended wait transfers 122
  initialization 123–148
  memory mapped peripherals 123
  write protection 122
static memory controller, access sequencing and memory width 123
Static Memory Extended Wait register 224
static memory initialization 123–148
  bus turnaround 143–148
  static memory read control 124–136
  static memory write control 136–143
  wait state generation 123
Static Memory Output Enable Delay (0-3) registers 235
Static Memory Page Mode Read Delay 0-3 registers 237
static memory read control 124–136
Static Memory Read Delay 0-3 registers 236
Static Memory Turn round Delay 0-3 registers 239
static memory write control 136–143
Static Memory Write Delay 0-3 registers 238
Static Memory Write Enable Delay 0-3 registers 234
Station Address Filter register 366
station address logic (SAL) 321
Station Address registers 364
statistics module 321
Statistics registers (Ethernet) 368
  combined transmit and receive statistics counters 368
  receive statistics counters 369
  transmit statistics counters 373
status bit, Ethernet 326, 328
status field 506
Status Receive Data register 549
Status register, memory 207
STN display panels 564
STN displays 565
synchronization logic 568
system address map 261
system boot
  BBus bridge function 468
system bus 5
system bus arbiter 254–260
  configuration examples 258
  CPU arbiter, high-speed 255
  CPU subsystem 255
  locked bus sequence 257
  main arbiter, high-speed 256
  ownership 257
  procedure 256
  relinquishing the bus 257
  SPLIT transfers 258
system clock 13
system clock generation, PLL 272
system configuration registers 276–313
  register addresses 276

system control module 253–313
- Active Interrupt Level Status register 301
- AHB Arbiter Gen Configuration register 282
- bootstrap initialization 272–276
- BRC channel assignment 283
- BRC0, BRC1, BRC2, BRC3 registers 283
- bus arbiter configuration examples 258
- bus interconnection 254
- Clock Configuration register 293
- definition 253
- External Interrupt 0-3 Control register 313
- features 254
- Gen ID register 311
- general purpose timers/counters 263
- Int Config registers (0-31) 286
  - register address mapping 286
- interrupt controller 267–270
- Interrupt Status Active register 289
- Interrupt Status Raw register 290
- Interrupt Vector Address Register Level 0-31 285
- ISRADDR register 288
- Miscellaneous System Configuration and Status register 296
- PLL configuration 271
- PLL Configuration register 299
- programmable timers 263–264
- register addresses 276
- Reset and Sleep Control register 295
- Software Watchdog Configuration register 291
- software watchdog timer 263
- Software Watchdog Timer register 293
- SPLIT transfers 258
- system address map 261
- system attributes 271–276
- system bus arbiter 254–260
  - high-speed peripheral subsystem 255
- system configuration registers 276–313
- System Memory Chip Select 0 Dynamic Memory Base and Mask registers 303
- System Memory Chip Select 0 Static Memory Base and Mask registers 307
- System Memory Chip Select 1 Dynamic Memory Base and Mask registers 304
- System Memory Chip Select 1 Static Memory Base and Mask registers 308
- System Memory Chip Select 2 Dynamic Memory Base and Mask registers 305
- System Memory Chip Select 2 Static Memory Base and Mask registers 309
- System Memory Chip Select 3 Dynamic Memory Base and Mask registers 306
- System Memory Chip Select 3 Static Memory Base and Mask registers 310
- Timer 0-15 Control registers 301
- Timer 0-15 Read register 285
- Timer 0-15 Reload Count registers 284
- Timer Interrupt Status register 291
- vectored interrupt controller 270

system control module bootstrap configuration pins 273

system control processor (CP15) registers 51–77
- accessing 52
- ARM926EJ-S system addresses 51

R0, ID code and cache type status registers 55–57
R1, Control register 58–60
R10, TLB Lockdown register 73
R11 register 74
R12 register 74
R13, Process ID register 75–77
R14 register 77
R15, Test and debug register 77
R2, Translation Table Base register 61
R3, Domain Access Control register 61, 98
R4 register 62
R5, Fault Status registers 62
R6, Fault Address register 64, 97
R7, Cache Operations register 64–68
R8, TLB Operations register 68
R9, Cache Lockdown register 69–73
summary 53

System Memory Chip Select 0 Dynamic Memory Base and Mask registers 303
System Memory Chip Select 0 Static Memory Base and Mask registers 307
System Memory Chip Select 1 Dynamic Memory Base and Mask registers 304
System Memory Chip Select 1 Static Memory Base and Mask registers 308
System Memory Chip Select 2 Dynamic Memory Base and Mask registers 305
System Memory Chip Select 2 Static Memory Base and Mask registers 309
System Memory Chip Select 3 Dynamic Memory Base and Mask registers 306
System Memory Chip Select 3 Static Memory Base and Mask registers 310
system memory interface
  pinout 18
  signals 22
system timers 6
system-level interfaces 8

T

TFT display panels 564
TFT displays 565
Timer 0-15 Control registers 301
Timer 0-15 Read register 285
Timer 0-15 Reload Count registers 284
Timer Interrupt Status register 291
timing 787–839
  about 787
  absolute maximum ratings 788
  clock timing 837–839
  DC electrical characteristics 790–791
    inputs 790
    outputs 791
  electrical characteristics 788–789
  Ethernet timing 813–815
  I2C timing 821
  IEEE 1284 timing 831
  JTAG timing 836
  LCD timing 822–826
  maximum power dissipation 789
  memory timing 795–812
  PCI timing 816–820
  recommended operating conditions 788
  reset and hardware strapping 835
  SPI timing 827–830
  USB timing 832–834

timing controller, LCD 574
TLB structure 104
transaction ordering, AHB-to-PCI bridge 410
transferring a frame to system memory, Ethernet 325
translation faults 101
Translation Lookaside Buffer (TLB) 78
transmission error handling, USB 714
transmit broadcast packet counter 374
transmit buffer descriptor format
  Ethernet 327
Transmit Buffer Descriptor Pointer Offset register 394
transmit byte counter 374
transmit deferral packet counter 374
transmit excessive collision packet counter 375
transmit excessive deferral packet counter 375
transmit FCS error counter 376
transmit FIFO interface 605, 647
transmit fragment counter 377
transmit jabber frame counter 376
transmit late collision packet counter 375
transmit multicast packet counter 374
transmit multiple collision packet counter 375
transmit oversize frame counter 376
transmit packet counter 374
transmit packet processor 324, 327
Transmit Recover Buffer Descriptor Pointer register 389
transmit single collision packet counter 375
transmit statistics counters 373
transmit total collision packet counter 376
transmit undersize frame counter 377
transmitting a frame to the Ethernet MAC 330
TX Buffer Descriptor Pointer register 389
TX buffer descriptor RAM 396
TX Error Buffer Descriptor Pointer register 390
typical power dissipation 789

U

UART
  features 4
UART mode 604
  See also *serial controller, UART.*
UNDEFINED (CP15 registers) 53
UNPREDICTABLE (CP15 registers) 52
up/down counters 263
upper and lower panel formatters 574
upstream port (USB) 708
upstream transactions 405
USB clock 15
USB Configuration register 538
USB controller 707–783
  about 708
  DC electrical inputs 790
  DC electrical outputs 791
  device block 710–712
  device block architecture 710
  Device Control and Status register 718
  device endpoint 714
  device endpoint status 770
  Device IP Programming Control/Status register 724
  FIFO Interrupt Enable 0 register 776
  FIFO Interrupt Enable 1 register 777
  FIFO Interrupt Enable 2 register 778
  FIFO Interrupt Enable 3 register 779
  FIFO Interrupt Enable registers 776–

780
FIFO Interrupt Status 0 register 771
FIFO Interrupt Status 1 register 771
FIFO Interrupt Status 2 register 773
FIFO Interrupt Status 3 register 775
FIFO Interrupt Status registers 769–775
FIFO Packet Control registers 780
FIFO Status and Control registers 781
Global Interrupt Enable register 720
Global Interrupt Status register 721
HcBulkCurrentED register 744
HcBulkHeadED register 743
HcCommandStatus register 730
HcControl register 727
HcControlCurrentED register 742
HcControlHeadED register 741
HcDoneHead register 746
HcFmInterval register 747
HcFmNumber register 749
HcFmRemaining register 748
HcHCCA register 739
HcInterruptDisable register 737
HcInterruptEnable register 735
HcInterruptStatus register 733
HcLsThreshold register 751
HcPeriodCurrentED register 740
HcPeriodicStart register 750
HCRevision register 726
HcRhDescriptorA register 753
HcRhDescriptorB register 755
HcRhPortStatus1 register 759
HcRhStatus register 756
host block 712–714
module architecture 708
port features 4
register "block" addresses 716
registers 716–783
device block 765–767
device block addresses 765
Device Endpoint FIFO Control and Data register addresses 767
Device Endpoint FIFO Control and Data registers 767–783
global 716–725
Global Control and Status 717
global register addresses 716
host block 725–764
host block addresses 725
host block reserved bits 725
host block, root hub partition registers 752
root hub 752
transmission error handling 714
USB timing diagrams 832–834
USB interface pinout 43
USB timing 832–834

V

vector interrupt controller 6
vectored interrupt controller (VIC) 270
vertical axis panel 582
vertical compare interrupt (VCOMPINTR) 598

W

wait state generation 123
water level marks 568
WRAP bit 476, 505
WRAP bit, Ethernet 326, 327
write protection
dynamic memory controller 162
static memory controller 122

Digi®